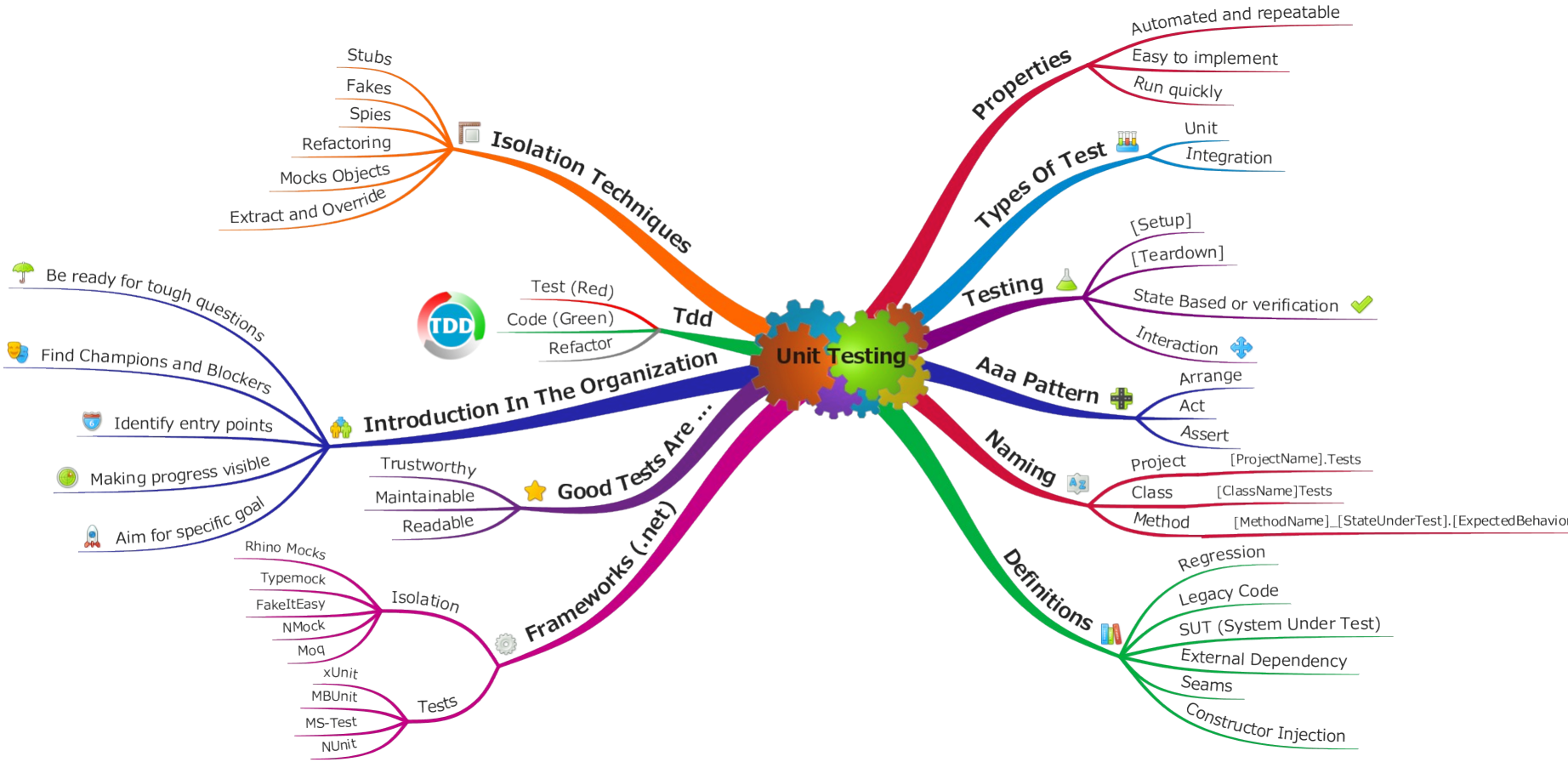
Testy jednostkowe w Visual Studio 2012

Wojciech Szymecki


Unit Testing
Properties
Automated and repeatable
Easy to implement
Run quickly
Types Of Test
Unit
Integration
Testing
[Setup]
[Teardown]
State Based or verification
Interaction
Aaa Pattern
Arrange
Act
Assert
Naming
Project [ProjectName].Tests
Class [ClassName]Tests
Method [MethodName]_[StateUnderTest].[ExpectedBehavior]
Definitions
Regression
Legacy Code
SUT (System Under Test)
External Dependency
Seams
Constructor Injection
Isolation Techniques
Stubs
Fakes
Spies
Refactoring
Mocks Objects
Extract and Override
Tdd
Test (Red)
Code (Green)
Refactor
Introduction In The Organization
Be ready for tough questions
Find Champions and Blockers
Identify entry points
Making progress visible
Aim for specific goal
Good Tests Are ...
Trustworthy
Maintainable
Readable
Frameworks (.net)
Isolation
Rhino Mocks
Typemock
FakeItEasy
NMock
Moq
Tests
xUnit
MBUnit
MS-Test
NUnit

Czym są testy jednostkowe?

- Krótkie kawałki kodu sprawdzające wynik działania funkcji pod kątem różnych danych wejściowych.
- Każdy test jest ujęty w osobną metodę.
- Metody znajdują się w klasie.
- Testy integracyjne to kod który testuje współdziałanie klas/modułów sprawdzonych najpierw przez testy jednostkowe.

Właściwości testów jednostkowych

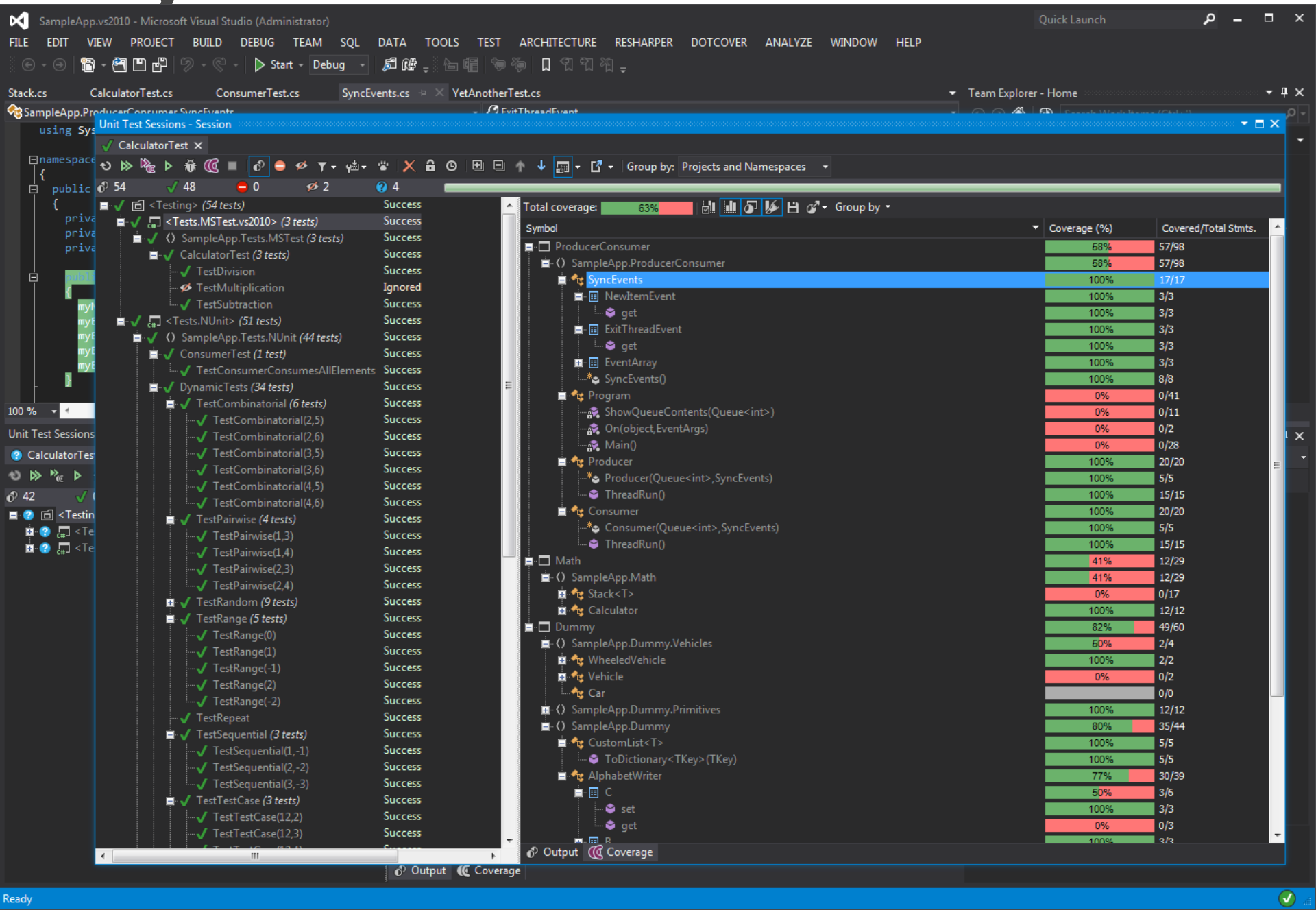
- Są zautomatyzowane i powtarzalne.
- Łatwe w implementacji.
- Powinny się szybko wykonywać.

Wzorzec AAA

- Arrange
- Act
- Assert

```
 1:  [TestFixture]
 2:  public class CalcTest_NUnit
 3:  {
 4:      private LocalEuroCalculator _calc;
 5:
 6:      [SetUp]
 7:      public void CreateCalculator()
 8:      {
 9:          _calc = new LocalEuroCalculator();
10:      }
11:
12:      [Test]
13:      public void Calculates_Pln()
14:      {
15:          decimal input = 50.0m;
16:          decimal euroRate = 4.35m;
17:          decimal correctResult = 217.5m;
18:
19:          decimal result = _calc.Calculate(input,
euroRate);
20:
21:          Assert.AreEqual(correctResult, result);
22:      }
23:  }
```

Wyniki

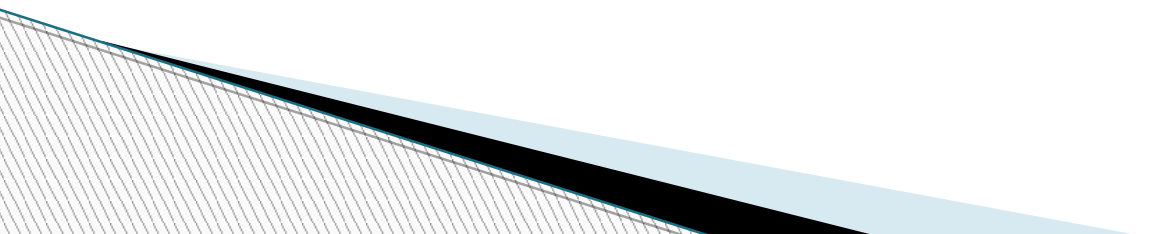

SampleApp.vs2010 - Microsoft Visual Studio (Administrator)
Unit Test Sessions - Session
CalculatorTest
Group by: Projects and Namespaces
54
48
0
2
4
Total coverage: 63%
Group by
Symbol
Coverage (%)
Covered/Total Stmts.
ProducerConsumer 58% 57/98
SampleApp.ProducerConsumer 58% 57/98
SyncEvents 100% 17/17
NewItemEvent 100% 3/3
get 100% 3/3
ExitThreadEvent 100% 3/3
get 100% 3/3
EventArray 100% 3/3
SyncEvents() 100% 8/8
Program 0% 0/41
ShowQueueContents(Queue<int>) 0% 0/11
On(object,EventArgs) 0% 0/2
Main() 0% 0/28
Producer 100% 20/20
Producer(Queue<int>,SyncEvents) 100% 5/5
ThreadRun() 100% 15/15
Consumer 100% 20/20
Consumer(Queue<int>,SyncEvents) 100% 5/5
ThreadRun() 100% 15/15
Math 41% 12/29
SampleApp.Math 41% 12/29
Stack<T> 0% 0/17
Calculator 100% 12/12
Dummy 82% 49/60
SampleApp.Dummy.Vehicles 50% 2/4
WheeledVehicle 100% 2/2
Vehicle 0% 0/2
Car 0/0
SampleApp.Dummy.Primitives 100% 12/12
SampleApp.Dummy 80% 35/44
CustomList<T> 100% 5/5
ToDictionary<TKey>(TKey) 100% 5/5
AlphabetWriter 77% 30/39
C 50% 3/6
set 100% 3/3
get 0% 0/3
Output
Coverage
Ready

Konwencje nazewnictwa

- [NazwaProjektu].Tests
- [NazwaKlasy]Tests
- [NazwaMetody]_[TestowanyStan].
  [SpodziewaneZachowanie]

Test driven development (TDD)

- Test
- Code
- Refactor



Mock

```
[Test]
public void LogOn_Authenticates()
{
    string userName = Randomizer.String();
    string password = Randomizer.String();
    var authenticationService = MockRepository.GenerateMock<IAuthenticationService>();

    AccountController controller = new AccountController(authenticationService);
    controller.LogOn(userName, password);

    authenticationService.AssertWasCalled(x => x.Authenticate(userName, password));
}
```

Stub

```
[Test]
public void OnAuthenticationFailure_InvalidatesModel()
{
    string userName = Randomizer.String();
    string password = Randomizer.String();
    var authenticationService = MockRepository.GenerateStub<IAuthenticationService>();
    authenticationService.Stub(x => x.Authenticate(userName, password)).Return(false);

    AccountController controller = new AccountController(authenticationService);
    controller.LogOn(userName, password);

    Assert.IsFalse(controller.ModelState.IsValid);
}

[Test]
public void OnAuthenticationSuccess_ModelValid()
{
    string userName = Randomizer.String();
    string password = Randomizer.String();
    var authenticationService = MockRepository.GenerateStub<IAuthenticationService>();
    authenticationService.Stub(x => x.Authenticate(userName, password)).Return(true);

    AccountController controller = new AccountController(authenticationService);
    controller.LogOn(userName, password);

    Assert.IsTrue(controller.ModelState.IsValid);
}
```

Frameworki do testów

- MS-Test
- nUnit
- xUnit

Frameworki do izolacji

- Moq
- FakeItEasy
- RhinoMocks

Przydatne linki / bibliografia

- http://www.maciejaniserowicz.com/2011/08/08/ut-0-zapowiedz-minicyklu-o-testach/
- http://msdn.microsoft.com/pl-pl/library/testy-jednostkowe-w-visual-studio

DEMO

Dziękuję za uwagę