Behaviour Driven Development with Cucumber for Java

Thomas Sundberg
Developer 20+ years
Master Degree in Computer Science

I write computer programs.

@thomassundberg
tsu@kth.se
http://thomassundberg.wordpress.com
Goal

Introduce Cucumber for Java
Tools are useless

If you don't know how to use them
Outline

Why
What
How
Why

Communication
Bandwidth
1 Car Maintenance

Car need to be maintained. This document defines how the maintenance should be done.

A colleague of mine said a while ago that he used to ask that question to a certified scrum trainer. The trainer always answers that he will have to get back with an answer. One can speculate why the person who got the question didn’t have a good answer ready. The are a few options, either he does not know what he does because he has a passion for it or because he is good at it and it pays good money.

The question is valid, not only to a scrum trainer, but to anyone who sells his services. It could be a consultant or perhaps someone applying for a job.

1.1 Background

Without maintenance all cars will enter a state where they can’t be used due to breakdown, insufficient fuel or similar.

It happens that I asked interviews with candidates that are applying for a job as developers. I always try to find out if they have a passion for their profession or not. If you have a passion for what you do, you will be good at it after a while. You may not be the best developer yet, but since you have a passion you have a good chance of becoming a great developer.

In passion always good? It depends. I have a friend who is studying to become a nurse. She was overestimated her first time in school because many of the other students had a fantastic passion, or possible a fanatic passion, for nursing. This is of course a good thing. But if they are willing to do the job without getting paid, they will of course do the job without getting paid. Or with a very bad salary. This may not be the best thing for you as a person if your colleague will do the same job as you, but will do it for free. You will have a hard time getting paid in that situation. And that is the case with nurses both in Sweden and in Finland.

1.2 Fuelling

A car with an empty gas tank need to be fuelled. After fuelling the car, it should be operational again until the next time it needs fuel.

The same situation applies to musicians. Professional musicians are often paid really bad. The reason is simple, there are a lot of amateur musicians that will play for free. The amateurs aren’t as good as the professionals, but the normal listener can’t hear the difference. Chances are that you who are reading this don’t understand the difference. I am an amateur musician. I play the trombone, and I know that I can’t always hear the difference.

There are cases that are different. Doctors are often paid better than nurses, but to become a good doctor you need passion. But in this case, they have realized that they are confident for two.
A specification

1.3 Test cases
To be defined.
Examples

Create concrete examples

Mark Twain:
There is nothing so annoying as a good example
Executable

How should the examples be expressed?
@Test
public void orderLessThenAMonthBeforeChristmasShouldGiveZeroShipping() {
    Calendar purchaseDate = GregorianCalendar.getInstance();
purchaseDate.set(Calendar.YEAR, 2012);
purchaseDate.set(Calendar.MONTH, Calendar.NOVEMBER);
purchaseDate.set(Calendar.DAY_OF_MONTH, 24);

    int christmasEve = 24;
    int expectedShipping = 0;
    String expectedCurrency = "euro";

    Date purchase = purchaseDate.getTime();
    Shipping shipping = new Shipping(purchase, christmasEve);

    int actualCost = shipping.getCost();

    assertThat(actualCost, is(expectedShipping));
}
Scenario: Free shipping a month before Christmas
Given a customer that Christmas is celebrated 24 of December
When a customer buys a book on 2012-12-10
Then the shipping cost should be 0 euro
@Test
public void orderLessThenAMonthBeforeChristmasShouldGiveZeroShipping() {
    Calendar purchaseDate = GregorianCalendar.getInstance();
    purchaseDate.set(Calendar.YEAR, 2012);
    purchaseDate.set(Calendar.MONTH, Calendar.NOVEMBER);
    purchaseDate.set(Calendar.DAY_OF_MONTH, 24);
    int christmasEve = 24;
    int expectedShipping = 0;
    String expectedCurrency = "euro";
    Date purchase = purchaseDate.getTime();
    Shipping shipping = new Shipping(purchase, christmasEve);
    int actualCost = shipping.getCost();
    assertThat(actualCost, is(expectedShipping));
}

Scenario: Free shipping a month before Christmas
Given a customer that Christmas is celebrated 24 of December
When a customer buys a book on 2012-12-10
Then the shipping cost should be 0 euro

Which is easier to understand?
Regression testing

Can we deliver?
Living documentation

These examples are actually working
What
Behaviour Driven Development
Behaviour Driven Development
Black box
High level

Not always end to end
Not always through the GUI
Common language

Used by all involved
  Customer
  Developers
  Testers
Three core principles

Business and Technology should refer to the same system in the same way

Any system should have an identified, verifiable value

Up-front analysis, design and planning all have a diminishing return
How
The right tool for the job
Audience

Readers
  Customers
  Developers

Maintainers
  Product owner
  Developers
Why Cucumber

It is one of the least technical tools
It descends from RSpec
It is a very active open source project
  Official release Mars 2012
It supports a variety of languages
Pattern

System under test, SUT

Steps – Glue (Native code)

Execute Specification

Execute SUT

Assert SUT

Specification
Gherkin

Small API

Features

- Scenarios
- Background
- Scenario outlines

Translated to 47 languages
Keywords

Given – setup SUT
When – execute SUT
Then – verify SUT
And, But – flow to the language
Feature: Hello World

Scenario: Say hello
Given I have a hello app with "Howdy"
When I ask it to say hi
Then it should answer with "Howdy World"
Advantages

Easy to read
Easy to understand
Easy to discuss
Easy to parse
Features

Order not relevant

When
Given
Then
Backgrounds

Executed before each scenario

Powerful
Scenario outline

Substitute keywords with values from a table
Each row in the table makes up one scenario
Steps – Glue

Java methods

Global

   You can't describe two different things with the same words

Annotated with the keywords

   Found by regular expressions
@When("^I add (\d+) copies the (.*) book$")

public void methodName(int copies, String title) throws Throwable {
    StepHelper stepHelper = new StepHelper();
    stepHelper.addCopies(title, copies);
}

An example
Hooks

@Before
@After
Executed before or after each step
  Very often
Transformers

Transform a String to a class

@When("^a customer buys a book on (.*)$")

public void methodName(@Format("yyyy-MM-dd") Date purchaseDate) throws Throwable {
    this.purchaseDate = purchaseDate;
}


Drivers

JUnit

Command line
Continuous Integration

Maven

Ant
Example

A Maven project
Simple model
Use a Continuous Integration server
Extend to a web application
Live coding
Small example

All large systems consists of small pieces

You can only view a small portion of a system at one time

~30 – 50 loc

http://www.casualmiracles.com/2010/02/21/large-systems/
Workflow
1. Describe the 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
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```
2. Write a step definition in Ruby

```ruby
Given /I have entered (.*\*) into the calculator/ do
  calculator = Calculator.new
  calculator.push(n.to_i)
end
```
3. Run it 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  # features/addition.feature
   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
```
4. Write code to make the step pass

class Calculator
  def push(n)
    @args ||= []
    @args << n
  end
end
5. Run it again and see the step pass
6. Repeat step 2 – 5 until green like a Cuke

```bash
$ cucumber features/addition.feature
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
```
7. Repeat step 1 – 6 until the money runs out

1. Describe the behaviour in plain text
2. Write a step definition
3. Run it and watch it fail
4. Write code to make the step pass
5. Run it again and see the step pass
6. Repeat step 2 – 5 until green like a Cuke
Why

Communication
Regression testing
Living documentation
What

Behaviour
How

Features
Scenarios
Glue code
System under test
Continuous Integration

Preparation for fast deployment

It works as we have specified
Declarative

Don't write scripts
Define what should work
No implementation details
Cucumber is a good hammer
All problems are not nails
Right tool for the right problem
Do not focus on the tools

Tools will never solve the problem

A fool with a tool is still a fool
Resources

Cucumber – http://cukes.info/
Cucumber – https://github.com/cucumber/cucumber/wiki
Selenium – http://seleniumhq.org/
Maven – http://maven.apache.org/
Jenkins – http://jenkins-ci.org/
Blog – http://thomassundberg.wordpress.com/
Behaviour Driven Development with Cucumber for Java

Thomas Sundberg
Developer 20+ years
Master Degree in Computer Science

I write computer programs.

@thomassundberg
tsu@kth.se
http://thomassundberg.wordpress.com