ICT365

Software Development Frameworks

Dr Afaq Shah

Murdoch
UNIVERSITY

Unit Testing

Murdoch
UNIVERSITY

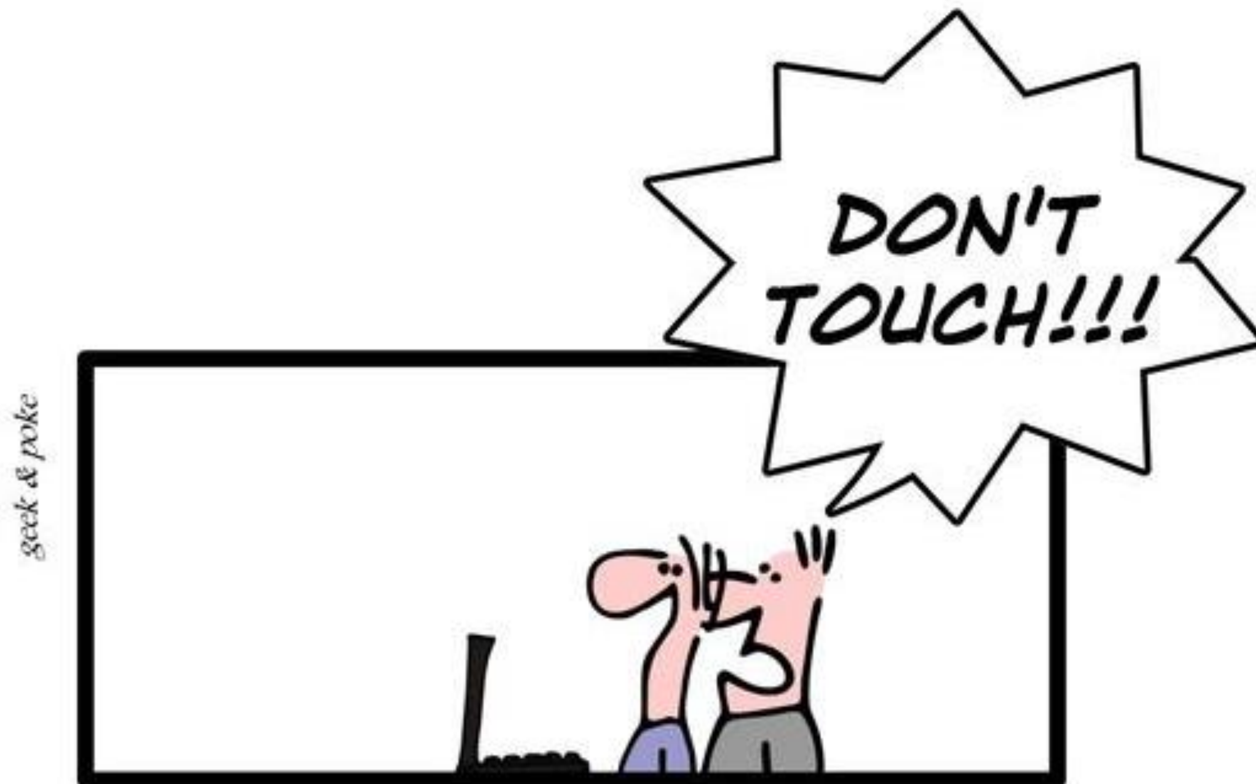
In this Topic

- What is Unit Testing?
- Advantages
- Disadvantages
- nUnit
- nUnitAsp
- nUnitForms
- Demo
- DotNetMock & nMock
- nCover
- What is TDD?
- Test-Driven UI Development
- TestDriven .NET
- Resources

Best Practices for Agile/Lean Documentation

- http://www.agilemodeling.com/essays/agileDocumentationBestPractices.htm
- Ideally, an agile document is just barely good enough, or just barely sufficient, for the situation at hand.
- Documentation is an important part of agile software development projects, but unlike traditionalists who often see documentation as a risk reduction strategy, agilists typically see documentation as a strategy which increases overall project risk and therefore strive to be as efficient as possible when it comes to documentation.
- Agilists write documentation when that's the best way to achieve the relevant goals, but there often proves to be better ways to achieve those goals than writing static documentation.

GOOD CODE IS...



... LIKE A MING VASE: BEAUTIFUL BUT FRAGILE

EDIT

COMPILE

RUN

UNDO

Tests are a validation tool…

Introduction

Murdoch UNIVERSITY

- Check that your code is working as expected by creating and running unit tests.
- It's called unit testing because you break down the functionality of your program into discrete testable behaviors that you can test as individual *units*.

- Run unit tests with Test Explorer
- https://msdn.microsoft.com/en-us/library/hh270865.aspx
- How to: Run Tests from Microsoft Visual Studio
- https://msdn.microsoft.com/en-us/library/ms182470.aspx
- Quick Start: Test Driven Development with Test Explorer
- https://msdn.microsoft.com/en-us/library/hh212233.aspx

Introduction

Murdoch UNIVERSITY

- Visual Studio Test Explorer provides a flexible and efficient way to run your unit tests and view their results in Visual Studio.
- Visual Studio installs the Microsoft unit testing frameworks for managed and native code.
- Use a *unit testing framework* to create unit tests, run them, and report the results of these tests. Rerun unit tests when you make changes to test that your code is still working correctly.
- When you use Visual Studio Enterprise, you can run tests automatically after every build.

Code quality

Murdoch UNIVERSITY

- Unit testing has the greatest effect on the quality of your code when it’s an integral part of your software development workflow.
- As soon as you write a function or other block of application code, create unit tests that verify the behavior of the code in response to standard, boundary, and incorrect cases of input data, and that check any explicit or implicit assumptions made by the code.
- With *test driven development*, you create the unit tests before you write the code, so you use the unit tests as both design documentation and functional specifications.

Unit Testing

Murdoch University

nUnit, nUnitAsp, nUnitForms

What is Unit Testing?

Murdoch University

Unit Test

From Wikipedia, the free encyclopedia.

In computer programming, a **unit test** is a procedure used to verify that a particular module of source code is working properly. The idea about unit tests is to write test cases for all functions and methods so that whenever a change causes a regression, it can be quickly identified and fixed. Ideally, each test case is separate from the others; constructs such as mock objects can assist in separating unit tests. This type of testing is mostly done by the developers and not by end-users.

Advantages

Murdoch UNIVERSITY

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

Unit testing allows the programmer to re-factor code at a later date, and make sure the module still works correctly (i.e. regression testing). This provides the benefit of encouraging programmers to make changes to the code since it is easy for the programmer to check if the piece is still working properly.

Advantages

Unit testing helps eliminate uncertainty in the pieces themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

Unit testing provides a sort of "living document". Clients and other developers looking to learn how to use the class can look at the unit tests to determine how to use the class to fit their needs and gain a basic understanding of the API.

Advantages

Because some classes may have references to other classes, testing a class can frequently spill over into testing another class. A common example of this is classes that depend on a database: in order to test the class, the tester often writes code that interacts with the database. This is a mistake, because a unit test should never go outside of its own class boundary. As a result, the software developer abstracts an interface around the database connection, and then implements that interface with their own mock object. This results in loosely coupled code, minimizing dependencies in the system

Disadvantages

Murdoch UNIVERSITY

Unit-testing will not catch every error in the program. By definition, it only tests the functionality of the units themselves. Therefore, it will not catch integration errors, performance problems and any other system-wide issues. In addition, it may not be easy to anticipate all special cases of input the program unit under study may receive in reality. Unit testing is only effective if it is used in conjunction with other software testing activities.

It is unrealistic to test all possible input combinations for any non-trivial piece of software. A unit test can only show the presence of errors; it cannot show the absence of errors.

Unit Testing Frameworks

www.junit.org

www.nunit.org

www.xprogramming.com

Characteristics of UTFs

Murdoch UNIVERSITY

- Most UTFs target OO and web languages
- UTFs encourage separation of business and presentation logic
- Tests written in same language as the code
- Tests are written against the business logic
- GUI and command line test runners
- Rapid feedback

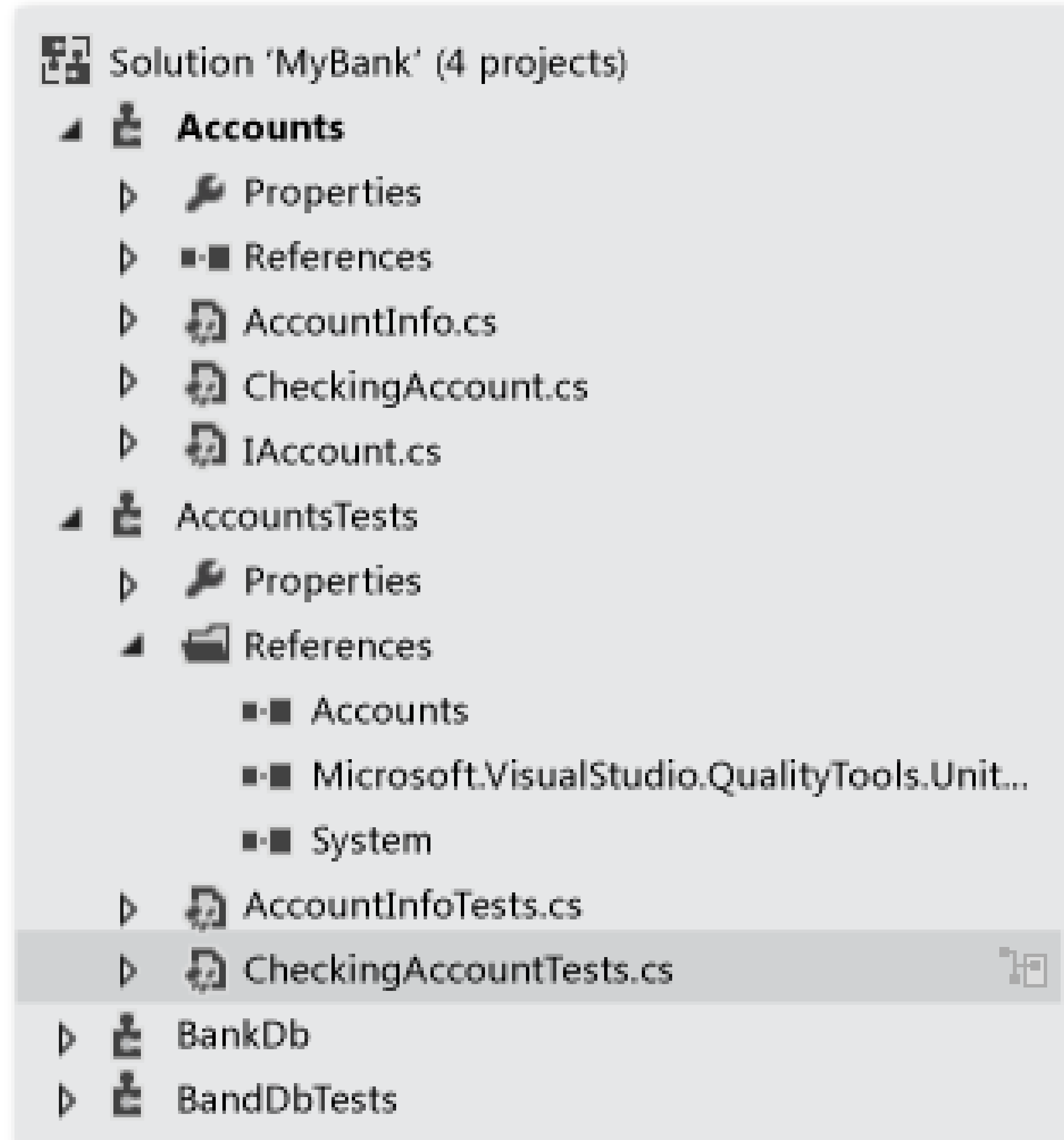
Plugins and Extensions

Murdoch UNIVERSITY

- You can quickly generate test projects and test methods from your code, or manually create the tests as you need them.
- When you use IntelliTest to explore your .NET code, you can generate test data and a suite of unit tests. For every statement in the code, a test input is generated that will execute that statement.
- Test Explorer can also run third-party and open source unit test frameworks that have implemented Test Explorer add-on interfaces. You can add many of these frameworks through the Visual Studio Extension Manager and the Visual Studio gallery.
- Install third-party unit test frameworks
- https://msdn.microsoft.com/en-au/library/hh598952.aspx

Example

- **MyBank Solution**
- Fictional application called MyBank as an example.
- Test methods are written in C# and presented by using the Microsoft Unit Testing Framework for Managed Code,
- Concepts are easily transferred to other languages and frameworks.



First attempt at a design

Murdoch UNIVERSITY

Includes an accounts component that represents an individual account and its transactions with the bank, and a database component that represents the functionality to aggregate and manage the individual accounts.

- We create a MyBank solution that contains two projects:
- Accounts
- BankDb

First attempt at designing the Accounts project

Murdoch UNIVERSITY

- Contains a class to hold basic information about an account, an interface that specifies the common functionality of any type of account, like depositing and withdrawing assets from the account, and a class derived from the interface that represents a checking account.
- We begin the Accounts projects by creating the following source files: …

First attempt at designing the Accounts project

Murdoch UNIVERSITY

- AccountInfo.cs defines the basic information for an account.
- IAccount.cs defines a standard IAccount interface for an account, including methods to deposit and withdraw assets from an account and to retrieve the account balance.
- CheckingAccount.cs contains the CheckingAccount class that implements the IAccounts interface for a checking account.

First attempt at designing the Accounts project

- We know from experience that one thing a withdrawal from a checking account must do is to make sure that the amount withdrawn is less than the account balance.
- So we override the IAccount.Withdaw method in CheckingAccount with a method that checks for this condition.

The method might look like this:

Murdoch UNIVERSITY

```
public void Withdraw(double amount)
{
        if(m_balance >= amount)
         {
                m_balance -= amount;
        }
        else
         {
                throw new ArgumentException(amount,
                                "Withdrawal exceeds balance!")
         }
 }
```

Now that we have some code, it’s time for testing.

Create unit test projects and test methods

Murdoch UNIVERSITY

It is often quicker to generate the unit test project and unit test stubs from your code.

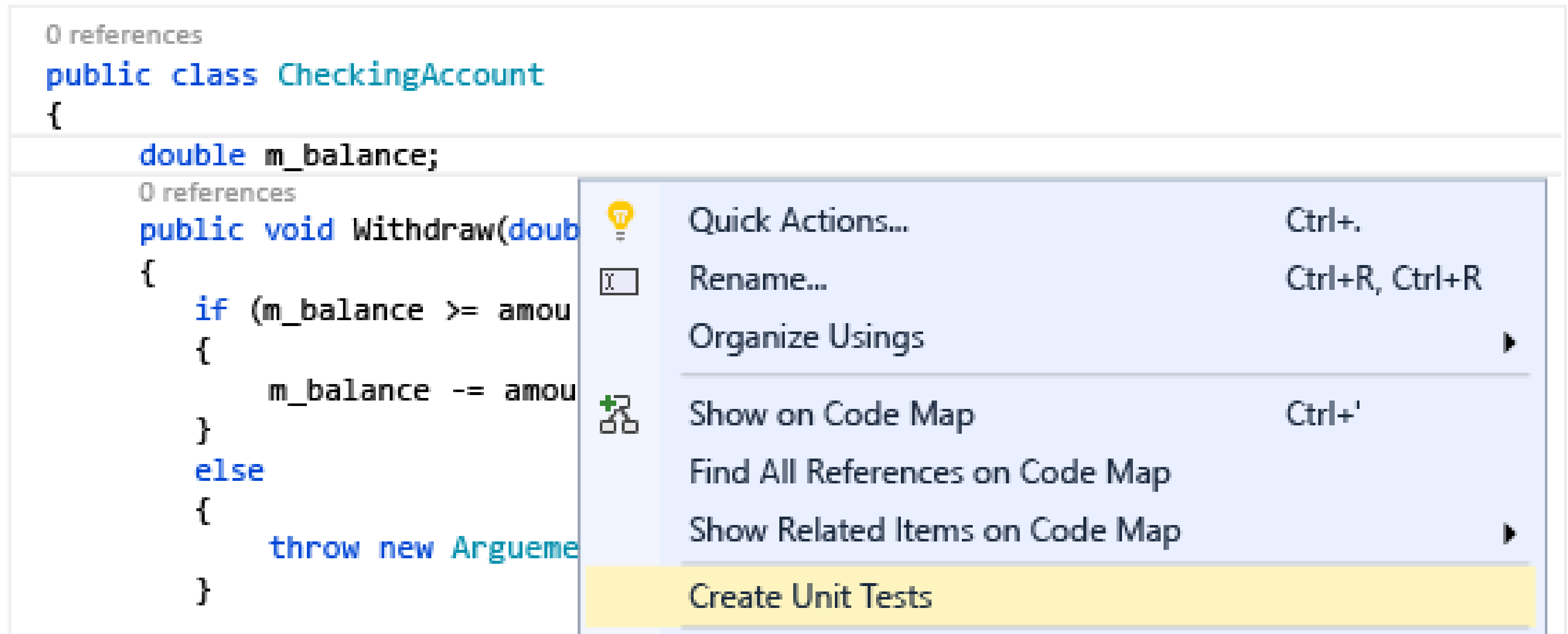
Or you can choose to create the unit test project and tests manually depending on your requirements.

Generate unit test project and unit test stubs

Create unit test projects and test methods

Murdoch University

From the code editor window, right-click and choose **Create Unit Tests** from the context menu



Click OK to accept the defaults to create your unit tests, or change the values used to create and name the unit test project and the unit tests.

You can select the code that is added by default to the unit test methods.

Create Unit Tests

Test Framework:	MSTest
Test Project:	<New Test Project>
Name Format for Test Project:	[Project]Tests
Namespace:	[Namespace].Tests
Output File:	<New Test File>
Name Format for Test Class:	[Class]Tests
Name Format for Test Method:	[Method]Test
Code for Test Method:	Assert failure

OK Cancel

Murdoch
UNIVERSITY

The unit test stubs are created in a new unit test project for all the methods in the class.

CheckingAccountTests.cs*

AccountsTests | Accounts.Tests.Check | WithdrawTest()

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Accounts;
using System;
using System.Collections.Generic;
using System.Lynq;
using System.Text;
using System.Threading.Tasks;

namespace Accounts.Tests
{
    [TestClass()]
    0 references
    public class CheckingAccountTests
    {
        [TestMethod()]
        0 references
        public void WithdrawTest()
        {
            Assert.Fail();
        }
    }
}
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'MyBank' (2 projects)
  - Accounts
    - Properties
    - References
    - CheckingAccount.cs
  - AccountsTests
    - Properties
    - References
      - Analyzers
      - Accounts
      - Microsoft.VisualStudio.Quality.To...
      - System
    - CheckingAccountTests.cs

Next

- Add code to the unit test methods to make the unit test meaningful, and any extra unit tests that you might want to add to thoroughly test your code.

Create your unit test project and unit tests manually

- A unit test project usually mirrors the structure of a single code project. In the MyBank example, you add two unit test projects named AccountsTests and BankDbTests to the MyBanks solution. The test project names are arbitrary, but adopting a standard naming convention is a good idea.
- To add a unit test project to a solution:
- On the **File** menu, choose **New** and then choose **Project** (Keyboard Ctrl + Shift + N).
- On the New Project dialog box, expand the **Installed** node, choose the language that you want to use for your test project, and then choose **Test**.

Create your unit test project and unit tests manually

- To use one of the Microsoft unit test frameworks, choose **Unit Test Project** from the list of project templates. Otherwise, choose the project template of the unit test framework that you want to use.
- To test the Accounts project of our example, you would name the project AccountsTests.

Reference another code project

Murdoch UNIVERSITY

•In your unit test project, add a reference to the code project under test, in our example to the Accounts project.

To create the reference to the code project:

1. Select the project in Solution Explorer.
2. On the **Project** menu, choose **Add Reference**.
3. On the Reference Manager dialog box, open the **Solution** node and choose **Projects**. Select the code project name and close the dialog box.

- Each unit test project contains classes that mirror the names of the classes in the code project. In our example, the AccountsTests project would contain the following classes:
- AccountInfoTests class contains the unit test methods for the AccountInfo class in the BankAccount project
- CheckingAccountTests class contains the unit test methods for CheckingAccount class.

Write your tests

Murdoch UNIVERSITY

- The unit test framework that you use and Visual Studio IntelliSense will guide you through writing the code for your unit tests for a code project. To run in Test Explorer, most frameworks require that you add specific attributes to identify unit test methods. The frameworks also provide a way—usually through assert statements or method attributes—to indicate whether the test method has passed or failed. Other attributes identify optional setup methods that are at class initialization and before each test method and teardown methods that are run after each test method and before the class is destroyed.

Arrange, Act, Assert

Murdoch UNIVERSITY

- The AAA (Arrange, Act, Assert) pattern is a common way of writing unit tests for a method under test.
- The Arrange section of a unit test method initializes objects and sets the value of the data that is passed to the method under test.
- The Act section invokes the method under test with the arranged parameters.
- The Assert section verifies that the action of the method under test behaves as expected.

To test the CheckingAccount.Withdraw method of our example, we can write two tests:

one that verifies the standard behavior of the method, and

one that verifies that a withdrawal of more than the balance will fail.

In the CheckingAccountTests class, we add the following methods:

```
[TestMethod]
public void Withdraw_ValidAmount_ChangesBalance()
 {
       // arrange
      double currentBalance = 10.0;
      double withdrawal = 1.0; double expected = 9.0;
      var account = new CheckingAccount("JohnDoe", currentBalance);
      // act
      account.Withdraw(withdrawal); double actual = account.Balance;
      // assert Assert.AreEqual(expected, actual); }

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Withdraw_AmountMoreThanBalance_Throws()
 {
      // arrange
      var account = new CheckingAccount("John Doe", 10.0);
      // act
      account.Withdraw(20.0);
      // assert is handled by the ExpectedException
}
```

- Note that Withdraw_ValidAmount_ChangesBalance uses an explicit Assert statement to determine whether the test method passes or fails, while Withdraw_AmountMoreThanBalance_Throws uses the ExpectedException attribute to determine the success of the test method. Under the covers, a unit test framework wraps test methods in try/catch statements.
- In most cases, if an exception is caught, the test method fails and the exception is ignored. The ExpectedException attribute causes the test method to pass if the specified exception is thrown.

- For more information about the Microsoft Unit Testing Frameworks, see one of the following topics:

Writing Unit Tests for the .NET Framework with the Microsoft Unit Test Framework for Managed Code

- https://msdn.microsoft.com/en-au/library/hh598960.aspx

Writing Unit tests for C/C++ with the Microsoft Unit Testing Framework for C++

- https://msdn.microsoft.com/en-au/library/hh598953.aspx

Set timeouts for unit tests

Murdoch UNIVERSITY

- To set a timeout on an individual test method:

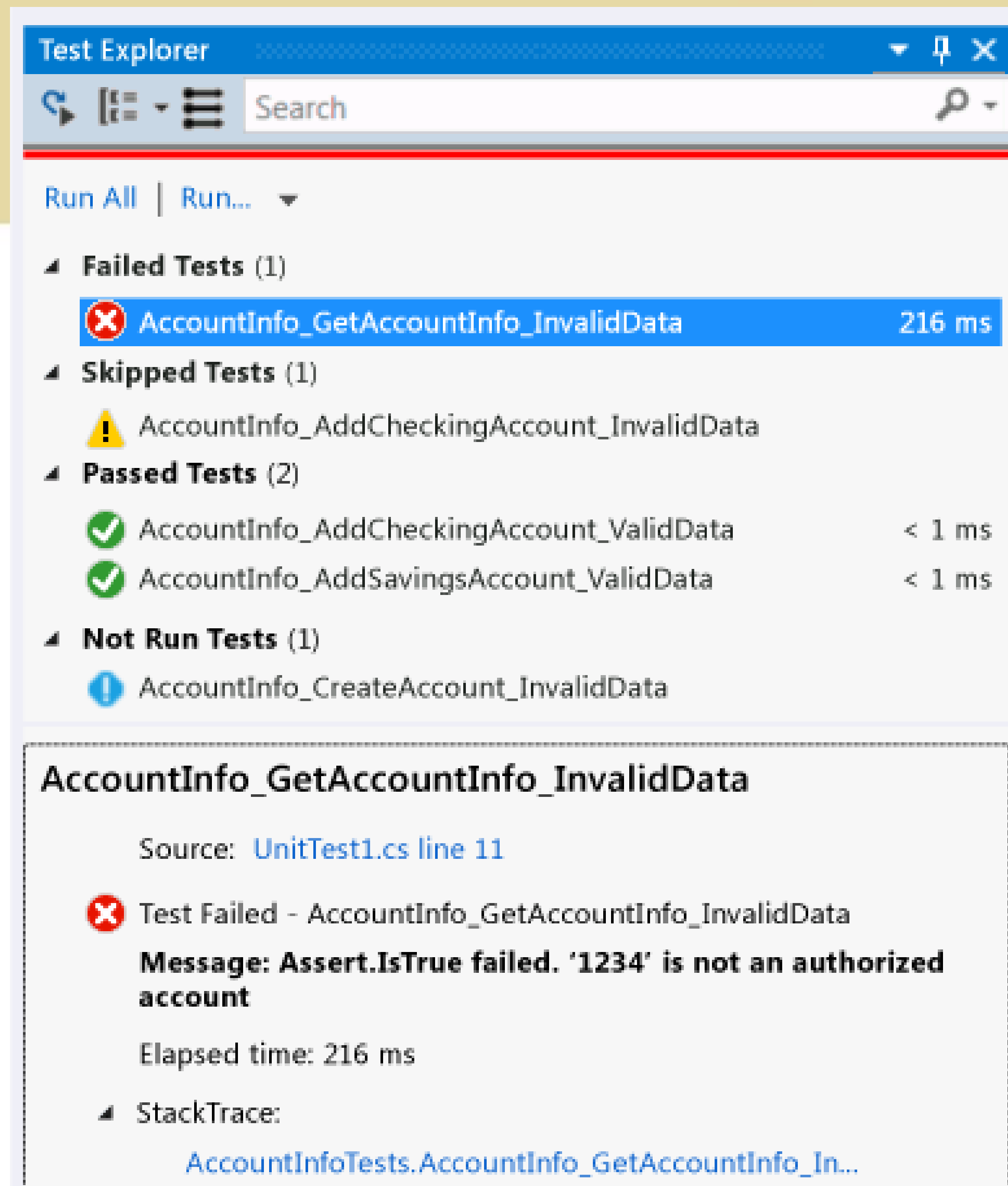
```
[TestMethod]
[Timeout(2000)]
// Milliseconds
public void My_Test()
{ ... }
```

- To set the timeout to the maximum allowed:

```
[TestMethod]
 [Timeout(TestTimeout.Infinite)]
// Milliseconds
 public void My_Test () { ... }
```

Run tests in Test Explorer

When you build the test project, the tests appear in Test Explorer. If Test Explorer is not visible, choose **Test** on the Visual Studio menu, choose **Windows**, and then choose **Test Explorer**.

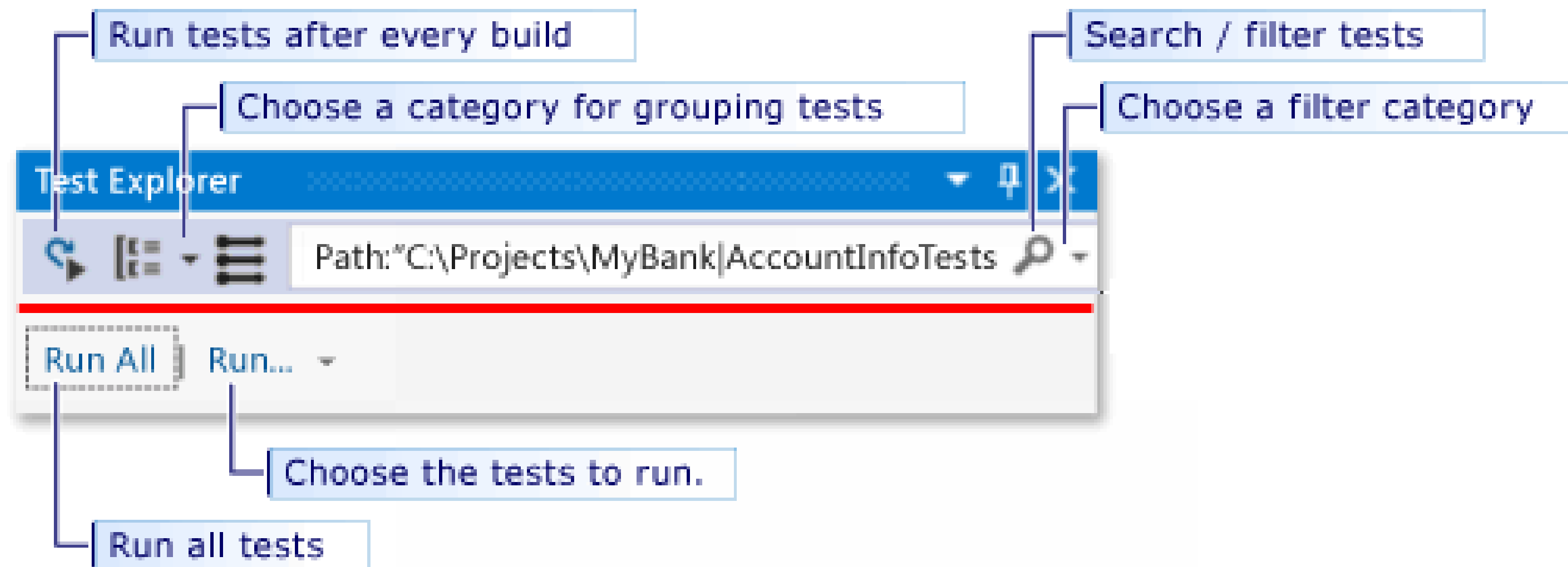


- As you run, write, and rerun your tests, the default view of Test Explorer displays the results in groups of **Failed Tests**, **Passed Tests**, **Skipped Tests** and **Not Run Tests**.
- You can choose a group heading to open the view that displays all them tests in that group.

- You can also filter the tests in any view by matching text in the search box at the global level or by selecting one of the pre-defined filters.
- You can run any selection of the tests at any time.
- The results of a test run are immediately apparent in the pass/fail bar at the top of the explorer window.
- Details of a test method result are displayed when you select the test.

Run and view tests

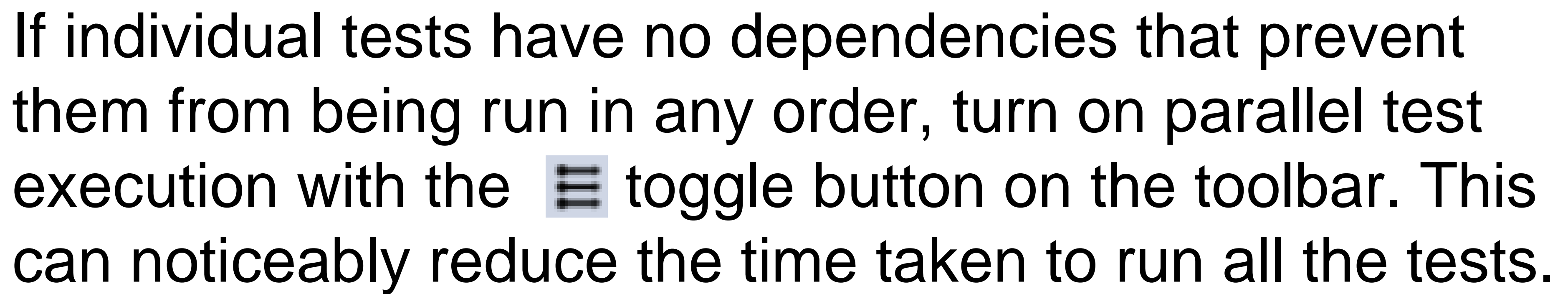
The Test Explorer toolbar helps you discover, organize, and run the tests that you are interested in.



Murdoch
UNIVERSITY

You can choose **Run All** to run all your tests, or choose **Run** to choose a subset of tests to run. After you run a set of tests, a summary of the test run appears at the bottom of the Test Explorer window.

Select a test to view the details of that test in the bottom pane. Choose **Open Test** from the context menu (Keyboard: F12) to display the source code for the selected test.



If individual tests have no dependencies that prevent them from being run in any order, turn on parallel test execution with the toggle button on the toolbar. This can noticeably reduce the time taken to run all the tests.

Run tests after every build

Murdoch University

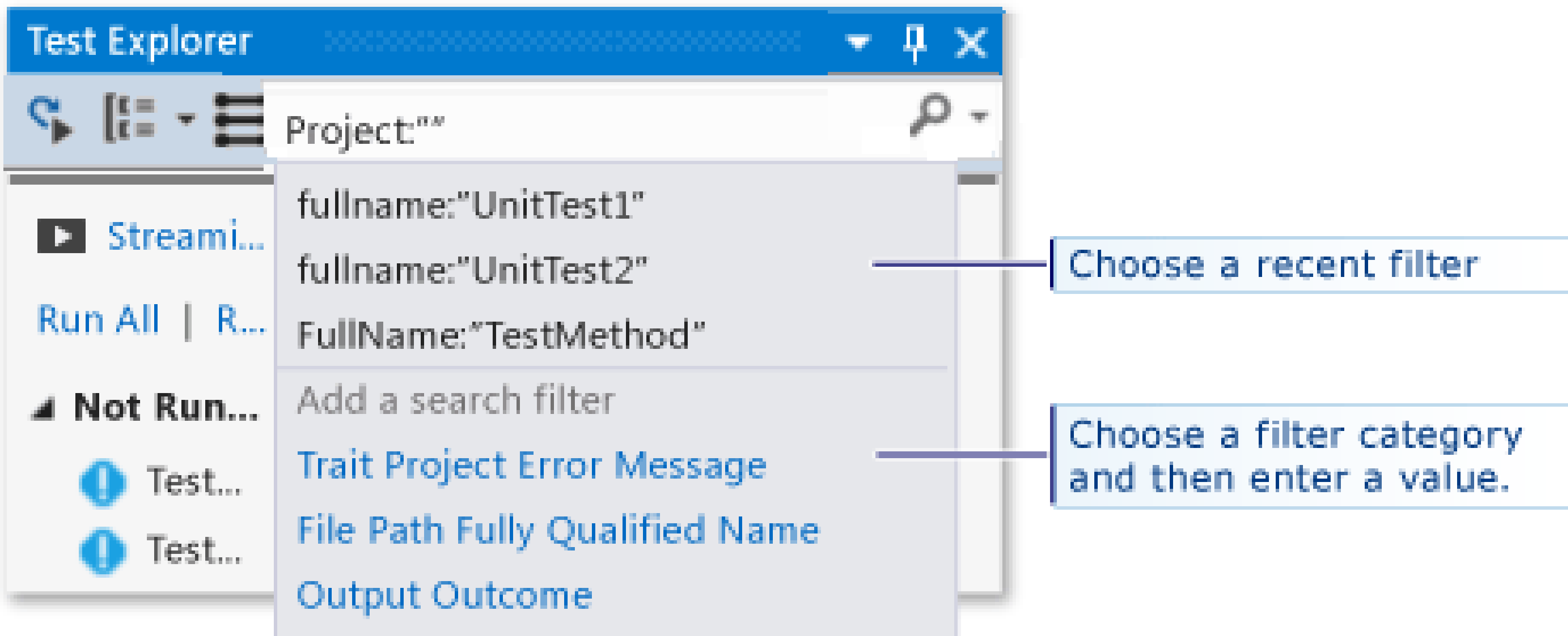
- To run your unit tests after each local build, choose **Test** on the standard menu, choose **Run Tests After Build** on the Test Explorer toolbar.

- Running unit tests after every build is supported only in Visual Studio Enterprise.

Filter and group the test list

Murdoch University

When you have a large number of tests, you can Type in Test Explorer search box to filter the list by the specified string. You can restrict your filter event more by choosing from the filter list.



To group your tests by category, choose the **Group By** button.

Q: If I'm using TDD, how do I generate code from my tests?

Murdoch UNIVERSITY

A: Use IntelliSense to generate classes and methods in your project code. Write a statement in a test method that calls the class or method that you want to generate, then open the IntelliSense menu under the call.

Q: If I'm using TDD, how do I generate code from my tests?

If the call is to a constructor of the new class, choose **Generate new type** from the menu and follow the wizard to insert the class in your code project. If the call is to a method, choose **Generate new method** from the IntelliSense menu.

```
accountInfo.ChangeAddress(newAddress);
```

Generate method stub for 'ChangeAddress' in 'MyBank.AccountInfo'

Q: Can I create unit tests that take multiple sets of data as input to run the test?

A: Yes. *Data-driven test methods* let you test a range of values with a single unit test method. Use a DataSource attribute for the test method that specifies the data source and table that contains the variable values that you want to test. In the method body, you assign the row values to variables using the TestContext.DataRow[*ColumnName*] indexer.

For example, assume we add an unnecessary method to the CheckingAccount class that is named AddIntegerHelper. AddIntegerHelper adds two integers.

To create a data-driven test for the AddIntegerHelper method, we first create an Access database named AccountsTest.accdb and a table named AddIntegerHelperData. The AddIntegerHelperData table defines columns to specify the first and second operands of the addition and a column to specify the expected result. We fill a number of rows with appropriate values.

The attributed method runs once for each row in the table. Test Explorer reports a test failure for the method if any of the iterations fail. The test results detail pane for the method shows you the pass/fail status method for each row of data.

```
[DataSource( @"Provider=Microsoft.ACE.OLEDB.12.0;
Data Source=C:\Projects\MyBank\TestData\AccountsTest.accdb",
"AddIntegerHelperData" )]
 [TestMethod()]
public void AddIntegerHelper_DataDrivenValues_AllShouldPass()
{
        var target = new CheckingAccount();
        int x = Convert.ToInt32(TestContext.DataRow["FirstNumber"]);
        int y = Convert.ToInt32(TestContext.DataRow["SecondNumber"]);
        int expected = Convert.ToInt32(TestContext.DataRow["Sum"]);
        int actual = target.AddIntegerHelper(x, y);
        Assert.AreEqual(expected, actual);
}
```

Data-driven unit tests

https://msdn.microsoft.com/en-au/library/ms182527.aspx

Q: Can I view how much of my code is tested by my unit tests?

A: Yes. You can determine the amount of your code that is actually being tested by your unit tests by using the Visual Studio code coverage tool. Native and managed languages and all unit test frameworks that can be run by the Unit Test Framework are supported.

You can run code coverage on selected tests or on all tests in a solution. The Code Coverage Results window displays the percentage of the blocks of product code that were exercised by line, function, class, namespace and module.

Using Code Coverage to Determine How Much Code is being Tested

Murdoch UNIVERSITY

To determine what proportion of your project's code is actually being tested by coded tests such as unit tests, you can use the code coverage feature of Visual Studio.

To guard effectively against bugs, your tests should exercise or 'cover' a large proportion of your code.

Code coverage analysis can be applied to both managed (CLI) and unmanaged (native) code.

TEST ARCHITECTURE ANALYZE WIN...

Run
Debug
Test Settings
Analyze Code Coverage
Windows

Selected Tests
All Tests

Test Explorer

Search

Run All | Run...

Passed Tests (3)

Test	Time
QuickNonZero	15 ms
RootTestNeg...	13 ms
SignatureTest	1 ms

```
public double SquareRoot(double x)
{
    if (x < 0.0)
    {
        throw new ArgumentOutOfRangeException();
    }
    double estimate = x;
    double previousEstimate = -x;
    while (System.Math.Abs(estimate - previousEstimate) >...
    {
```

Not covered

Covered

Turn on coloring

100 %

Code Coverage Results

ctsoasm_MAIN50531 2012-06-07 02...

Hierarchy	Not Cov...	Not Covered (%...	Cov...
ctsoasm_MAIN50531 201...	44	80.00%	11
fabrikam.math.dll	7	50.00%	7
{ } Fabrikam.Math	7	50.00%	7

Code coverage is an option when you run test methods using Test Explorer. The results table shows the percentage of the code that was run in each assembly, class, and method. In addition, the source editor shows you which code has been tested.

Q: Can I view how much of my code is tested by my unit tests?

Murdoch
UNIVERSITY

To run code coverage for test methods in a solution, choose **Tests** on the Visual Studio menu and then choose **Analyze code coverage**.

Coverage results appear in the Code Coverage Results window.

Code Coverage Results

user1_FABLAB-0529 2012-06-04 10_40...

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)
AccountInfoTests	3	25.00%
AccountInfo_GetAccountInfo_InvalidData()	1	50.00%
AccountInfo_GetAccount_ValidData()	0	0.00%
AccountInfo_GetAccount_InvalidData()	0	0.00%
AccountInfo_AddSavingsAccount_ValidData()	0	0.00%
AccountInfo_AddCheckingAccount_ValidDa...	0	0.00%

Code coverage

https://msdn.microsoft.com/en-au/library/dd537628.aspx

Q: How can I test methods in my code that have external dependencies?

Murdoch UNIVERSITY

A: Yes. If you have Visual Studio Enterprise, Microsoft Fakes can be used with test methods that you write by using unit test frameworks for managed code.

Microsoft Fakes uses two approaches to create substitute classes for external dependencies.

Stubs generate substitute classes derived from the parent interface of the target dependency class. Stub methods can be substituted for public virtual methods of the target class.

Shims use runtime instrumentation to divert calls to a target method to a substitute shim method for non-virtual methods.

Q: How can I test methods in my code that have external dependencies?

Murdoch University

In both approaches, you use the generated delegates of calls to the dependency method to specify the behavior that you want in the test method.

Isolating unit test methods with Microsoft Fakes

https://msdn.microsoft.com/en-au/library/hh549175.aspx

Q: Can I use other unit test frameworks to create unit tests?

Murdoch
UNIVERSITY

A: After you restart Visual Studio, reopen your solution to create your unit tests, and then select your installed frameworks here:

Your unit test stubs will be created using the selected framework.

Find and install other frameworks

https://msdn.microsoft.com/en-au/library/hh598952.aspx

Create Unit Tests

Test Framework:	MSTest
Test Project:	<New Test Project>
Name Format for Test Project:	[Project]Tests
Namespace:	[Namespace].Tests
Output File:	<New Test File>
Name Format for Test Class:	[Class]Tests
Name Format for Test Method:	[Method]Test
Code for Test Method:	Assert failure

OK Cancel

nunit.org

News | Download | Documentation | Contact | Twitter | Gitter | GitHub

Download

What Is NUnit?

NUnit is a unit-testing framework for all .Net languages. Initially ported from JUnit, the current production release, version 3, has been completely rewritten with many new features and support for a wide range of .NET platforms.

License

NUnit is Open Source software and NUnit 3 is released under the MIT license. Earlier releases used the NUnit license.

Both of these licenses allow the use of NUnit in free and commercial applications and libraries without restrictions.

About Us

NUnit 3 was created by Charlie Poole, Rob Prouse, Simone Busoli, Neil Colvin and numerous community contributors.

Earlier versions of NUnit were developed by Charlie Poole, James Newkirk, Alexei Vorontsov, Michael Two and Philip Craig.

$ Donations

The NUnit team invests a great deal of time and effort to make NUnit a useful tool. In addition, we have expenses. We have to purchase domain names, arrange for web site hosting and acquire equipment.

Financial contributions are one way you can help to ensure that NUnit continues to develop and remains free and open software. For more information or to view a list of donors, see our Donations page.

nUnit for ASP.net

nUnitAsp is a class library for use within your NUnit tests. It providesNUnit with the ability to download, parse, and manipulate ASP.NET web pages.

http://nunitasp.sourceforge.net/quickstart.html

With nUnitAsp, your tests don't need to know how ASP.NET renders controls into HTML. Instead, you can rely on the nUnitAsp library to do this for you, keeping your test code simple and clean. For example, your tests don't need to know that a DataGrid control renders as an HTML table. You can rely on nUnitAsp to handle the details. This gives you the freedom to focus on functionality questions, like whether the DataGrid holds the expected values.

ASP.NET MVC Model Testing using NUnit and MOQ

http://www.dotnetcurry.com/aspnet-mvc/1103/aspnet-mvc-model-testing-using-nunit-moq

Testing an ASP.NET MVC Controller using NUnit

http://www.dotnetcurry.com/aspnet-mvc/1110/testing-mvc-controller-using-nunit

User Interfaces

Murdoch UNIVERSITY

- Difficult to write automated tests to, because they are designed to be exercised by a user and often hide their programmatic interface.
- Inherent challenges in testing Windows Forms applications:

Are highly coupled

.Exe assemblies cannot be referenced from the IDE

NUnitForms

- NUnit Extension for testing Windows Forms applications

nUnitForms

NunitForms
windows.forms unit testing

Murdoch University

nUnitForms is an nUnit extension for unit and acceptance testing of Windows Forms applications

nUnit test can open a window and interact with the controls. The test will automatically manipulate and verify the properties of the GUI. nUnitForms takes care of cleaning up the forms between test, detecting and handling modal dialog boxes, and verifying that your expectations for the test are fullfilled.

http://nunitforms.sourceforge.net/

Steps to test a WinForms App

Murdoch UNIVERSITY

1. Place the forms in a separate class library
2. Create an Application Launcher that executes the forms
3. Reference the class library that contains the forms in the Test assembly
4. Design the Form
5. Use NUnitForms to write the tests
6. Write the code to pass the tests

DotNetMock & nMock

Murdoch University

DotNetMock is a dynamic mock-object library for .NET. Its purpose is to provide a clean API for rapidly creating mock implementations of custom objects and verifying that objects interact with them correctly from unit tests.

NMock is a dynamic mock-object library for .NET. Its purpose is to provide a clean API for rapidly creating mock implementations of custom objects and verifying that objects interact with them correctly from unit tests.

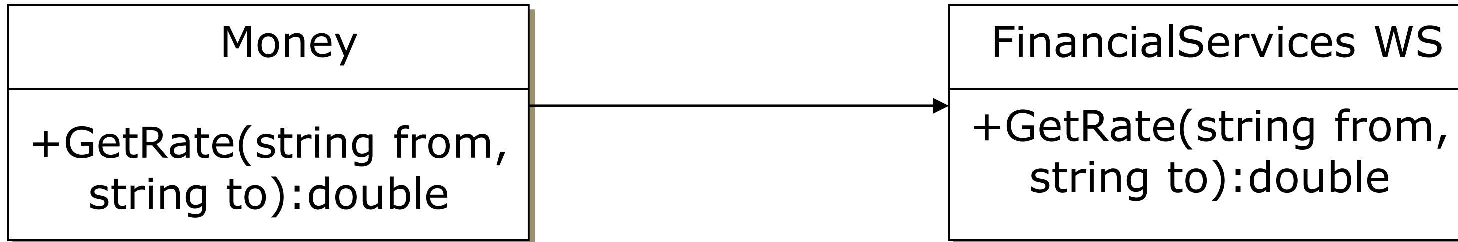
Mock Objects

- Inherent challenges in testing dependant objects

Objects dependant on ephemeral objects produce unpredictable behavior

User Interfaces, Databases and the like are inherently ephemeral

Example

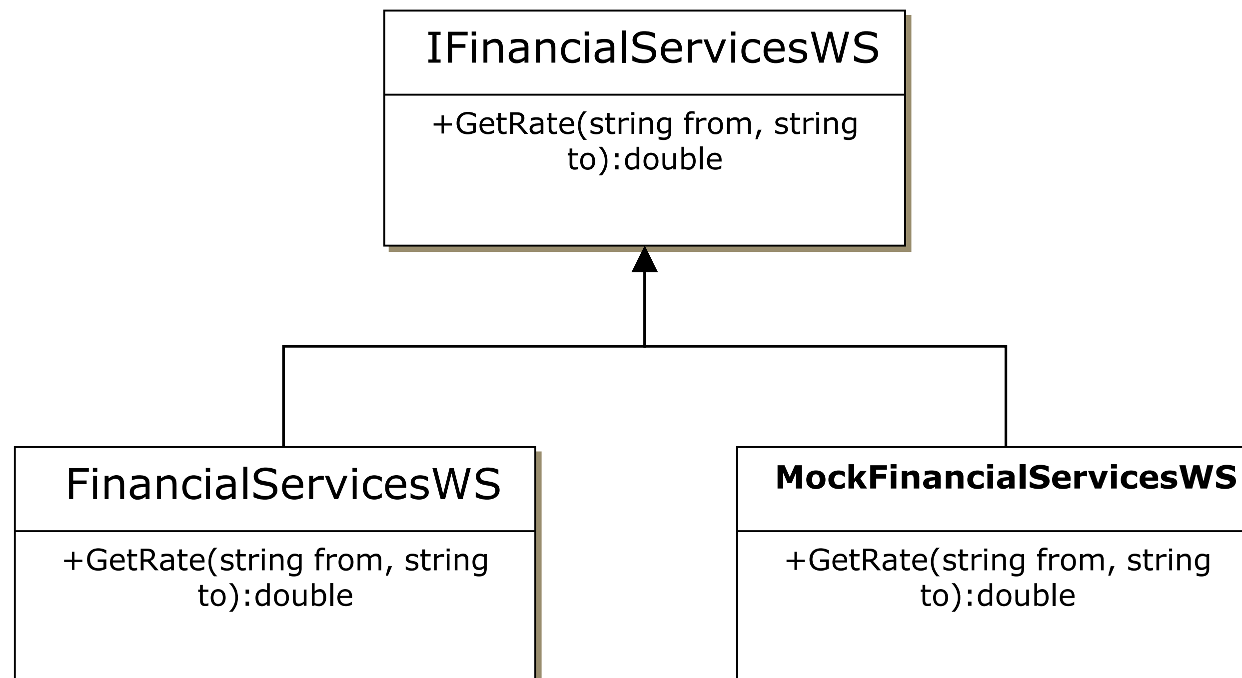


- How can we write a test for GetRate() when the result of FinancialServicesWS.GetRate() is unpredictable?

Mock Objects

- Solution:

Replace the unpredictable object with a testing version that produces predictable results



Steps to create a Dynamic Mock Object with DotNetMock

1. FinancialServicesWS is a proxy generated from the WSDL of the web service. Refactor the proxy to extract an interface: IFinancialServicesWS
2. Create a controller based on the extracted interface
3. Have the controller create a dynamic mock object
4. Specify the expected behavior of the mock object
5. Use the mock object in place of the real object in the test

Mock Objects

Murdoch UNIVERSITY

- Advantages of Mock Objects:

  Promote design to interfaces

  Promote testability and isolation of tests

  Promote decoupling

- Challenges of Mock Objects:

  More classes to maintain

  Requires breaking encapsulation to replace real object with mock object, sometimes resulting in less “elegant” code

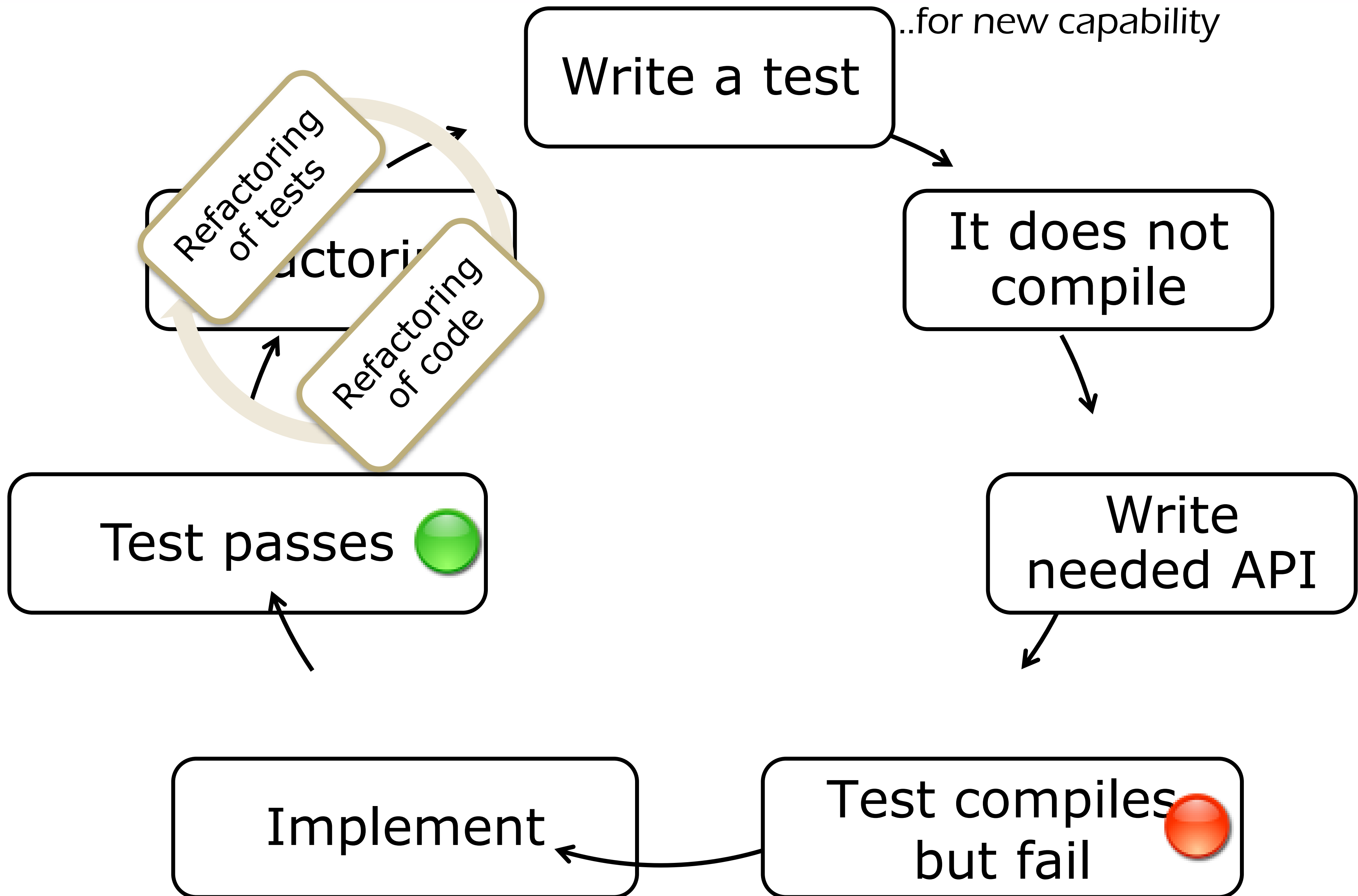
nCover

Murdoch UNIVERSITY

nCover provides statistics about your code, telling you how many times each line of code was executed during a particular run of the application. The most common use of code coverage analysis is to provide a measurement of how thoroughly your unit tests exercise your code. After running your unit tests under nCover, you can easily pinpoint sections of code that are poorly covered and write unit tests for those portions. Code coverage measurement is a vital part of a healthy build environment.

Test Driven Development Cycle

Red/Green/Refactor



www.testdriven.net

HOME | PRODUCT | QUICKSTART | DOWNLOAD | PURCHASE | SUPPORT | CONTACT

Home

→ Buy Now

Welcome to TestDriven.Net

The Zero Friction Unit Testing Extension for Visual Studio

TestDriven.Net makes it easy to run unit tests with a single click, anywhere in your Visual Studio solutions. It supports all versions of Microsoft Visual Studio and it integrates with the best .NET development tools.

Latest release

TestDriven.NET-4.1.3531 Beta is available here.

(now compatible with Visual Studio 2017)

Supports

- NUnit
- xUnit
- MSTest
- MbUnit
- TypeMock
- NCover

TestDriven.NET makes it easy to run unit tests with a single click, anywhere in your Visual Studio solutions. It supports *all* versions of Microsoft Visual Studio .NET meaning you don't have to worry about compatibility issues and fully integrates with all major unit testing frameworks including nUnit, MbUnit, & MS Team System.

Mechanics of TDD

Murdoch UNIVERSITY

- Always start with a failing test
- Quickly write the simplest code needed to pass the test
- Remove duplication (AKA Refactor)
- Repeat as needed to meet requirements
- Test everything that could possibly break

What can be tested?

Murdoch University

- Valid inputs
- Invalid inputs
- Errors, exceptions, and events
- Boundary conditions
- Everything that could possibly break!

TDD Benefits for Developers

- Much less debug time
- Code proven to meet requirements
- Tests become Safety Net
- Eliminate Bug Pong
- Rhythm of Success

TDD Benefits for Business

Murdoch University

- Shorter development cycles
- Near zero defects
- Tests become an asset
- Tests are documentation
- Competitive advantage!

Getting started with TDD

Murdoch University

- Get some training
- Start with a small visible project
- Shoot for 100% test coverage
- Don't expect to be perfect
- Expect to improve dramatically in time
- Measure the results

TDD in Legacy Code

Murdoch University

- Go Hunting for Bear
- Build the Safety Net
- Work in small increments
- Expect to slow down, then speed up
- Measure the results

Got Bugs?

Murdoch UNIVERSITY

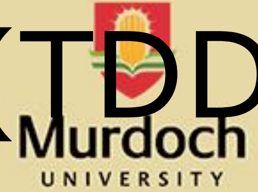
- Go after your most painful defect
- Write a test to expose it
- Write code needed to fix it
- Refactor
- Repeat
- Measure the results

Tips for Success

Murdoch University

- TDD is a skill
- Just Do It
- Insist on achieving 100% coverage
- Measure and Review
- Make Results Visible
- Just Keep Doing It

Test-Driven Development (TDD)



- Is a programming practice
- Unit tests are written before the domain code
- Namely:

  Write a test that fails

  Write code to make the test pass

  Refactor
- Unit Tests and Refactoring are the tools of TDD

Unit Tests

Murdoch University

- Test specific aspects of a functionality
- Execute rapidly
- Independent of each other
- Independent of the surrounding environment
- Independent of execution order
- Automated

Refactoring

Murdoch University

- Change the internal structure of the code without changing it’s external behavior.
- Associated with arithmetic factorization:

$$ab + ac = a(b+c)$$

- Same result but the expression is simplified.

Characteristics of TDD

Murdoch University

- TDD promotes code testability
- Testability is related to:

  Strong cohesion

  Loose coupling

  No redundancy

  Encapsulation
- These are good programming practices
- Striving for good tests results in better code

TDD Tenets

Murdoch UNIVERSITY

- Never write a single line of code unless you have a failing unit test
- Eliminate Duplication (Refactor mercilessly)

Observation

Murdoch UNIVERSITY

- It’s harder to write unit tests for components located at the “edge” of the system:

Web Services

Data Access Layer

User Interface

Conclusion

Murdoch UNIVERSITY

- Advantages:

  Easy to implement

  Development of UI can be completely test-driven

  Promotes decoupling

  Enables creation of automated User Acceptance Tests

- Challenges:

  Setup requires somewhat complex wiring

Conclusion

Murdoch UNIVERSITY

- TDD let you use tests as
  - ✓ A validation tool
  - ✓ A documentation tool
  - ✓ A design tool

- Follow the TDD rules
  - ✓ Be iterative: split your work in small (requirement) increments
  - ✓ Always start by writing a test
  - ✓ Then implement untill the test passes
  - ✓ And Cleanup! *(refactor)*

C# Reference

Murdoch UNIVERSITY

- **Unit Testing Framework**
- https://msdn.microsoft.com/en-us/library/ms243147(VS.80).aspx
- **Unit Test Basics**
- https://msdn.microsoft.com/en-AU/library/hh694602.aspx
- **Writing Unit Tests for the .NET Framework with the Microsoft Unit Test Framework for Managed Code**
- https://msdn.microsoft.com/en-us/library/hh598960.aspx
- **Get started with developer testing tools**
- https://www.visualstudio.com/en-us/docs/test/developer-testing/getting-started/getting-started-with-developer-testing

Non-MSDN links

Murdoch UNIVERSITY

- **Visual Studio Unit Testing Framework - Wikipedia, the free encyclopedia**
- https://en.wikipedia.org/wiki/**Visual_Studio_Unit_Testing_Framework**
- **MSTest - Wikipedia, the free encyclopedia**
- https://en.wikipedia.org/wiki/MSTest
- **c# - Which unit testing framework? - Stack Overflow**
- http://stackoverflow.com/questions/2262090/which-unit-testing-framework

Java UI Testing

Murdoch University

- Start:

https://code.google.com/archive/p/fest/

- Extra:
- http://stackoverflow.com/questions/2575837/unit-testing-framework-for-a-swing-ui
- http://stackoverflow.com/questions/5939749/swing-ui-testing-library-comparisons-fest-windowtester-pro-etc
- http://alexruiz.developerblogs.com/?p=160
- Example:
- https://github.com/hakurai/fest-swing-example

Youtube

Murdoch UNIVERSITY

- **How to implement TDD (Test driven development) in c# (Csharp) using VSTS unit testing?**
- https://www.youtube.com/watch?v=5gMBGVNR8wE
- **How to write Unit Tests in C#**
- https://www.youtube.com/watch?v=8YFZBNFm0OM
- Excellent Java Eclipse tutorials: Mark Dexter

Resources

- Visual Studio .NET 2005

http://msdn.microsoft.com/vstudio/teamsystem

- Visual Studio .NET 2003 Project Templates for NUnit:

http://www.pontonetpt.com/Downloads/317.aspx

- NUnit

http://www.nunit.org

- DotNetMock

http://sourceforge.net/projects/dotnetmock

- NUnitForms

http://nunitforms.sourceforge.net/

- C# Refactory

http://www.extreme-simplicity.net

- ReSharper

http://www.jetbrains.com/resharper

Resources

Murdoch University

- nUnit http://nunit.org
- nUnitAsp http://nunitasp.sourceforge.net
- nUnitForms http://nunitforms.sourceforge.net
- DotNetMock http://dotnetmock.sourceforge.net
- nMock http://www.nmock.org/
- TestDriven .NET http://www.testdriven.net
- nCover http://http://www.ncover.org
- Open Source Testing http://opensourcetesting.org

References

Murdoch UNIVERSITY

- “Test-Driven Development By Example”, Kent Beck
- “Test-Driven Development in Microsoft .NET”, James Newkirk, Alexei Vorontsov
- “Refactoring, Improving The Design Of Existing Code”, Martin Fowler
- “Patterns Of Enterprise Application Architecture”, Martin Fowler
- “The Humble Dialog Box”, http://www.objectmentor.com/resources/articles/TheHumbleDialogBox.pdf

Other reading:

Murdoch UNIVERSITY

- *Extreme Programming Explained*, by Kent Beck
- Anything by Jon Bentley - *Programming Pearls*, *More Programming Pearls*, *Writing Efficient Programs*

Acknowledgements

Murdoch UNIVERSITY

Sources used in this presentation include:

- Refactoring and Code Maintenance by Marty Stepp
- Jay Smith   http://nwaDnug.org
- Harry Erwin