



PROPERTY BASED BDD EXAMPLES

Presented by Gaspar Nagy

gaspar@specsolutions.eu

[@gasparnagy](mailto:gasparnagy), <http://gasparnagy.com>

© All rights reserved

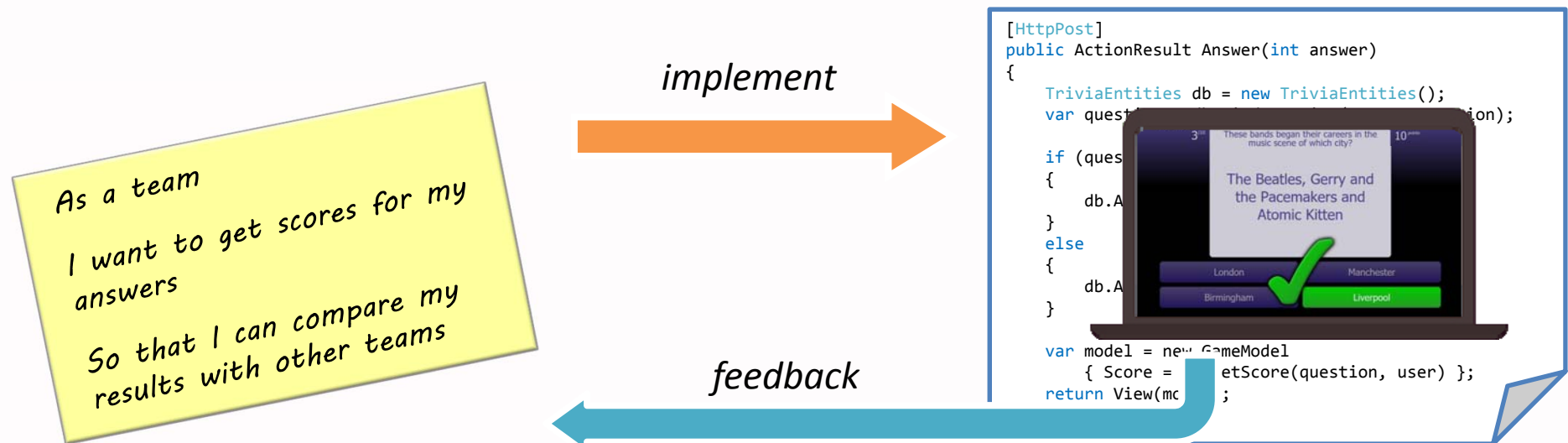
BDD

- Behavior Driven Development
- ~Specification by Example (SbE)
- ~Acceptance Test Driven Development (ATDD)
- ~Keyword Driven Testing

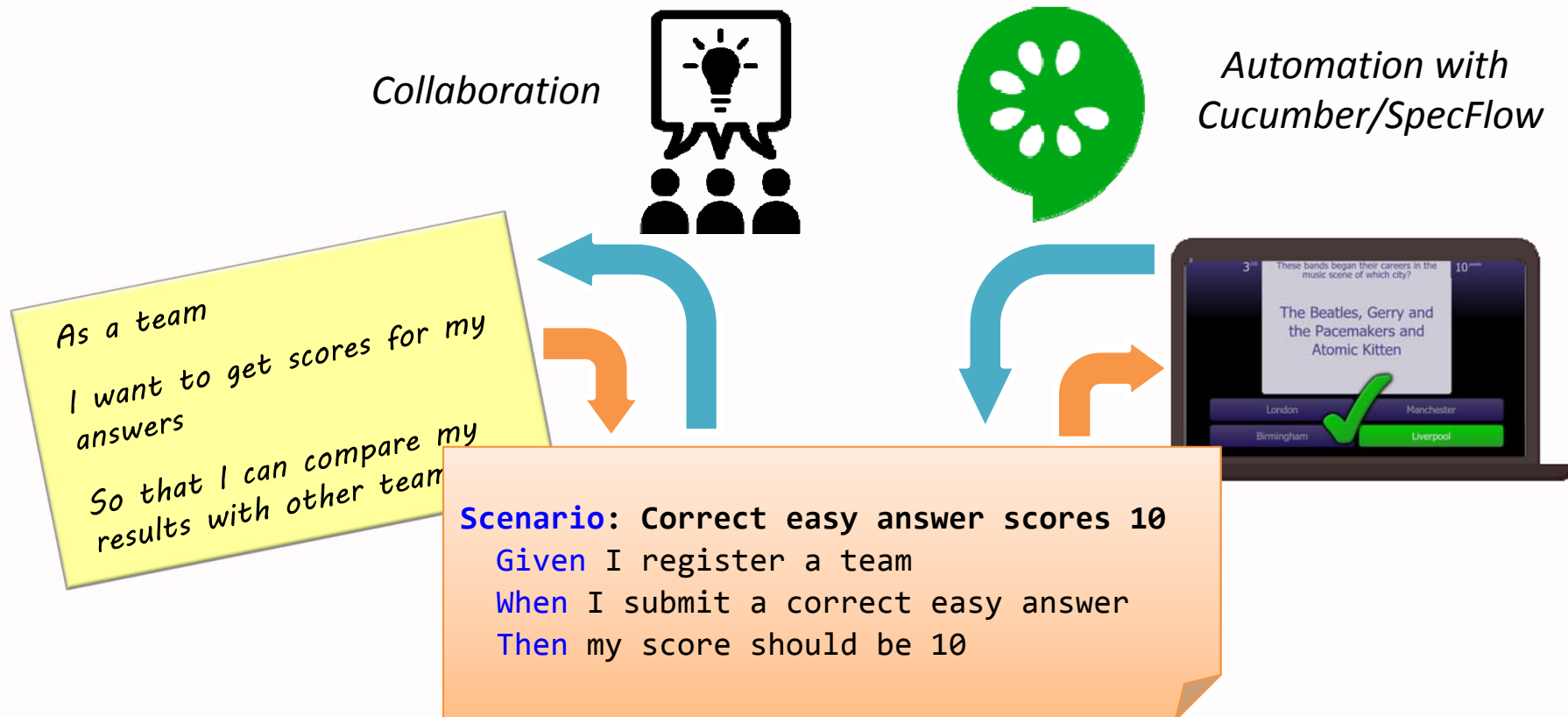
PBT

- Property Based Testing
- Property Testing
- ~Random Testing
- ~Model-based Testing

What is BDD? – Classic model



What is BDD?



This is an example!

Scenario: Correct easy answer scores 10
Given I register a team
When I submit a correct easy answer
Then my score should be 10

How many examples you need?

Scenario: Add two numbers

Given I have

And I

When I

Then I

- *BDD uses the examples to help understanding the requirements*
- *You need as many examples as many makes you understand the business problem*
- *“Full coverage” is not a direct goal*

the calculator
the calculator
on the screen

should be 2 on the screen

Scenario Outlines

Scenario Outline: Add two numbers

Given I have entered `<a>` into the calculator

And I have entered `` into the calculator

When I press add

Then the result should be `<result>` on the screen

Examples:

<code>a</code>	<code>b</code>	<code>result</code>
1	2	3
5	-7	-2
2	0	2

BDD is for...

understanding & validating
business requirements
through illustrative examples

PBT is for...

verifying
implementation
the “properties”
through checking statements about the
output for many different possible inputs

source: <http://blog.jessitron.com/2013/04/property-based-testing-what-is-it.html>

For example for addition...

- Commutative property: $a + b = b + a$
- Associative property: $(a + b) + c = a + (b + c)$
- Identity property: $a + 0 = a$
- Distributive property: $a * (b + c) = a*b + a*c$

We would like to verify these for ~ALL input combinations!

There is a tool for doing this!

- QuickCheck (Haskell) is the canonical framework, but there are many different ports of it to other programming languages
 - QuickCheck for Java
 - PhpQuickCheck for PHP
 - ScalaCheck for Scala
 - FsCheck for .NET (F#, C#)
 - ... (see more at <https://en.wikipedia.org/wiki/QuickCheck>)

FsCheck Sample

```
[TestMethod]
public void Addition_Identity()
{
    Func<int, bool> identity =
        (a) => Addition.Add(a, 0) == a;

    Prop.ForAll(identity).QuickCheckThrowOnFailure();
}

[TestMethod]
public void Addition_Commutativity()
{
    Func<int, int, bool> commutative =
        (a, b) => Addition.Add(a, b) == Addition.Add(b, a);

    Prop.ForAll(commutative).QuickCheckThrowOnFailure();
}
```

When the implementation is wrong...

```
public static int Add(int op1, int op2)
{
    // after 38, every year counts twice
    return op1 >= 38 ? op1 + op2 * 2 : op1 + op2;
}
```

Addition_Commutativity

Source: FsCheckDemo.cs line 30

✗ Test Failed - Addition_Commutativity

Message: Test method MyCalculator.Tests.FsCheckDemo.Addition_Commutativity threw exception:
System.Exception: Falsifiable, after 68 tests (9 shrinks) (StdGen (1826352274,296211082)):

Original:

(66, 53)

Shrunk:

(1, 38)

Elapsed time: 78 ms

BDD

- $1 + 2 = 3$
- $5 + -7 = -2$
- $2 + 0 = 2$

PBT

- Commutative
- Associative
- Identity
- Distributive

*Both are important for
understanding the
requirements*

Examples in SpecFlow

```
[Binding]
public class AdditionSteps
{
    private readonly Calculator calculator;

    [Given]
    public void Given_I_have_entered_NUMBER_into_the_calculator(int number)
    {
        calculator.Enter(number);
    }

    [When]
    public void When_I_press_add()
    {
        calculator.Add();
    }

    [Then]
    public void Then_the_result_should_be_EXPECTEDRESULT_on_the_screen(int expectedResult)
    {
        Assert.AreEqual(expectedResult, calculator.Result);
    }
}
```



```
Addition.feature  AddTwoNumbers.cs
1  Feature: Addition
2
3  Scenario: Add two numbers
4      Given I have entered 1 into the calculator
5      And I have entered 2 into the calculator
6      When I press add
7      Then the result should be 3 on the screen
8
```



AddTwoNumbers

Source: Addition.feature line 3



Test Passed - AddTwoNumbers

Elapsed time: 34 ms

Output

Identity property BDD style...

`@propertyBased`

`Scenario: Identity property`

`Given I have entered any number into the calculator`

`And I have entered 0 into the calculator`

`When I press add`

`Then the result should be the first number on the screen`


Defining constraints and expectations...

```

[Binding]
public class Constraints : ConstraintsBase
{
    [StepArgumentTransformation("any number")]
    public int AnyNumber()
    {
        return AsParam("any", Arb.Default.Int32());
        //could be constrained: AsParam("any", Gen.Choose(0, 100));
    }

    [StepArgumentTransformation("the first number")]
    public int TheFirstNumber()
    {
        return AsFormula(actualParams => actualParams.First().Value);
    }
}


```



```

@propertyBased
Scenario: Identity property
    Given I have entered any number into the calculator
    And I have entered 0 into the calculator
    When I press add
    Then the result

```



IdentityProperty

Source: Addition.feature line 11

✖ Test Failed - IdentityProperty

Message: TestCleanup method MyCalculator.Tests.AdditionFalsifiable, after 87 tests (4 shrinks) (StdGen (284620898,296

Original:
74

Shrunk:
38

More real life examples...

- [-] **Scenario:** Should be able to choose color
 - Given the user is logged in
 - And selected a *product with color variations*
 - When the user adds the product to the basket
 - Then it should be able to choose the color

- [-] **Scenario:** Restricted pages require login
 - Given the user has not logged in yet
 - When the user tries to access a *restricted page*
 - Then the user should be redirected to the login page

- [-] **Scenario:** Do not let the user type while driving
 - Given the vehicle is driving with speed *greater than 5 km/h*
 - When the driver attempts to type in an address
 - Then a warning should be displayed

Summary

- BDD turns examples into automated tests
- PBT automates rules with many different input
- The power of this two can be combined to achieve an executable specification
- The BDD and the PBT tools can work together for this
- See <http://github.com/gasparnagy/SpecFlow.FsCheck>



QUESTIONS?

gaspar@specsolutions.eu
[@gasparnagy](#)
<http://gasparnagy.com>

Special thanks to

- Ciaran McNulty
- Konstantin Kudryashov