ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

502045

Software Engineering

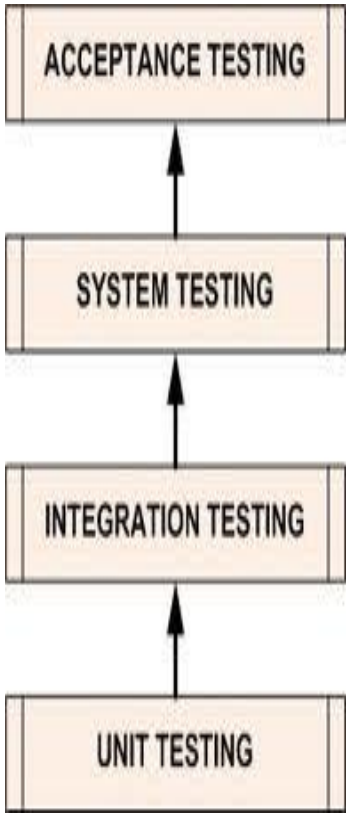
Chapter 08

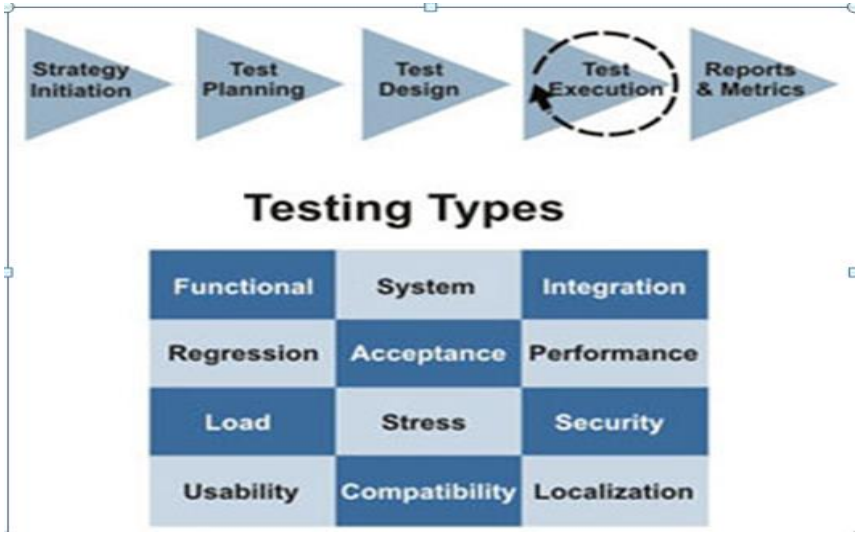
Lesson 11: Testing

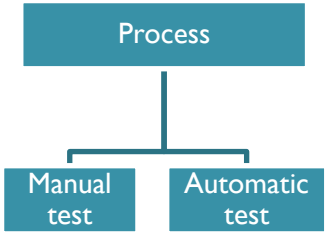
Contents

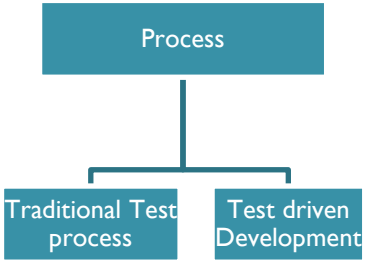
- Software testing levels
- Manual unit testing
- Unit Testing based on UT cases
- Automated Unit Testing
- Automated Unit Testing with NUnit
- Automated Tests vs. Manual Tests
- Best Practices

Too many of Software Testing Levels









How we test this function?

- Requirement :
  - Write a module to add an User to Database
- Business rule :
  - Email can not be duplicated
  - Email must be in valid form
  - UserName ‘s length must be > 8
  - UserName can not be dupplicated
  - Password length must be > 8

Manual Unit Testing

- Write code
- Uploading the code to some place
- Build it
- Running the code manually (in many cases filling up forms etc step by step)
- Check Log files, Database, External Services, Values of variable names, Output on the screen etc
- If it does not work, repeat the above process

Manual Unit Testing - Limitation

- It depends on developer's memory
- The more test case, the less coverability of developer
- Many duplicate test cases
- Lack of test cases
- Team lead cannot review everything

Unit Testing based on UT cases

- Describe test cases on word or excel

Function Code	Function1	Function Name	Function A
Created By	<Developer Name>	Executed By	
Lines of code	100	Lack of test cases	-5
Test requirement	<Brief description about requirements which are tested in this function>		

Passed	Failed	Untested	N/A/B			Total Test Cases
0	0	15	5	1	1	15

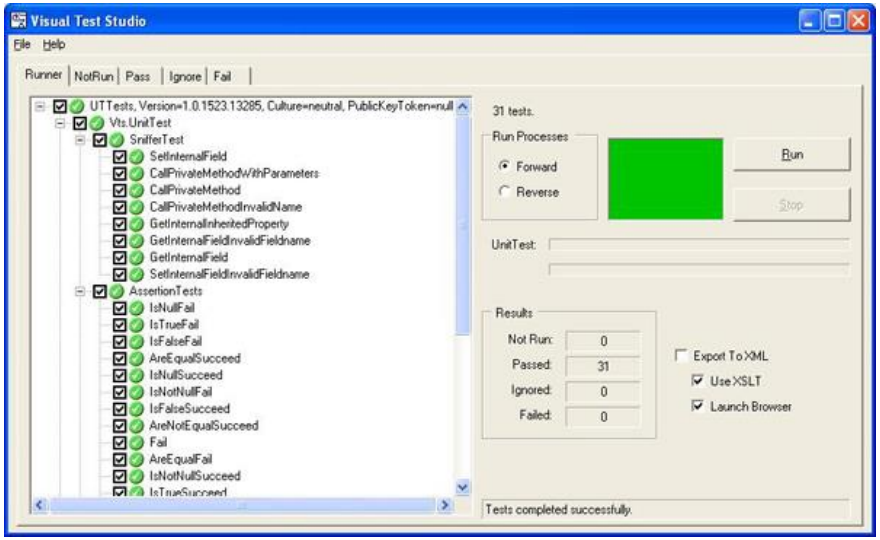
		UTCID01	UTCID02	UTCID02	UTCID02	UTCID02	UTCID02	UTCID07	UTCID08	UTCID09	UTCID10	UTCID11	UTCID12	UTCID13	UTCID14	UTCID15
Condition	Precondition															
	a															
	-2	O														
	-1							O								
	0		O	O	O											
	1					O	O									
	b															
	0		O	O												
	-2					O	O	O								
	2				O											
	c															
	0		O													
	1			O	O	O										
	3							O								
	5						O									
Confirm	Return															

Automated Unit Testing First Step

- Coding Process with Automated Unit Tests
  - Write code
  - Write one or more test cases script
  - Auto-compile and run
  - If tests fail -> make appropriate modifications
  - If tests pass -> repeat for next method

Automated Unit Testing Common Tools

- UT Tools for references:
  - Java: JUnit, J2MEUnit
  - C/C++: cppUnit
  - Python: pyUnit
  - Perl: PerlUnit
  - Visual Basic: vbUnit
  - C# .NET: Nunit,csUnit



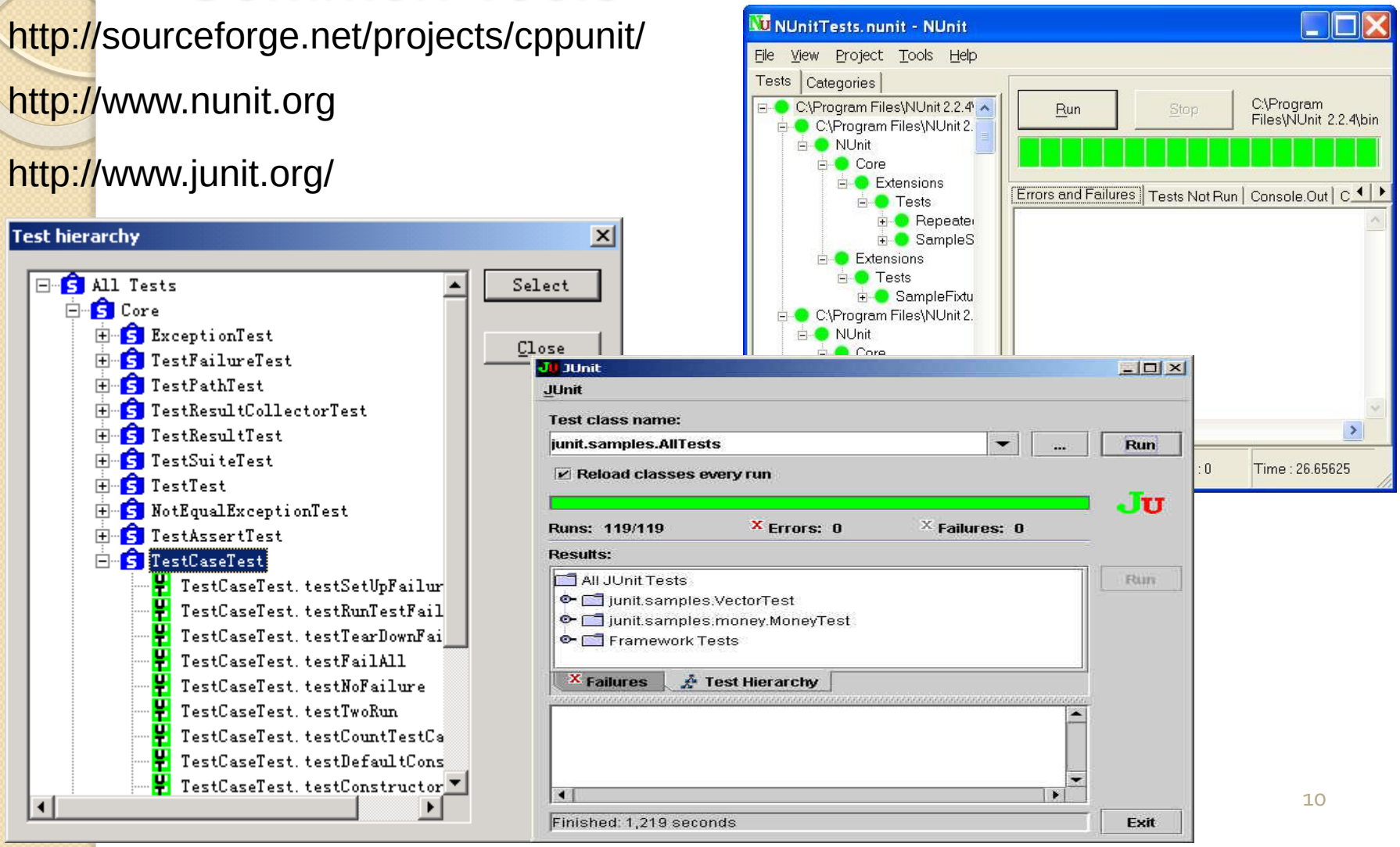
- Refferences:
  - http://www.testingfaqs.org/t-unit.html
  - www.junit.org
  - http://www.codeproject.com/gen/design/autp5.asp

Automated Unit Testing Common Tools

http://sourceforge.net/projects/cppunit/

http://www.nunit.org

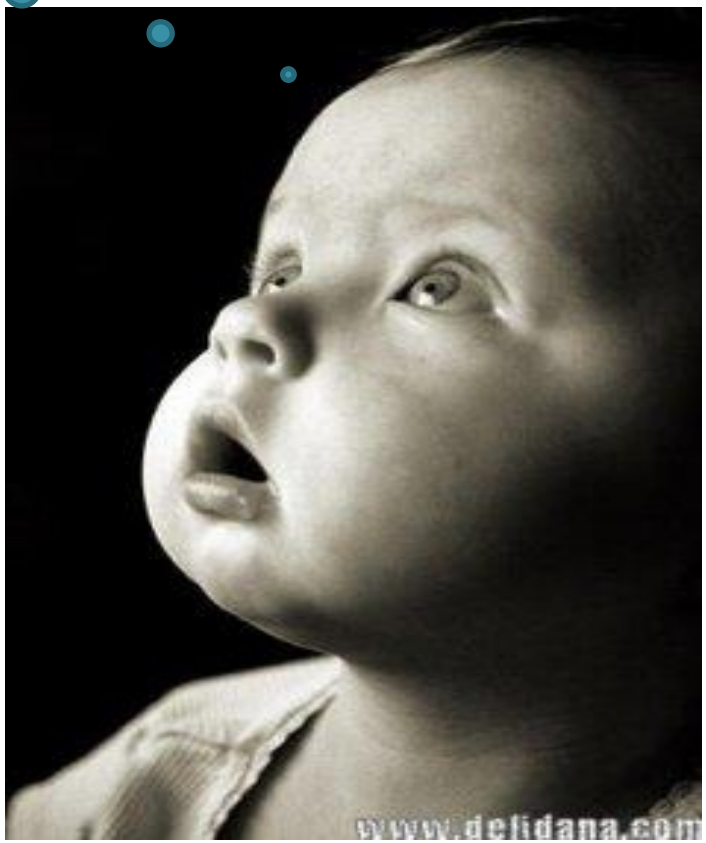
http://www.junit.org/



Automated Unit Testing Demo

Automated Unit Testing with NUnit
What is NUnit?

- NUnit – an open source test tool for .NET
- Useful for development and regression
- Leads to a design-for-test approach
- Tests can be written in VB.NET or C#



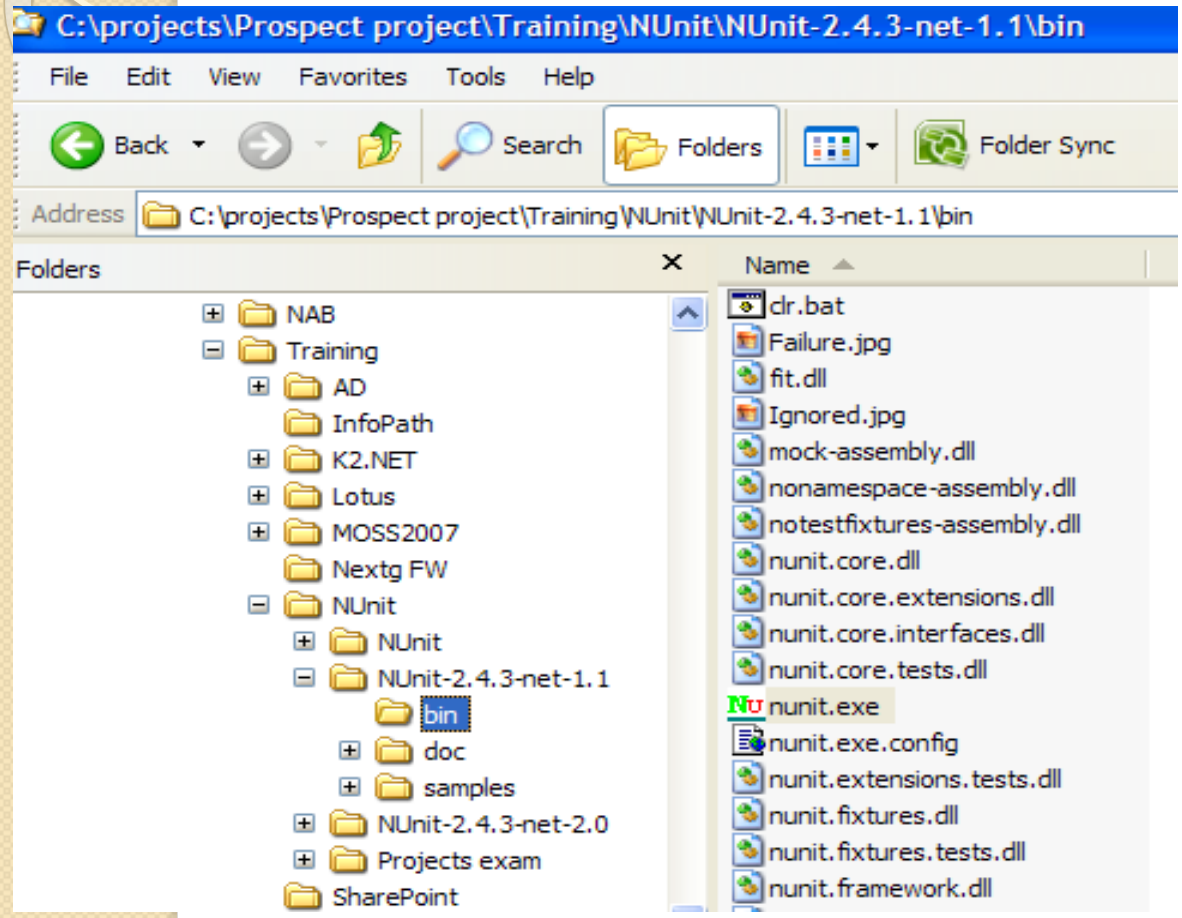
Automated Unit Testing with NUnit
Where to get NUnit?

- Let's go to website: http://www.nunit.org/index.php?p=download

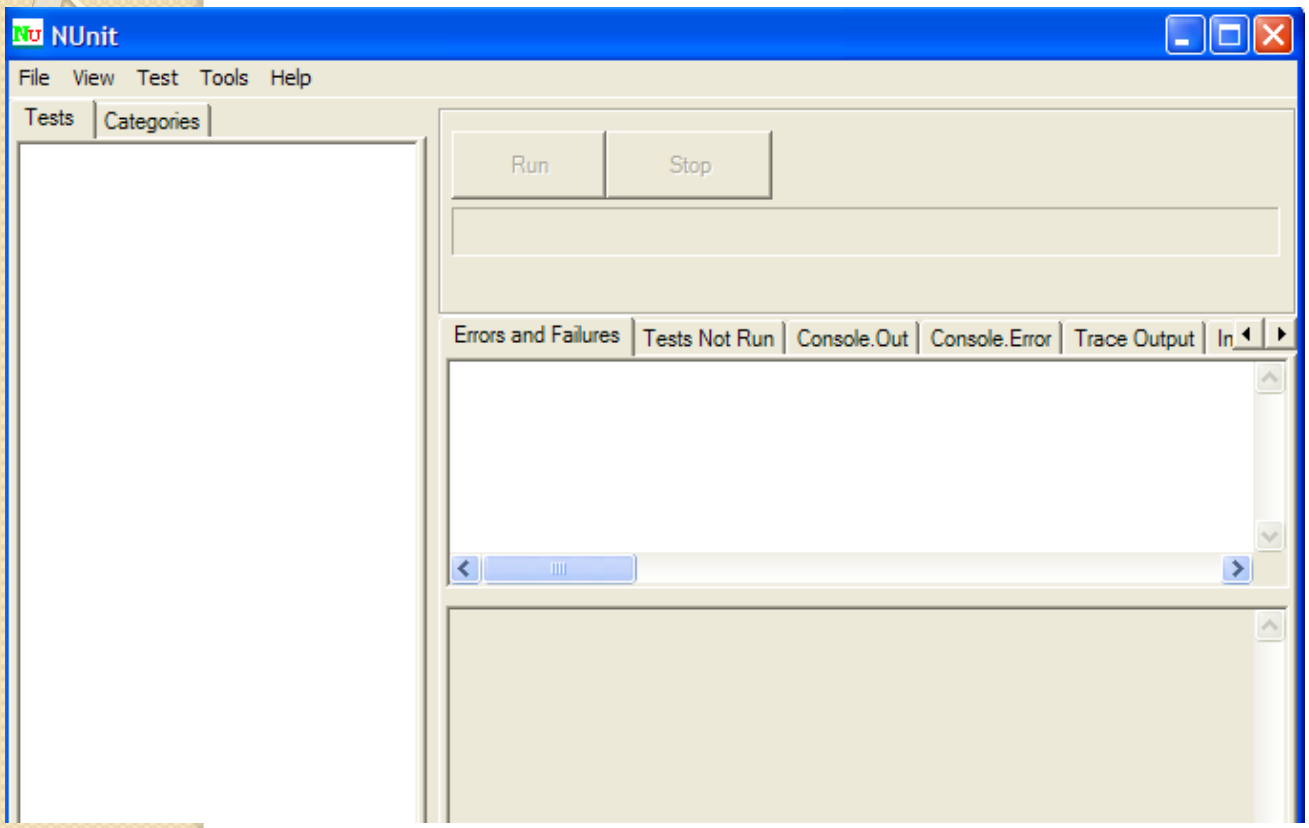
NUnit 2.4.3 (Recommended)	
win .net 1.1	NUnit-2.4.3-net-1.1.msi
win .net 2.0	NUnit-2.4.3-net-2.0.msi
bin .net 1.1	NUnit-2.4.3-net-1.1.zip
bin .net 2.0	NUnit-2.4.3-net-2.0.zip
src	NUnit-2.4.3-src.zip
doc	NUnit-2.4.3-doc.zip
NUnit 2.4.2	
win .net 1.1	NUnit-2.4.2-net-1.1.msi
win .net 2.0	NUnit-2.4.2-net-2.0.msi
bin .net 1.1	NUnit-2.4.2-net-1.1.zip
bin .net 2.0	NUnit-2.4.2-net-2.0.zip
src	NUnit-2.4.2-src.zip
doc	NUnit-2.4.2-doc.zip

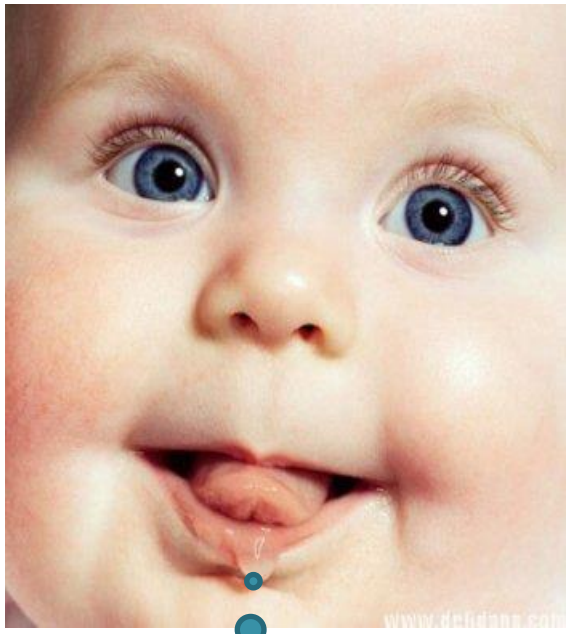
Yeahh, I got it

Automated Unit Testing with NUnit
Where to get Nunit? - Extract to any folder



Automated Unit Testing with Nunit
Screens of tool





Yeahh, It tastes good

Automated Unit Testing with Nunit
How to use NUnit?

- Create a test case base on NUnit framework
- Deploy and Run

Automated Unit Testing with Nunit
Create a test case

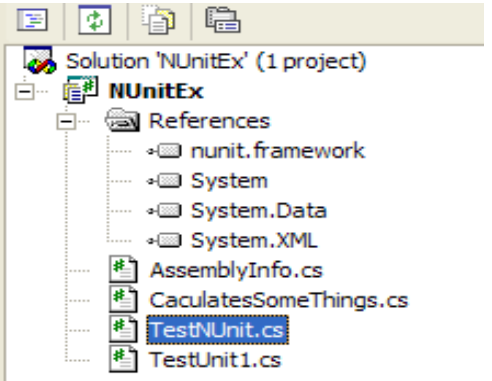
- Step 1: Create a Class
- Step 2: Add a reference nunit.Framework.dll to this class
- Step 3: Add a reference to *.dll contains function which you want to do Unit test
- Step 4: Restructure class following Nunit frame work
- Step 5: Write a test case

Automated Unit Testing with Nunit
Create a test case

- Step 1: Create a Class
- Step 2: Add a reference nunit.Framework.dll to this class
- Step 3: Add a reference to *.dll contains function which you want to do Unit test
- Step 4: Restructure class following Nunit frame work

NUnitEx.TestNUnit1 | testMultifly1()

```
namespace NUnitEx
{
    using System;
    using NUnit.Framework;
    /// <summary>
    /// Description about this class
    /// </summary>
    [TestFixture]
    public class TestNUnit1
    {
        private int intA;

        [SetUp]
        protected void SetUp()
        {
            intA = 0;
        }
        /// <summary>
        /// Write test case here
        [Test]
        public void testMultifly1()
        {
            Assert.AreEqual(0, intA);

        }
    }
}
```



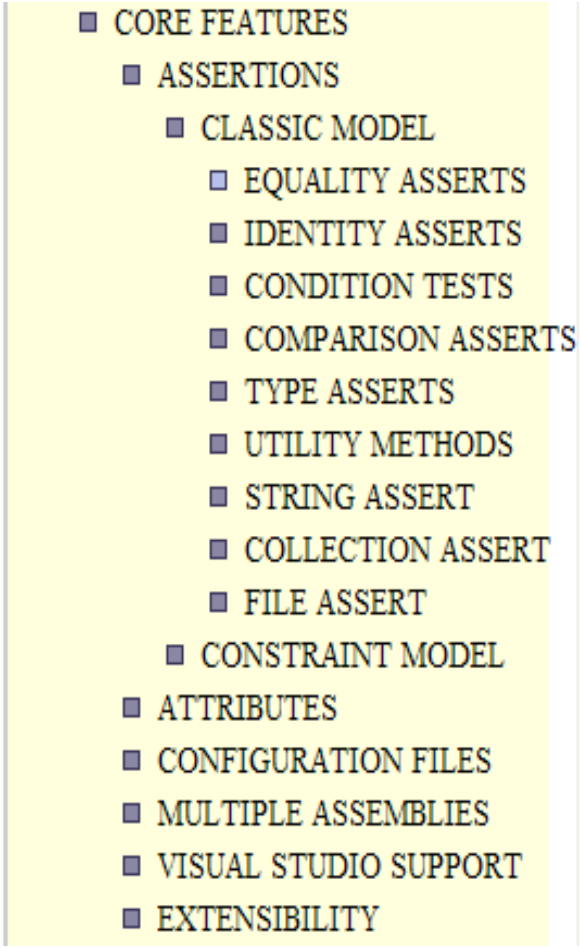
Automated Unit Testing with Nunit
Create a test case

- Step 5: Write a test case
  - Each test case will be a function/method of class
  - Must have attribute [Test] above a function/method
  - Ex:

```
    [Test]
public void testCase1()
{
    Assert.AreEqual(0, intA);
}

    [Test]
public void testCase2()
{
    Assert.AreEqual(0, divides(intA, intB));
}
```

Automated Unit Testing with Nunit Core Features

- Core Features to code a test case
  - **Assertions**
    - Equality Assserts:
      - Ex: Assert.AreEqual(int expected, int actual);
    - Condition Tests:
      - Ex: Assert.IsTrue(bool condition);
    - Comparrison Asserts
      - Ex: Assert.Greater(int arg1, int arg2);
    - Type Asserts
      - Ex: Assert.IsInstanceOfType(Type expected, object actual);
    - Utility methods
      - Ex: Assert.Fail();
    - String Assert
      - Ex: StringAssert.Contains(string expected, string actual);
    - Collection Asserts
      - Ex: CollectionAssert.AreEqual(Collection expected, Collection actual);
  - **Attributes**



Automated Unit Testing with Nunit Core Features

- **Attributes**

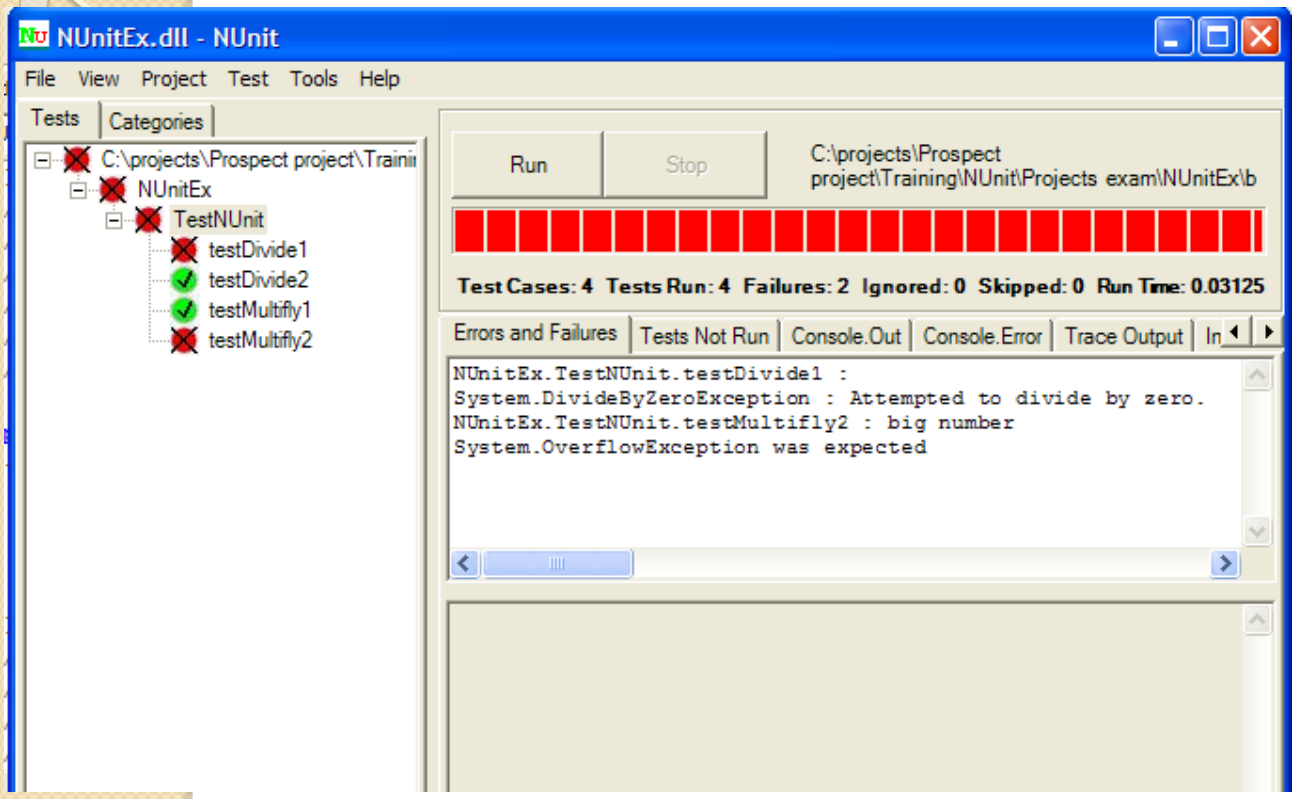
```
[TestFixture]
[Category("TestUnitExample")]
    public class TestNUnit
    {
        private int intA;
        private int intB;
        private CaculatesSomeThings objCal;

        [SetUp]
        protected void SetUp()
        {
                intA = 0;
                intB = 0;
                objCal = new CaculatesSomeThings();
        }

        [Test]
        Public void TestCase1()
        {
                Assert.AreEqual(0, objCal.Multifly(intA, intB))
        }
```

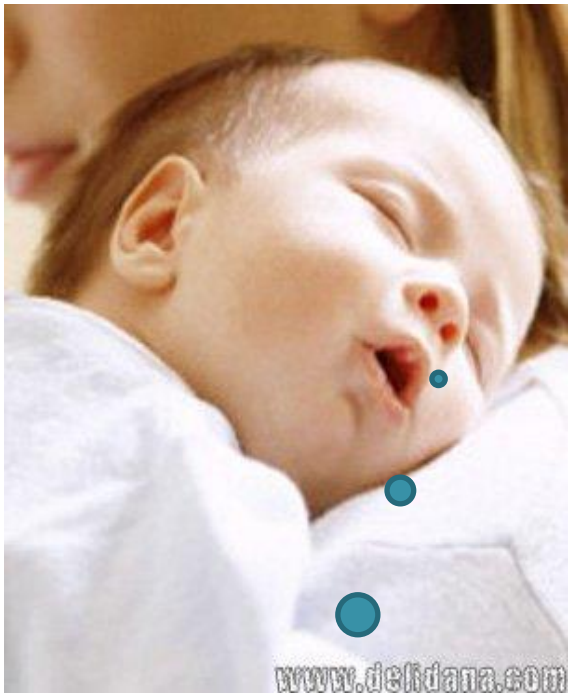
- CORE FEATURES
  - ASSERTIONS
  - ATTRIBUTES
    - CATEGORY
    - CULTURE
    - DESCRIPTION
    - EXPECTED EXCEPTION
    - EXPLICIT
    - IGNORE
    - PLATFORM
    - PROPERTY
    - SETCULTURE
    - SETUP
    - SETUP FIXTURE
    - SUITE
    - TEARDOWN
    - TEST
    - TEST FIXTURE
    - TEST FIXTURE SETUP
    - TEST FIXTURE TEARDOWN
  - CONFIGURATION FILES
  - MULTIPLE ASSEMBLIES
  - VISUAL STUDIO SUPPORT
  - EXTENSIBILITY

Automated Unit Testing with Nunit Deploy and Run

- Step 1: Compile a test case class to dll
- Step 2: Run NUnit tool
- Step 3: Open *.dll contains test case class
- Step 4: Choose the test case you want to run
- Step 5: Click run button to see the report

Automated Unit Testing with Nunit Deploy and Run





Quality… God let me sleep