Unit Testing And Beyond Using Visual Studio 2005 Team Test

Guy Smith-Ferrier
Courseware Online
gsmithferrier@coursewareonline.com

Agenda

- Information Sources and Overview
- Unit Tests
- Code Coverage
- Manual Tests
- Generic Tests
- Web Tests
- Load Tests
- Static Analysis

Team System Information Sources

Visual Studio 2005 NewsGroups	http://communities.microsoft.com/newsgroups/default.asp?icp=whidbey&slcid=us
Visual Studio 2005 Team System Articles	
Overview	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-over.asp
Enabling Better Software Through Better Testing	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-test.asp
Designing Distributed Systems for Deployment	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-arch.asp
Building Robust and Reliable Software	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-dev.asp
Software Project Management	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-pm.asp
Enterprise-Class Source Control and Work Item Tracking	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-team.asp
Microsoft Solutions Framework	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-msf.asp
Extending the Suite	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvsent/html/vsts-ext.asp

MSDN Online Information Sources

- Team System Home Page
  - http://msdn.microsoft.com/vstudio/teamsystem
- MSDN TV “Introduction To Visual Studio Team System”
  - http://msdn.microsoft.com/msdntv
- MSDN Architecture Webcast: Test-Driven Development in Visual Studio 2005 Team System, Thursday 15 July 2004 7:00PM GST
  - msdn.microsoft.com/training/webcasts
- Improve the Design and Flexibility of Your Project with Extreme Programming Techniques
  - http://msdn.microsoft.com/msdnmag/issues/04/04/ExtremeProgramming/default.aspx

TDD Information Sources

- Test Driven Development in Microsoft .NET, James Newkirk, Alexei Vorontsov, Microsoft Press ISBN 0-7356-1948-4
- Test Driven Development, Kent Beck, Addison Wesley ISBN 0321146530
- Test Driven Development, A Practical Guide, David Astels

Other Online Information Sources

- James Newkirk's blog
  - http://weblogs.asp.net/jamesnewkirk
- How To Misuse Code Coverage
  - http://www.testing.com/writings/coverage.pdf
- http://workspaces.gotdotnet.com/tdd
- http://www.testdriven.com
- http://www.xprogramming.com

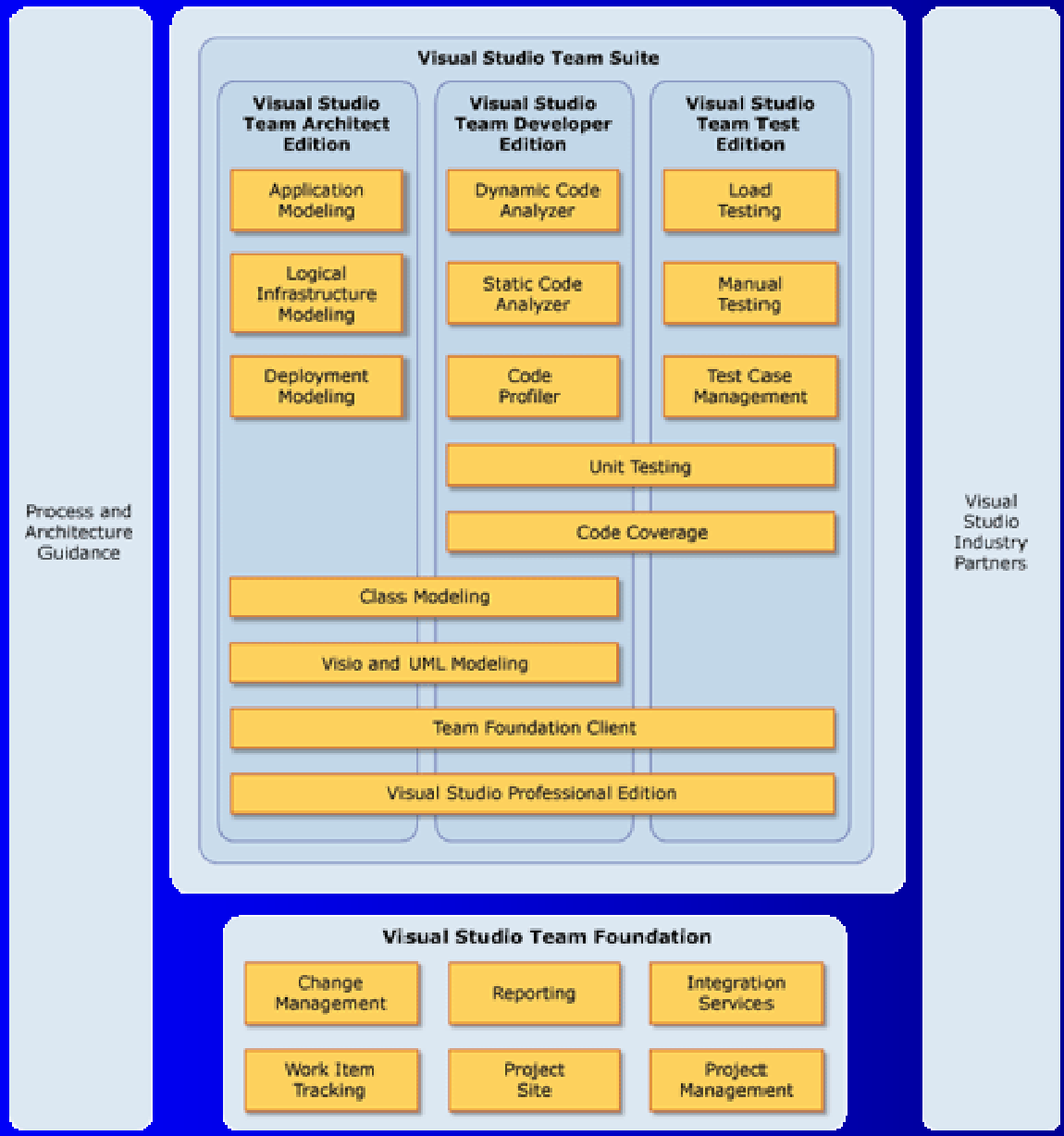
Unit Testing Tools For .NET Framework 1.1

Test Tool	Download	Description
NUnit	www.nunit.org	Test framework, GUI tool, Console tool
NUnitAddIn	sourceforge.net/projects/nunitaddin	VS.NET Add In for NUnit
NUnitForms	nunitforms.sourceforge.net	Testing Tool For WinForms
NUnitAsp	nunitasp.sourceforge.net	NUnit extensions for ASP.NET
csUnit	www.csunit.org	Test framework, VS.NET Add In, Console tool
MbUnit	www.mbunit.tigris.org	Test framework and more
ClrUnit	www.gotdotnet.com	Test framework, VS.NET Add In

Getting Team System

Visual Studio.NET 2005 Release	Released At	Release Date	Includes Team Foundation Client	Includes Team Foundation Server
PDC Release	PDC US 2003	Nov 2003	No	No
Community Technology Preview March 2004	?	March 2004	No	No
Community Technology Preview May 2004	Tech Ed US 2004	May 2004	Team Test	No
Beta 1	Tech Ed Europe 2004	July 2004	Team Architect	No ?
Community Technology Preview		Aug 2004	Yes ?	Yes ?
Beta 2		Dec 2004	Yes	Yes

Visual Studio Team System



Simple Demo

- Create a new ClassLibrary project
- Change the class to:-

```
public class Person
{
    public static string Greeting(
        string title, string firstName, string lastName)
    {
        return title + " " + firstName + " " + lastName;
    }
}
```

- Add a reference to Microsoft.VisualStudio.QualityTools.UnitTesting

Simple Demo (continued)

- Add the following using clause:-

```
using
Microsoft.VisualStudio.QualityTools.UnitTesting.Framework;
```

- Add a new class:-

```
[TestClass]
public class PersonTest
{
    [TestMethod]
    public void GreetingTest()
    {
        Assert.AreEqual("Mr Thomas Anderson",
        Person.Greeting("Mr", "Thomas", "Anderson"));
    }
}
```

Simple Demo (continued)

- View the Test View window (View | Quality Tools | Test View)
- Select the test and click on the run button
- The Test Results window will be displayed showing the result of the test

Unit Test Attributes

Team Test Attribute	Description	NUnit, csUnit & ClrUnit Equivalent
TestClass	Identifies a class which has test methods	TestFixture
TestMethod	Identifies a method which performs unit tests	Test
TestInitialize	Identifies a method which is called before every test	Setup
TestCleanup	Identifies a method which is called after every test	TearDown
TestProperty	?	n/a
ExpectedException	Test fails unless the specified exception is thrown	ExpectedException
ClassInitialize	Identifies a static method which is called once for each test class before all tests	TestFixtureSetup
ClassCleanup	Identifies a static method which is called once for each test class after all tests	TestFixtureTearDown
AssemblyInitialize		
AssemblyCleanup		
n/a	Ignores a unit test	Ignore

Assert Class

Team Test Assert	NUnit Assert	csUnit Assert	ClrUnit
AreEqual	AreEqual	Equals	AreEqual
AreNotEqual	n/a	NotEquals	AreDifferent
AreNotSame	n/a	n/a	n/a
AreSame	AreSame	n/a	AreSame
Collections	n/a	n/a	n/a
Equals	n/a	Equals	Equals
EqualsTests	n/a	n/a	n/a
Fail	Fail	Fail	Fail
GetHashCodeTests	n/a	n/a	n/a
Inconclusive	n/a	n/a	n/a
IsFalse	IsFalse	False	IsFalse
IsInstanceOfType	n/a	n/a	n/a
IsNotNull	IsNotNull	NotNull	IsNotNull
IsNull	IsNull	Null	IsNull
IsTrue	IsTrue	True	IsTrue
ReferenceEquals	ReferenceEquals	ReferenceEquals	ReferenceEquals
Text	n/a	n/a	n/a

Tests That Fail

- Add a new test to the test class:-

```
[TestMethod()]
public void GreetingTest2()
{
    Assert.AreEqual("Mr Anderson",
        Person.Greeting("Mr", "", "Anderson"));
}
```

- Run the unit tests again and show that the test fails (because there are too many spaces in the result)
- Change the implementation to:-

Tests That Fail (continued)

```
public static string Greeting(
    string title, string firstName, string lastName)
{
    string output = String.Empty;
    if (title != String.Empty)
        output = title;

    if (firstName != String.Empty)
    {
        if (output == String.Empty)
            output = firstName;
        else
            output += " " + firstName;
    }
```

Tests That Fail (continued)

```
    if (lastName != String.Empty)
    {
        if (output == String.Empty)
            output = lastName;
        else
            output += " " + lastName;
    }

    return output;
}
```

- Run the test again and show that the test passes

Test Projects

- Add a new Test Project to the solution (File | Add | New Project | Visual C# | Test Project, click OK, click Finish, in the “Add New Item” dialog click Cancel)
- In the Source Code Editor, right click on the Person class, select “Create Tests…”, in the “Output Project” dialog select “TestProject1”, click “Generate”
  - A new file, Class1Test.cs, will be created containing a PersonTest class
  - Examine the resulting code

Testing Private, Protected And Static Methods

- Add a new class, Human, which has a protected method called PlayFootball which returns a void
- Generate unit tests for the Human class and observe that the unit tests create the Human object using an “accessor” class (in CodeGenAccessors.cs)

Code Coverage

- In “Test View” select the Greeting unit test and click on the Run button
  - In the “Select a Test Run Configuration” dialog select the “Code Coverage” tab
    - Check the “Enable code coverage instrumentation” checkbox
    - Select ClassLibrary1.dll and click Add
    - Click OK

Code Coverage (continued)

- When the tests have completed show the source code for the Person class click the “Show code coverage” toolbutton in the top left above the Source Code Editor
  - Lines which were executed are shown in green, lines which were not are shown in red
- Add a new test which tests Person.Salutation("Mr", "", "Anderson"), re-run the unit tests and show that more of the red lines are now green

Test Driven Development

- Add a new unit test for the following non-existent method

```
Person.Address(
    "12 Avenue Des Anglais", "", "Cardiff", "Wales");
```

- Right click the method and select Refactor | Generate Method Stub
  - An Address method will be added to the Person class

Problems With The Community Technology Preview

- Problem: Several controls (such as “Enable code coverage instrumentation”) in the “Select a Test Run Configuration” dialog are not enabled
  - Solution: Search for localtestrun.rcg in the Visual Studio directory, copy it to the solution directory and add it to the solution
- Problem: Code coverage is not being recalculated for each successive test
  - Solution: The “Enable code coverage instrumentation” check box does not remember its setting. You must enable it each and every time.
- Problem: “Refactor” does not work in VB.NET
  - Solution: Use C#

Manual Tests

- In Solution Explorer right click the test project, select Add | New Item… | Text Manual Test and type:-

```
<<Letterhead Paper Print Test>>
Description
<<Test to ensure that the standard letter print
  option prints correctly on letterheaded paper>>
Test Target
<<PrintTest>>
Version of Test Target
<<1.0>>
```

Manual Tests (continued)

```
Test Steps
<<1. Start application
2. Insert letterheaded paper in printer
3. Select File | Print... | OK
4. Check that standard letter is printed correctly
>>
```

- Save the test
- Run the manual test
- In the Test Results window double click the Pending test
- In the “ManualTest1” dialog click the “Pass” radio button and then click Apply

Generic Tests

- Generic tests are not included in CTP May 2004
- A generic test is a test which can be run as a command line utility
- There are many uses for generic tests:-
  - Compatibility with existing testing frameworks
    - NUnit, csUnit, ClrUnit, MbUnit etc.
  - Support for testing facilities which are not covered by Team Test
    - NUnitForms, XmlUnit, HtmlUnit, HttpUnit
  - Testing using your own utilities
    - Localization
- It will also be possible to create your own test types

Using NUnit 2.1 With VS 2005

- You can install NUnit with VS 2005 but when you run your tests you will get:-

```
System.BadImageFormatException : The format of the file
   'TestLibrary' is invalid.
```

- In NUnit's directory (C:\Program Files\NUnit V2.1\bin) change the startup section of nunit-gui.exe.config and nunit-console.exe.config to:-

```
<startup>
  <!-- Community Technology Preview May 2004 -->
  <requiredRuntime version="v2.0.40426" />
  <!– Beta 1 <requiredRuntime version="v2.0.40607" /> -->
</startup>
```

Web Tests

- Create a new Web Application (File | New | Web Site… | ASP.NET Web Site)
- Add a TextBox, a Button and a Label
- Add a Test Project (File | Add | New Project… | Test Project), click OK, click Finish, select Record a Web Test and click Add
  - In Internet Explorer's Address type:- `http://localhost:5892/WebApplication1/Default.aspx`
  - In TextBox1 type "Hello World"
  - Click Button1
  - In the Web Test Recorder click Save

Web Tests (continued)

- In Visual Studio click on the newly created WebTest1 node
  - Expand the second node (for default.aspx)
  - Right click the node and select Add Validation Rule
  - Click on the new "Select Class" rule and open the Properties Window
  - Set Class Name to ValidationRuleFindText
  - Set TextToFind to "World"
  - Click on the Run button
  - In the player click on the Play button and the test will fail
- Add the following Click event to the button:-

```
Label1.Text = TextBox1.Text;
```

- Re-run the test and it will pass

Load Tests

- In Solution Explorer, right click TestProject1, select Add | New Item…, select Load Test, click Add
  - Click Next
  - Click the "Add" button (on the left)
    - In the "Add Tests" dialog check the checkbox next to WebTest1 and click OK
  - Check the check box next to the newly added WebTest1
  - Click Next
  - Click Next
  - In "Select browser(s) to add to the mix" check the checkbox next to IE6
  - Click Next four times and the click Finish

Static Analysis

- Static analysis tools analyze code without running the code
- Visual Studio 2005 will include two static analysis tools:- PREfast, FxCop
- PREfast
  - Not included with any beta yet
  - Used with C++ only
  - Used to test problems including null pointers, some buffer overruns and failing to check HResults

Static Analysis (continued)

- FxCop
  - Included with CTP May 2004 (but not Beta 1)
  - Available for .NET 1.1 from www.gotdotnet.com
  - Used with any .NET language
  - Used to test conformance to coding/development standards
  - Can be customized with new rules
  - To enable select the project's Properties window, select Build and enable the Run FxCop checkbox
  - To configure select the project's Properties window and select Rules

Summary

- Information Sources and Overview
- Unit Tests
- Code Coverage
- Manual Tests
- Generic Tests
- Web Tests
- Load Tests
- Static Analysis