Методы разработки качественного и "чистого" кода

Новиков Иван

Новиков Иван

http://jonnynovikov.com/

@jonny

Моя позиция может не совпадать с позицией компаний, в которых я работаю

Кто я?

Novikov Ivan

CTO at Amiforus & Backend engineer at ozon.travel

http://jonnynovikov.com/

Agenda

TDD

РЕФАКТОРИНГ

АРХИТЕКТУРА

ПРОЦЕССЫ

ПРИМЕРЫ НА C#

ДИАГНОЗ И ЗАКЛЮЧЕНИЕ

Q&A

TDD?

TDD...

будем дальше считать академическим видом программирования

SUT, SUD, ... почему все-таки TDD - огромный вклад в развитие инженерии в целом?

РЕФАКТОРИНГ

не должен изменять семантику

должен улучшать качество

РЕФАКТОРИНГ

не панацея

стоимость, необходимость

АРХИТЕКТУРА

Несовершенная архитектура

непредвиденные изменения

существенные изменения

REFACTOR!

непредвиденные изменения

REBUILD!

существенные изменения

РЕФАКТОРИНГ ТРЕБУЕТ АНАЛИЗА

Рефактори СВОЁ!

хлам от рефакторинга останется хламом (GIGO)

никакое его количество не спасет дефектную архитектуру

Rebuild

Удачи!

ПРОЦЕССЫ

РАЗРАБОТКА, управляемая тестами

полноценный метод **разработки** ПО

Test First Development

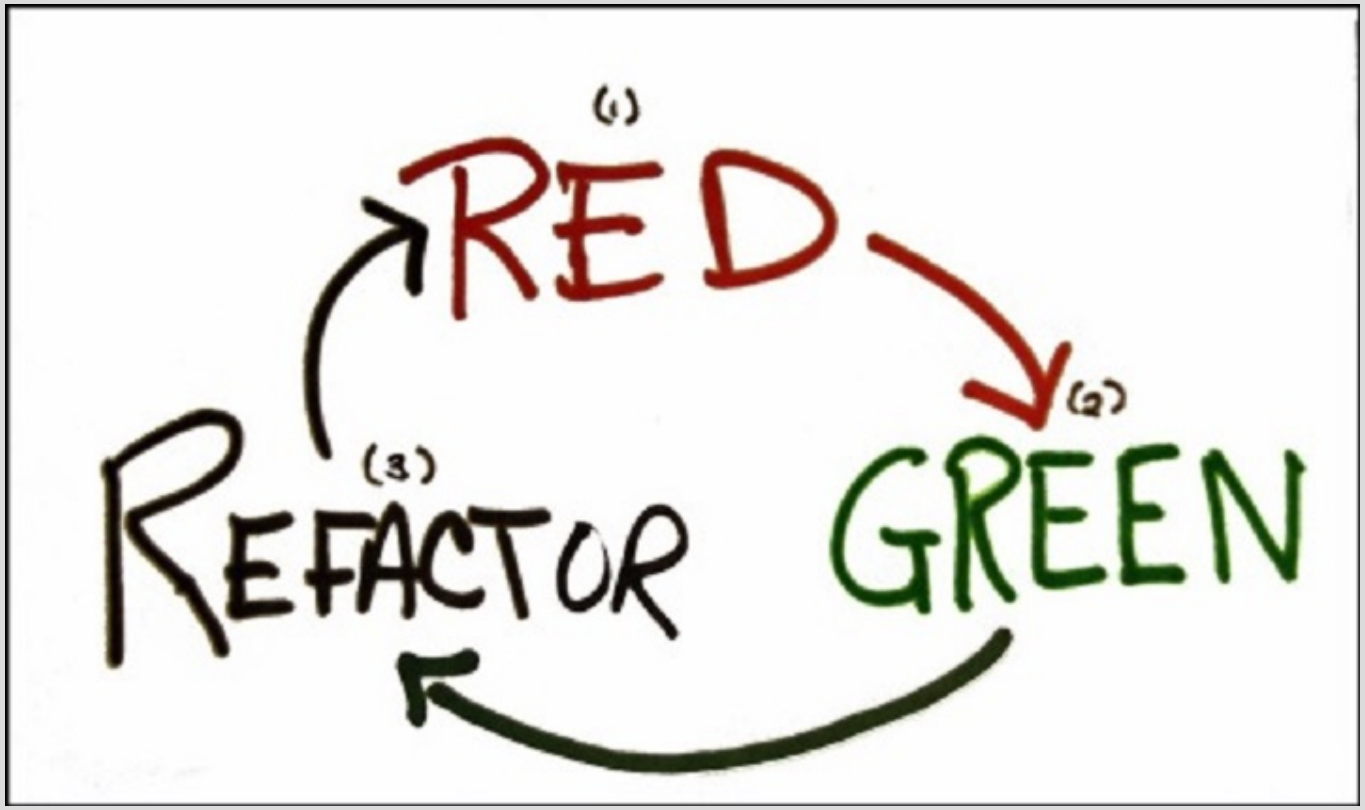
create test

modify: code the simple thing

refactor

"Make it work. Make it right. Make it fast."

Kent Beck



Test Driven Development

create test

check test is failed

modify: code the simple thing

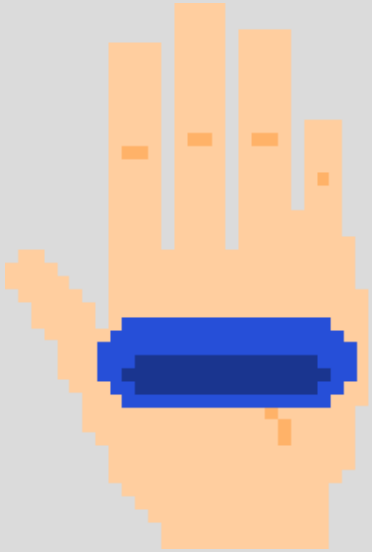
ensure test is passed

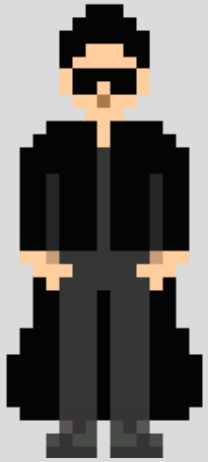
continuous refactor (unify)

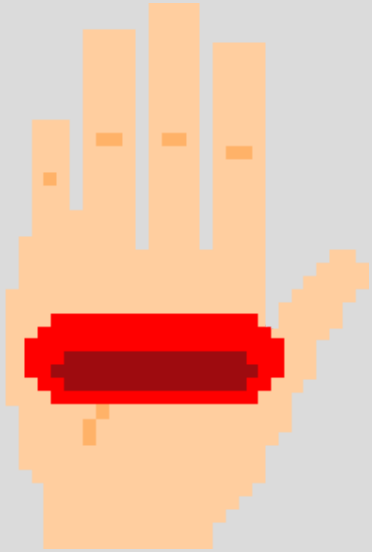
"Лондонская школа TDD"

моки как средство унификации проектирования ПО

Реальность







Code example

```
 7 module("Module 1");
 6
 5 test("test example1a", function() {
 4   equal(5, 5, "The values are equal");
 3 });
 2
 1 test("test example2a", function() {
8    equal(5, 5, "The values are equal");
 1 });
 2
 3 module("Module 2");
 4
 5 test("test example2a", function() {
 6   equal(5, 5, "The values are equal");
 7 });
~
~
~
~
~
~
~
~
~
 NORMAL  tests/example_tests.js
:R
```

О спецификациях (MSpec)

Основные принципы

Простота (KISS)

"Чистый" код

Не делай то, что сейчас не используешь (YAGNI)

Attention

Будьте скептиком

Разработку, управляемую тестами стоит попробовать

Моки необходимо использовать

ТОЛЬКО там, где они действительно нужны

и еще...

Где меня обманули

тесты - это не все, что нужно для спецификации ПО

тест более специфичен, чем user story

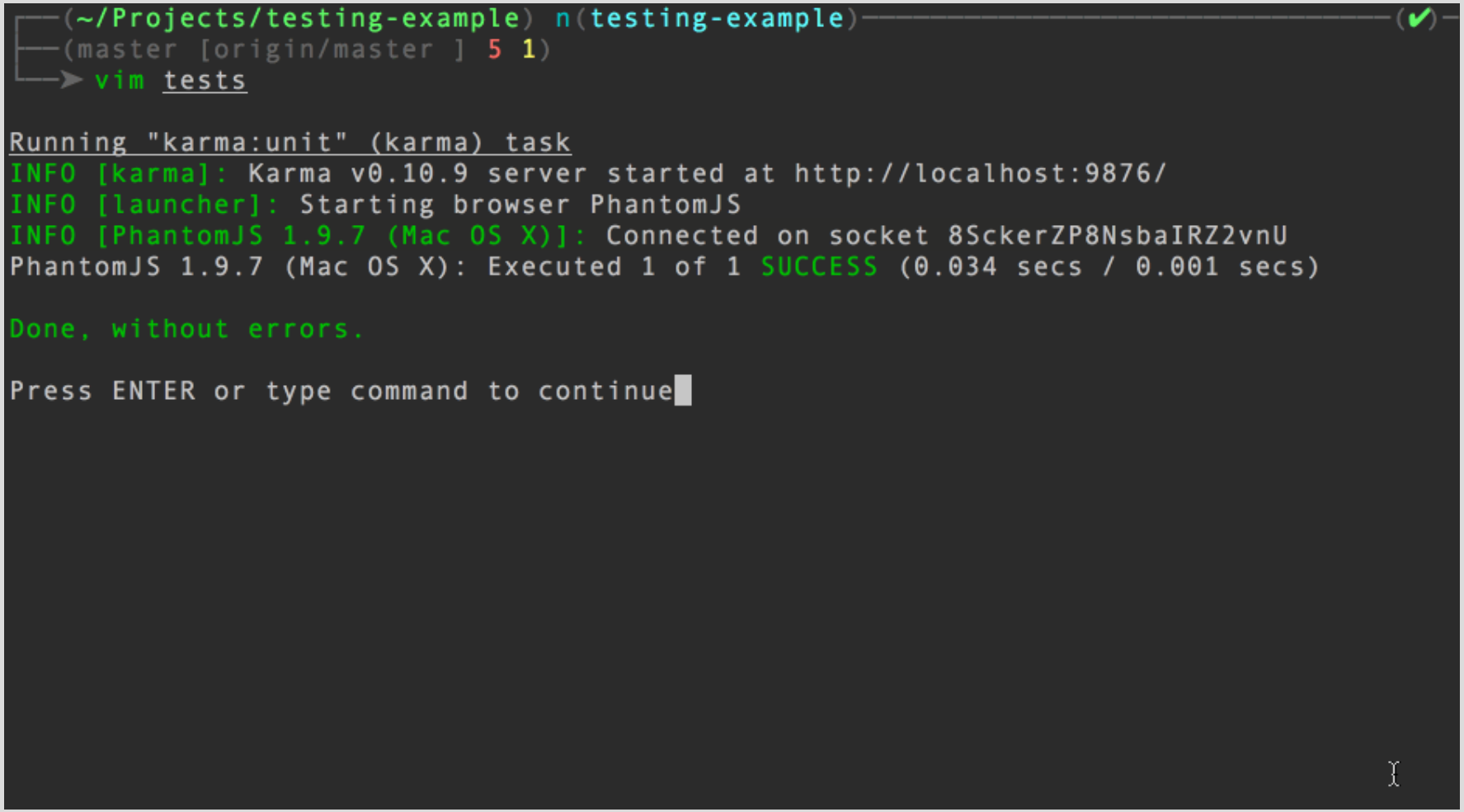
поиски общих правил, нет рецепта генерации спецификации по тестам

Почему все-таки классная штука?

каждый новый код должен сопровождаться тестами

экстремальный вариант хороший способ заставить себя сделать это

Немного кода



Почему многие ВООБЩЕ не пишут тесты?

- "Зачем проверять только состояния?"
- "Это отнимет всё время"
- "Их невозможно поддерживать"

Почему ПИШУТ и даже разрабатывают через тесты?

У них есть рецепт, который другие не знают?

Проблемы заглушек

Isolation Framework

Используя NSubstitute
https://github.com/nsubstitute/NSubstitute
код с предыдущего слайда превращается в

```
[TestFixture]
internal class LogAnalyzerTests
{
    [Test]
    public void Analyze_TooShortFileName_CallLogger()
    {
        var logger = Substitute.For<ILogger>();
        new LogAnalyzer(logger) { MinNameLength = 6 }.Analyze("a.txt");
        logger.Received().LogError("Filename too short: a.txt");
    }
```

Mocking Framework

Moq https://github.com/Moq/moq4

```
public static IReturnsResult<TContext> ReturnsDbSet<TEntity, TContext>(
        this IReturns<TContext, DbSet<TEntity>> setup,
        ICollection<TEntity> entities)
    where TEntity : class
    where TContext : DbContext
{
    return setup.Returns(MockDbSet(entities));
}

    public static DbSet<T> MockDbSet<T>(ICollection<T> table) where T : class
    {
    var dbSet = new Mock<DbSet<T>>();
    dbSet.As<IQueryable<T>>().Setup(q => q.Provider).Returns(() => table.AsQueryable().Provider);
    dbSet.As<IQueryable<T>>().Setup(q => q.Expression).Returns(() => table.AsQueryable().Expression);
    dbSet.As<IQueryable<T>>().Setup(q => q.ElementType).Returns(() => table.AsQueryable().ElementType);
    dbSet.As<IQueryable<T>>().Setup(q => q.GetEnumerator()).Returns(() => table.AsQueryable().GetEnumerator());
    dbSet.Setup(set => set.Add(It.IsAny<T>())).Callback<T>(table.Add);
    dbSet.Setup(set => set.AddRange(It.IsAny<IEnumerable<T>>())).Callback<IEnumerable<T>>(ts => table.AddRange(ts));
    dbSet.Setup(set => set.Remove(It.IsAny<T>())).Callback<T>(t => table.Remove(t));
    dbSet.Setup(set => set.RemoveRange(It.IsAny<IEnumerable<T>>())).Callback<IEnumerable<T>>(ts => table.RemoveRange(ts));
    return dbSet.Object;
    }

public static DbSet<T> InMemoryDbSet<T>(HashSet<T> table) where T : class
{
    var dbSet = InMemory.InMemoryDbSet<T>.CreateGlobal(table);
    return dbSet.Object;
}
```

С блэкджеком и... девочками

IoC + Nancy https://github.com/NancyFx/Nancy

О необходимости тестирования

Тесты - тоже искусство

Selenium WebDriver http://seleniumhq.org

```
[TestFixture(typeof (Firefox), typeof (FirefoxDriver))]
[TestFixture(typeof (InternetExplorer), typeof (InternetExplorerDriver))]
[TestFixture(typeof (Chrome), typeof (ChromeDriver))]
[TestFixture(typeof (PhantomJS), typeof (PhantomJSDriver))]
0 references | 0 changes | 0 authors, 0 changes
public class DriverEnvironmentTests<TDriverEnvironment, TExpectedDriver>
    where TDriverEnvironment : IDriverEnvironment, new()
    where TExpectedDriver : IWebDriver
{
    [Test]
    0 references | 0 changes | 0 authors, 0 changes
    public void given_driver_environment_when_creating_web_driver_should_be_correct_type()
    {
        var environment = new TDriverEnvironment();

        using (var driver = environment.CreateWebDriver())
        {
            driver.Should().BeOfType<TExpectedDriver>();
        }
    }
}
```

Вырастающая сложность

Поддержка

Проектирование и программирование -

виды человеческой деятельности;

стоит об этом забыть – и все пропало.

Бьярн Страуструп, 1991

Что начать читать завтра?

Ironies of Automation, 1983

Test-Driven development: By example (Kent Beck, 2003)

Исскуство автономного тестирования (Рой Ошервуд):

русская или english

&

Что посмотреть вместо сериала?

Norwegian Developers Conference (NDC)

Q&A

Презентация тут: http://j.mp/quality-evolve