Requirements Engineering – DAT230/DIT276

Behavioral Driven Development

Emil Börjesson, 2010

Background

- Test Driven Development
  - Came out of eXtreme Programming (Agile)
  - Build the test, run the test, fail, implement the code, and iteratively test until the test passes.
  - Used by Microsoft, IBM, and more.
- Acceptance testing
  - Usually at the end of development (more often in Agile)
  - Focus on VALIDATION rather than VERIFICATION

NUnit test (C#)

```
using System;
using NUnit.Framework;

namespace UnitTestApplication.UnitTests
{
    [TestFixture()]
    publicclass Calculator_UnitTest
    {

        private UnitTestApplication.Calculator
    calculator = new Calculator();

        [SetUp()]
        publicvoid Init()
        {
            // some code here, that need to be
               run at the start of every test case.
        }

        [TearDown()]
        publicvoid Clean()
        {
            // code that will be called after each
               Test case
        }

        [Test]
        publicvoid Test()
        {

        }
    }
}
```

Behavioral driven development (BDD)

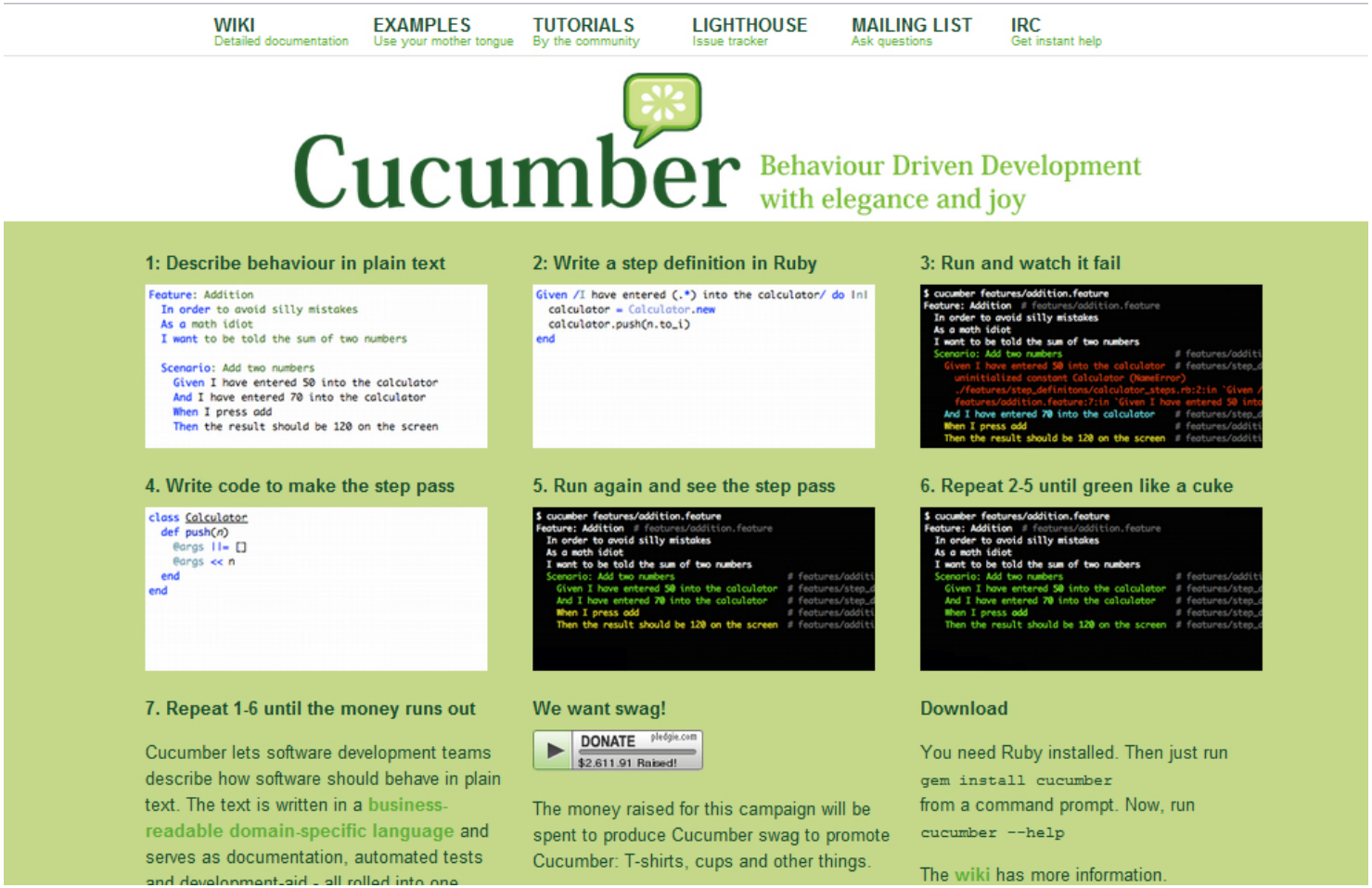
- TDD + Acceptance testing -> BDD!
  - Capture structured requirements within RUNNABLE Natural language test-cases!
  - Removes the interpretation phase between Requirements elicitation to test-case (Somewhat) and raises overall Validity

Cucumber

- It's a tool
  - Uses Ruby definitions, (base language) but can be used to test a lot of different language, such as Java, Pearl and so on.
- Has been used to develop iPhone apps
- Integration level testing
  - As well as touching on Unit-tests depending on scenario granularity
  - RSpec for Unit tests
- Installation: It's a Ruby gem! :D

  "gem install cucumber"

Cukes.info


WIKI
Detailed documentation
EXAMPLES
Use your mother tongue
TUTORIALS
By the community
LIGHTHOUSE
Issue tracker
MAILING LIST
Ask questions
IRC
Get instant help
Cucumber
Behaviour Driven Development
with elegance and joy
1: Describe behaviour in plain text
Feature: Addition
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
2: Write a step definition in Ruby
Given /I have entered (.*) into the calculator/ do |n|
calculator = Calculator.new
calculator.push(n.to_i)
end
3: Run and watch it fail
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers
Given I have entered 50 into the calculator
uninitialized constant Calculator (NameError)
./features/step_definitions/calculator_steps.rb:2:in `Given /
features/addition.feature:7:in `Given I have entered 50 into
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
4. Write code to make the step pass
class Calculator
def push(n)
@args ||= []
@args << n
end
end
5. Run again and see the step pass
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
6. Repeat 2-5 until green like a cuke
$ cucumber features/addition.feature
Feature: Addition # features/addition.feature
In order to avoid silly mistakes
As a math idiot
I want to be told the sum of two numbers
Scenario: Add two numbers
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press add
Then the result should be 120 on the screen
7. Repeat 1-6 until the money runs out
Cucumber lets software development teams describe how software should behave in plain text. The text is written in a business-readable domain-specific language and serves as documentation, automated tests and development-aid - all rolled into one
We want swag!
DONATE
pledgie.com
$2,611.91 Raised!
The money raised for this campaign will be spent to produce Cucumber swag to promote Cucumber: T-shirts, cups and other things.
Download
You need Ruby installed. Then just run
gem install cucumber
from a command prompt. Now, run
cucumber --help
The wiki has more information.

Scripts

- Keywords
  - Feature (General layout)
    - In order to <achieve something I value>
    - A user <such as you in your role>
    - Should/Would/Can <have access to this feature>
  - Scenario(s)
    - Given - Prerequisites
    - When - Action
    - Then - Goal

Example

Feature: Manage companies

In order to keep track of companies

A user

Should be able to manage companies

Scenario: Create a new company

Given I am logged in

When I create a new company named Acme

Then I should see that a company named Acme exists

Another Example

Feature: Happy Lecturer

In order to be happy

As a lecturer

I want everyone to learn BDD

Scenario: Teaching BDD

Given I have a class

And I have a prepared presentation

When I show the presentation

Then the students should learn something about BDD

Yet another Example ;)

Scenario: Teaching more BDD

Given I have a class

And class doesn’t want to fail the exam

When this lecture is over

And class are at home

Then class will read more about BDD on their own

Pros and Cons

- Pros
  - Structure
  - Valid test generation/less or no interpretation
  - Multi purpose (Test and Requirement)
- Cons
  - Low level
  - Feature alignment

Code scripting

- Develop test-code that fulfills the natural language test.

- Example shown in class, and for further home studies look at:
  - http://railscasts.com/episodes/155-beginning-with-cucumber

Railroad crossing

You are designing the software for a railway crossing. A sensor on the rails detects when a train is arriving and lowers the bars over the road. The bars remains lowered until another sensor detects that the train has passed or until a signal from the Railway Control Centre(RCC) is received. If the sensors malfunction, or if the connection to the RCC is lost, the bars are lowered and shall remain lowered until everything is working again and they are reset from the RCC. While the bar is being lowered or raised and while the bar is in its lowered position, red lights will flash and a bell will ring. When the bars are in their upper, un-lowered position a white light is shown (not flashing).

Task for this class

- Write BDD scripts based on the Rail-road crossing description.
- Focus is on elicitation! Use the structure of the BDD script.
- You don’t have to write the code! ☺
  - You can write Pseudo-code if you want.
- We then discuss the solutions together!

Example Script

Feature: Normal operation

In order to keep crossroad safe and available
a crossroad
should be able to manage bars lights and bells

Scenario: Open passage when train is gone

Given there is a train at all
And the system is working
When a train has passed
And no trains are coming
Then bars are raised
And lights a lit
And bell is off

Example Script

Feature: Fault tolerant operation
  In order to keep crossroad safe in presence of faults
  the Railroad crossing
  should be able put the system in a safe state

Scenario: Connection to RCC is lost
  Given that the systems exists
  And the system is working
  When connection is lost
  Then lower bars
  And flash red lights
  And ring the bell

Example Script

Feature:

In order

As a

I should

Scenario:

Given

When

Then

Example Script

Feature: Train passes the bar

In order the train to pass the railroad crossing

As a train

I should be able to get passed the bars

Scenario: Raising the bar

Given that the second sensor isn’t sending a signal

And the light is red

When the train passes the second sensor

Then raise the bar

And turn on the white lights

Example Script

Feature: Safety

In order to arrange a safe passage
for a trafficant
The railroad crossing should be closed when there is a train

Scenario: Turn on the white light

Given the light is red
When train has passed the second sensor
And the RCC isn't broken
And the bars are up
Then switch to white light

BDD Script 1:

Feature: Passing the railroad crossing

In order to cross the rails safely

As a traffic participant

Should find the crossing safe for passage

Scenario: Passing the crossing

Given that the bars are raised

And the bells do not ring

And the lights do not flash

When the traffic participant arrives at the crossing

Then the traffic participant should be able to pass safely

BDD Script 2

Feature: Crossing closed
In order to close the railway crossing
As a railway crossing
I want the train to be between detector 1 and 2

Scenario: Train between detector 1 and 2
Given detector 1 is working
And detector 2 is working
When train has passed detector 1
And train has not passed detector 2
Then bars should be lowered
And red light should be flashing
And bells should ring

Scenario: Broken detector
Given detector 1 or 2 are broken
And bars have not been reset by RCC
When traffic participant reaches crossing
Then the bars should be lowered