Automated code testing involves exercising code and testing its behaviour by writing more code that tests your original code.

ACT is structured.

AAA structure: Arrange, Act, Assert

Arrange the information the test needs.

Act to execute the code under test.

Assert the results

ACT are self-documented. Each test should be self-explanatory.

The automation comes from a testing framework. MSTest is the Microsoft Testing Framework.

Visual studio also supports other testing frameworks such as NUnit.

Repeatable – each time something is run it produces the same result.

ACT are:

* Structured
* Self-documented
* Automatic
* Repeatable

There are several types of ACT, unit testing is one of them.

Unit tests – tests written and executed by the developer as part of an overall coding process. The goal of a unit test is to isolate each unit of code in an application and verify that it behaves as expected. We achieve that goal by refactoring to clean and testable methods.

If the Assert method fails, the unit test fails.

Unit tests will save you time by only executing the part the code you want to test, narrowing down bugs, allows for safe refactoring and modifying of code. Protects your code over the lifetime of the application. Minimises interruptions as errors introduced by others can be quickly discovered.

Code First – you write the code first based on the requirements of the project or you have existing code that you write unit tests for.

Test First – you write the tests first based on the requirements then you write the code to pass the tests.

Tests can only discover the presence of errors, it can’t prove the absence of errors so defining the appropriate unit tests is important.

Tests:

* A test for each set of valid input and verify the correct output is generated. If you can’t test them all, use a representative sample.
* Tests for each set of invalid inputs and verify the code appropriately handles it. Use a representative sample set.
* Test each guard clause
* Test every assumption
* Write a test for each bug fix

Visual studio has 2 ways to create unit tests:

1. Manually
2. Visual Studio Extension to generate them automatically.

Attributes – the square brackets above classes and methods [attribute].

[TestClass] – attribute that defines a test class

[TestMethod] – attribute that defines a test method

The automated code testing features use these attributes to locate the tests. All tests will need these attributes for the test framework to find them. The above attributes work for MSTest. Other test frameworks will use different attributes.

Each unit test case has it’s own method

Unit test method names usually state method being tested followed by intent.

CollectionAssert.AreEqual – use this method to test if 2 collections are equal.

Assert class is part of the Microsoft.VisualStudio.TestTools.UnitTesting; namespace.

Test 1 test case in each method.

Test explorer – where the unit tests are displayed

Math.Round() – round a number to specified decimal places.

CodeLens – associates test with method above method signature.

Code coverage – a feature of Unit testing that allows you to see which code is exercised.

There’s a Microsoft extension available to automate the generation of attributes and other elements of a test. Tools->extensions and updates. Go online and Search for ‘test generation’. Unit test generator – ALREADY HAVE IT

Testm tab tab – generates test method with attribute. Or not!!!!

Write a test to check if an exception is correctly generated:

* UTF has an attribute for that
* [ExpectedException(typeOf(ArgumentExeption))]
* This tells the unit test framework we expecting an exception of type ArgumentException
* Assert is not needed. The framework will generate an assert if the exception is not raised.
* If you want to verify the exception data you can catch the exception in a try..catch.. within the test method with an assert and then rethrowing the exception so the framework gets the expected exception.

Dealing with dependencies – stubs, fakes and mocks. There’s a course on it.