PROPERTY BASED BDD EXAMPLES

Presented by Gaspar Nagy

gaspar@specsolutions.eu
@gasparnagy, http://gasparnagy.com
<table>
<thead>
<tr>
<th>BDD</th>
<th>PBT</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Behavior Driven Development</td>
<td>• Property Based Testing</td>
</tr>
<tr>
<td>• ~Specification by Example (SbE)</td>
<td>• Property Testing</td>
</tr>
<tr>
<td>• ~Acceptance Test Driven Development (ATDD)</td>
<td>• ~Random Testing</td>
</tr>
<tr>
<td>• ~Keyword Driven Testing</td>
<td>• ~Model-based Testing</td>
</tr>
</tbody>
</table>
What is BDD? – Classic model

As a team
I want to get scores for my answers
So that I can compare my results with other teams

Implement feedback

[HttpPost]
public ActionResult Answer(int answer)
{
    TriviaEntities db = new TriviaEntities();
    var question = db.FindQuestion(CurrentQuestion);
    if (question.Type == QuestionType.Easy)
    {
        db.AddScore(question, user, 10);
    }
    else
    {
        db.AddScore(question, user, 50);
    }
    var model = new GameModel
    {
        Score = GetScore(question, user)
    }
    return View(model);
}
What is BDD?

Scenario: Correct easy answer scores 10
Given I register a team
When I submit a correct easy answer
Then my score should be 10
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 entered 1 into the calculator
And I have entered 2 into the calculator
When I press add
Then the result should be 3 on the screen

- 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
Scenario Outlines

Scenario Outline: Add two numbers
  Given I have entered \(<a>\) into the calculator
  And I have entered \(<b>\) into the calculator
  When I press add
  Then the result should be \(<result>\) on the screen

Examples:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>-7</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
BDD is for...

understanding & validating
business requirements
through illustrative examples
PBT is for...

verifying implementation 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: \(a \times (b + c) = a \times b + a \times c\)

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

```csharp
[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...

```csharp
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)
Shrunken:
(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

```csharp
[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_beEXPECTEDRESULT_on_the_screen(int expectedResult)
    {
        Assert.AreEqual(expectedResult, calculator.Result);
    }
}
```

Feature: Addition

Scenario: Add two numbers
GIVEN I have entered 1 into the calculator
AND I have entered 2 into the calculator
WHEN I press add
THEN the result should be 3 on the screen

AddTwoNumbers
Source: Addition.feature line 3
Test Passed - AddTwoNumbers
Elapsed time: 34 ms
Output
Identity property BDD style...

```java
@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...

```csharp
[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);
    }
}
```
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@specolutions.eu
@gasparnagy
http://gasparnagy.com

Special thanks to
• Ciaran McNulty
• Konstantin Kudryashov

© All rights reserved