NET.C#. 03
Модульное тестирование

О чем пойдет речь?

- Что такое unit-тестирование?
- Плюсы unit-тестирования
- Unit-тестирование и стоимость разработки
- Устройство unit-теста
- Ограничения unit-тестирования

Что такое unit–тестирование?

- Unit-тестирование – проверка корректности небольших независимых кусочков кода.
- Цель unit-тестирование – показать, что каждый модуль приложения работает корректно.

Практика unit-тестирования

- Тестируемый кусочек кода = класс или метод класса
- Unit-тесты автоматизированы
- Unit-тесты пишутся на том же языке, что и тестируемый код
- Unit-тесты – простые!

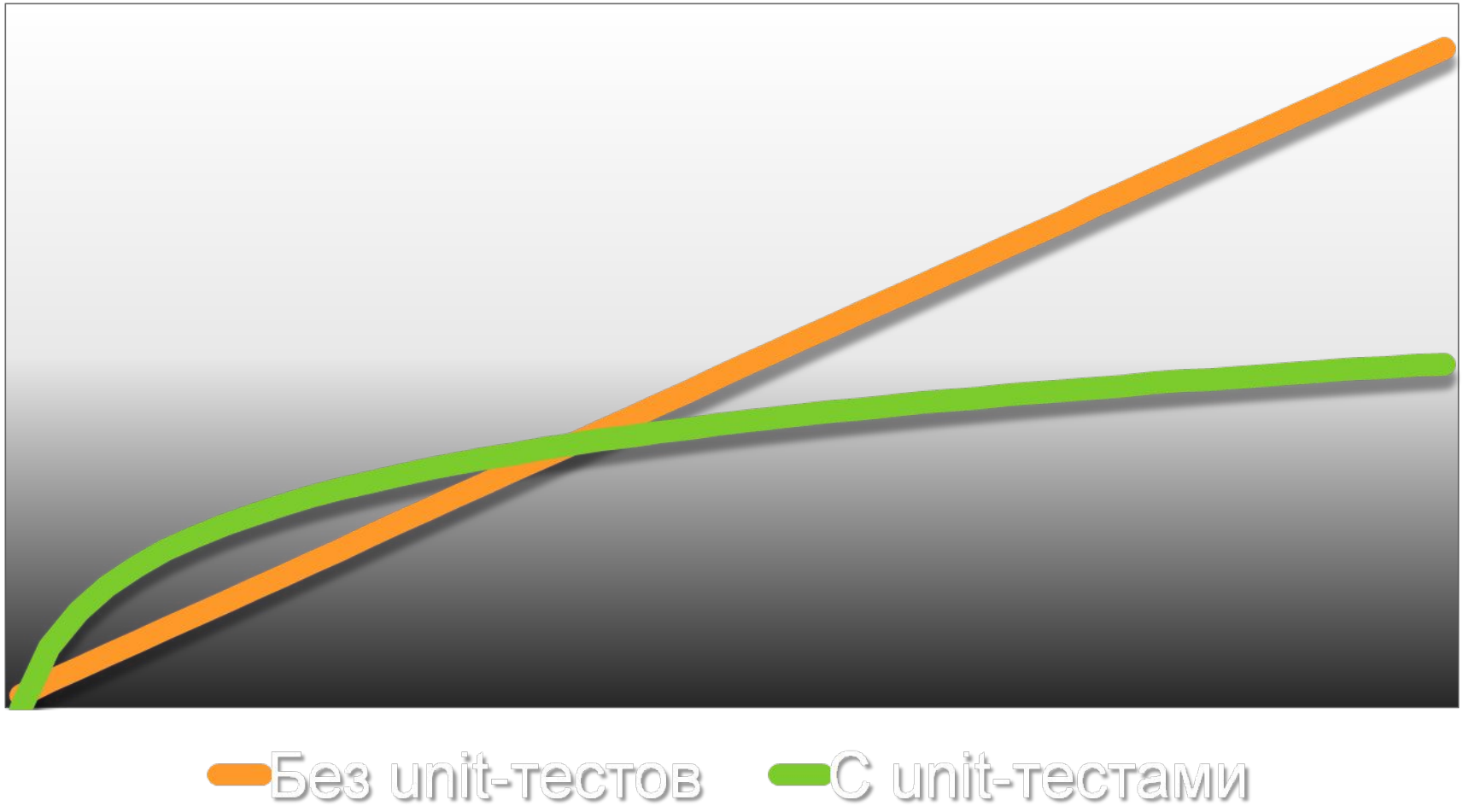
Плюсы unit-тестирования

- Упрощают регрессионное тестирование, стимулируют рефакторинг
- Поощряют написание слабосвязанного кода
- Unit-тесты = документация!

Практика unit-тестирования

- Тестируемый кусочек кода = класс или метод класса
- Unit-тесты автоматизированы
- Unit-тесты пишутся на том же языке, что и тестируемый код
- Unit-тесты – простые!

Unit-тесты и стоимость изменений



Устройство unit-теста

- Unit-тест = pattern-based тест
- Unit-тест не зависит от других тестов!

Тестирование метода

Шаблон для написания тестов – «Triple A» (Arrange-Act-Assert)

Согласно этому шаблону тест состоит из трех частей

1. Arrange (Устанавить) – осуществить настройку входных данных для теста;
2. Act (Выполнить) – выполнить действие, результаты которого тестируются;
3. Assert (Проверить) – проверить результаты выполнения

Тестирование метода

```
[Test Fixture]
public class ProgramTest
{
    [Test]
    public void Add2Numbers_CorrectResult()
    {
        var target = new ArithmeticUnit();
        target.OperandA = 2;
        target.OperandB = 3;

        target.Add();

        Assert.That(target.Result, Is.EqualTo(5));
}
```

Arrange

Act

Assert

Тестирование метода

```
public int Calculate(int operandOne, int operandTwo)
{
    int result = 0;
    // Perform some calculation.
    return result;
}
```

Create Unit Tests Wizard

```
/// <summary>
///A test for Calculate
///</summary>
[TestMethod()]
public void CalculateTest()
{
    Class1 target = new Class1(); // TODO: Initialize to an appropriate value
    int operandOne = 0; // TODO: Initialize to an appropriate value
    int operandTwo = 0; // TODO: Initialize to an appropriate value
    int expected = 0; // TODO: Initialize to an appropriate value
    int actual;
    actual = target.Calculate(operandOne, operandTwo);
    Assert.AreEqual(expected, actual);
    Assert.Inconclusive("Verify the correctness of this test method.");
}
```

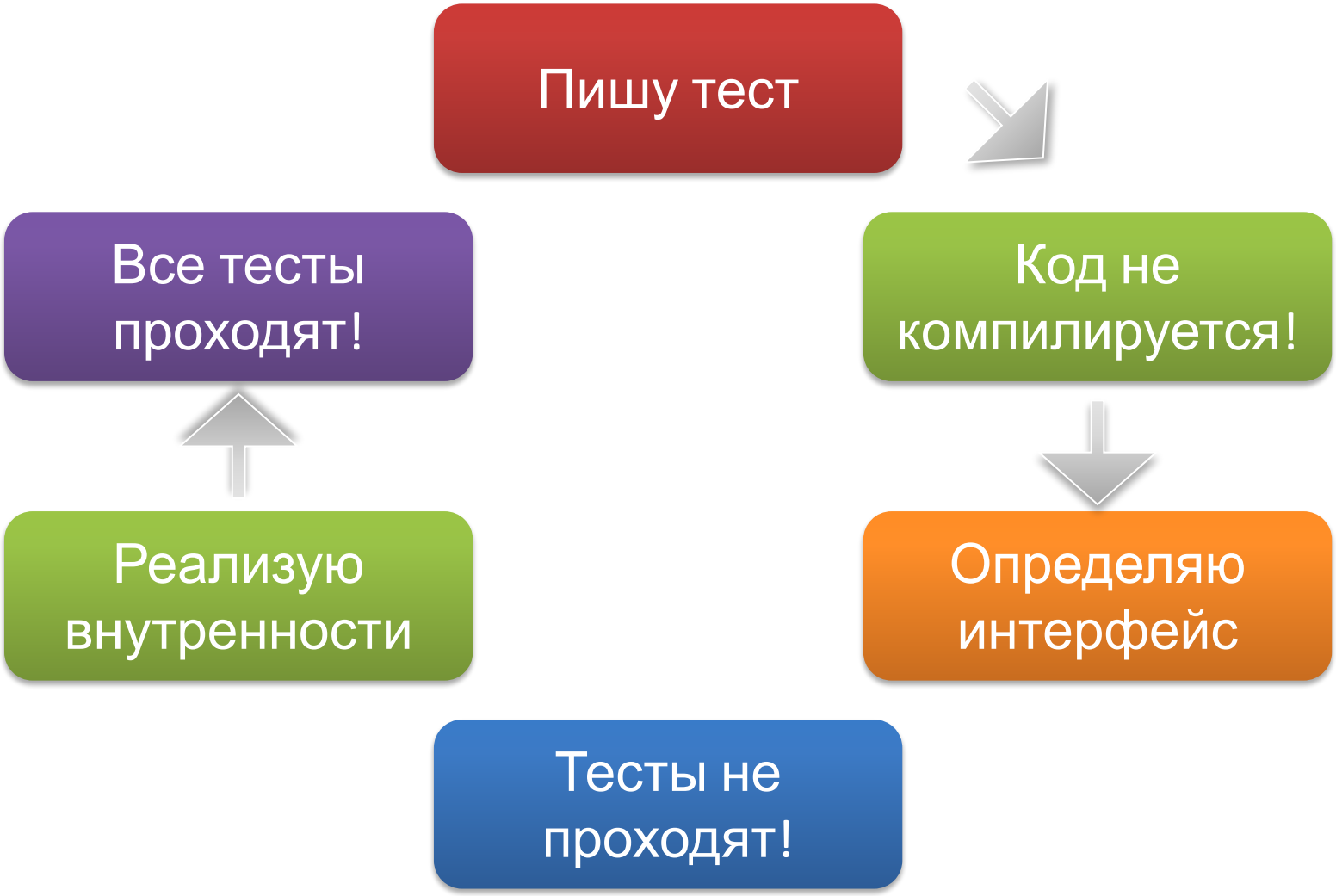
Тестирование метода

Разработка через тестирование (test-driven development, TDD)

Разработка через тестирование происходит в несколько этапов

- Добавление теста
- Запуск всех тестов: убедиться, что новые тесты не проходят
- Написание кода
- Запуск всех тестов: убедиться, что все тесты проходят
- Рефакторинг
- Повторение цикла

TDD: тесты пишутся до кода!



Unit-Testing Frameworks

Инструментарий для unit-тестирования

- Разметка тестов
- Проверка условий
- Выполнение тестов
- Создание отчетов

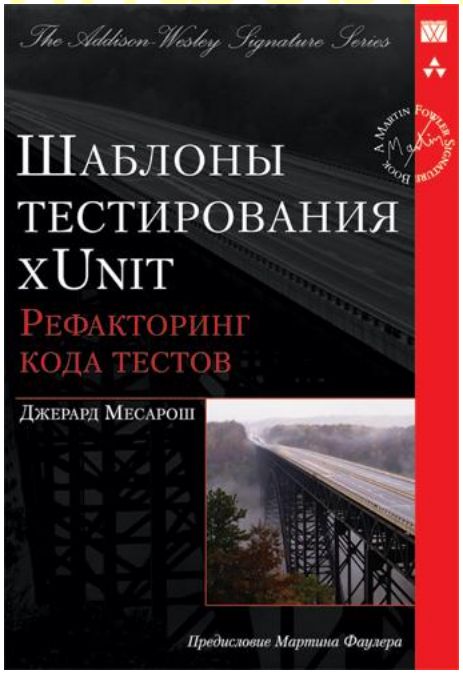
2011 © EPAM Systems, RD Dep.

Unit–Testing Frameworks / .NET

- Средства в составе Visual Studio
  - http://msdn.microsoft.com/en-us/library/dd264975.aspx
- NUnit
  - http://www.nunit.org/
- xUnit.net
  - http://www.codeplex.com/xunit
- MbUnit / Gallio Automation Platform
  - http://www.gallio.org/

2011 © EPAM Systems, RD Dep.

Дополнительная информация



Джерард Месарош

Шаблоны тестирования xUnit. Рефакторинг кода тестов.

Издательство: Вильямс, 2009 г.

http://www.williamspublishing.com/Books/978-5-8459-1448-4.html

2011 © EPAM Systems, RD Dep.

Спасибо за внимание