BDD: Behavior Driven Development

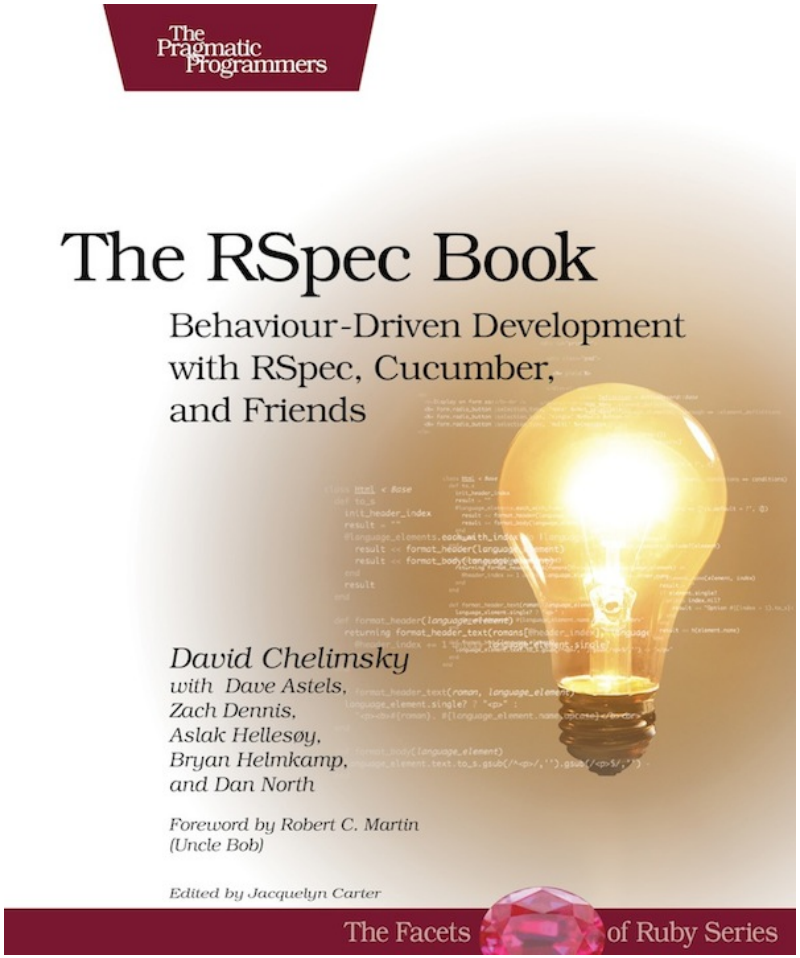
Felipe Kaufmann

So what is BDD?

- An agile software development technique
- Aims to bring developers and non-technical stakeholders closer together

- Originally coined in 2003 by Dan North, author of JBehave, the first BDD tool
- Evolved from an extension of TDD (TDD done well) to much more
- Critics still say it’s just a fancy word for TDD, but make your own opinion on that!

The RSpec Book



The RSpec Book says…

- Behavior Driven Development is about implementing an application by describing its behavior from the perspective of its stakeholders
- **Writing software that matters**

BDD Principles

- Enough is enough (don’t plan to much ahead)
- Up-Front analysis, design and planning all have a diminishing return over time

BDD Principles

- Deliver stakeholder value (don't do it if it does not add value)
- Any feature should have an identified, verifiable value to the business

BDD Principles

- It’s all behavior (everything can be described in behavior)
- Business and Technology should refer to the same feature in the same way

TDD (reminder/short intro)

- Write tests before you code (let them drive the code)
- red/green/refactor
- YAGNI

Some issues with TDD (practical issues of the developers)

- Where to start
- What to test and what not to test
- How much to test in one go
- How to understand why a test fails and what it means

More issues with TDD (organizational issues)

- (hopefully) written acceptance criteria on story cards
- (maybe, some kind of) acceptance criteria in the code (tests)
- **Duplication and/or divergence!**

Divergence

- The tests in the code do not correspond to the acceptance criteria in a user story
- Why not?
- They are much lower level and to abstract to match any kind of relevant business logic

Duplication

- The tests in the code do test the business logic, and thus do, to some extent, repreent acceptance criteria. But are not understood by stakeholders
- Why not?
- The stakeholders are non-technical and can not make sense out of Unit Tests etc.

We don’t know what we are doing!

- Because developers and stakeholders have two different points of view
- Because we don’t know what to test and where to start testing
- Because we don’t understand what it means when a test fails

Inspect and adapt!

- We need a better TDD!
- Tests that work on a higher level and correspond to acceptance criteria of a user story
- Tests (or test output) that are understandable by non-technical stakeholders

Pick your tool

- Cucumber (Ruby, Java)
- RSpec (Ruby)
- Concordion (Java)
- Jbehave (Java)
- EasyB (Java)
- …

Cucumber: The Holy Grail of BDD?

Cucumber Example Input

```
1  Feature: Beta Signup Page
2    In order to sign up for the beta
3    As a User
4    I want to read some information about the product
5    And I want to be able to sign up for the beta
6
7    Scenario: Visiting the Homepage
8      Given I am a User
9      And I am on the root page
10     Then I should see "The super duper product"
11     And I should see a the beta sign up form
12
13   Scenario: Submitting the sign up form with valid data
14     Given I am a User
15     And I am on the root page
16     When I fill in the following:
17       | First Name | John |
18       | Last Name | Doe |
19       | E-Mail | john.doe@something.com |
20     And I press "Submit"
21     Then my information should be saved
22     And I should see "Thank you for signing up!"
23
```

Cucumber Example Output

```
Feature: Beta Signup Page
  In order to sign up for the beta
  As a User
  I want to read some information about the product
  And I want to be able to sign up for the beta

  Scenario: Visiting the Homepage                # features/beta_signup_page.feature:7
    Given I am a User                            # features/step_definitions/demo_steps.rb:1
    And I am on the root page                    # features/step_definitions/web_steps.rb:44
    Then I should see "The super duper product"  # features/step_definitions/web_steps.rb:105
    And I should see a the beta sign up form     # features/step_definitions/demo_steps.rb:4

  Scenario: Submitting the sign up form with valid data # features/beta_signup_page.feature:13
    Given I am a User                                   # features/step_definitions/demo_steps.rb:1
    And I am on the root page                           # features/step_definitions/web_steps.rb:44
    When I fill in the following:                       # features/step_definitions/web_steps.rb:79
      | First Name | John                 |
      | Last Name  | Doe                  |
      | E-Mail     | john.doe@something.com |
    And I press "Submit"                                # features/step_definitions/web_steps.rb:52
    Then my information should be saved                 # features/step_definitions/demo_steps.rb:8
    And I should see "Thank you for signing up!"        # features/step_definitions/web_steps.rb:105

2 scenarios (2 passed)
10 steps (10 passed)
0m0.957s
```

RSpecExample Input

```
    @dummy.selected_day(@boardstock_day).should == Event::DATES.first
  end
end

context 'before or after the boardstock event' do
  it 'returns the first boardstock event day when no params given' do
    @dummy.selected_day(@some_other_day).should == Event::DATES.first
  end

  it 'returns the date from the params if given and param is valid' do
    @dummy.selected_day(@some_other_day, :day => '2011-07-22').should == Event::DATES.f
  end

  it 'returns the first boardstock event day when param is invalid' do
    @dummy.selected_day(@some_other_day, :day => 'boom').should == Event::DATES.first
  end

  it 'returns the first boardstock event day when param is not a boardstock event date'
    @dummy.selected_day(@some_other_day, :day => '2011-11-11').should == Event::DATES.f
  end
```

RSpecExample Output

```
EventsSelection
  #selected_day
    during the boardstock event
      returns the current day
    before or after the boardstock event
      returns the first boardstock event day when no params given
      returns the date from the params if given and param is valid
      returns the first boardstock event day when param is invalid
      returns the first boardstock event day when param is not a boardstock event date
```

Agiledox example

```
1  /**
2   * agiledox example (JAVA)
3   */
4
5  public class CustomerLookupTest extends TestCase {
6      testFindsCustomerById() {
7          /* ... */
8      }
9      testFailsForDuplicateCustomers() {
10         /* ... */
11     }
12     /* ... */
13
14 /**
15  *
16  * CustomerLookup
17  * - finds customer by id
18  * - fails for duplicate customers
19  * - ...
20  *
21  */
```

Inspect and adapt!

- **Clear workflow**
- From a business requirement/vision
- To features & user stories
- To acceptance tests and unit tests
- That drive the actual implementation

BDD Workflow: Planning

- Priorization first:
- Establish next best “vision” that will deliver value to the business
- Example: we need potential users for our beta launch

BDD Workflow: Planning

- Establish what is needed to achieve the vision, e.g:
- Product Owner wants to inform potential users about the product
- The users want to be able to sign up for the beta
- The developer that will work with the data want that only valid email adresses get trough the signup process

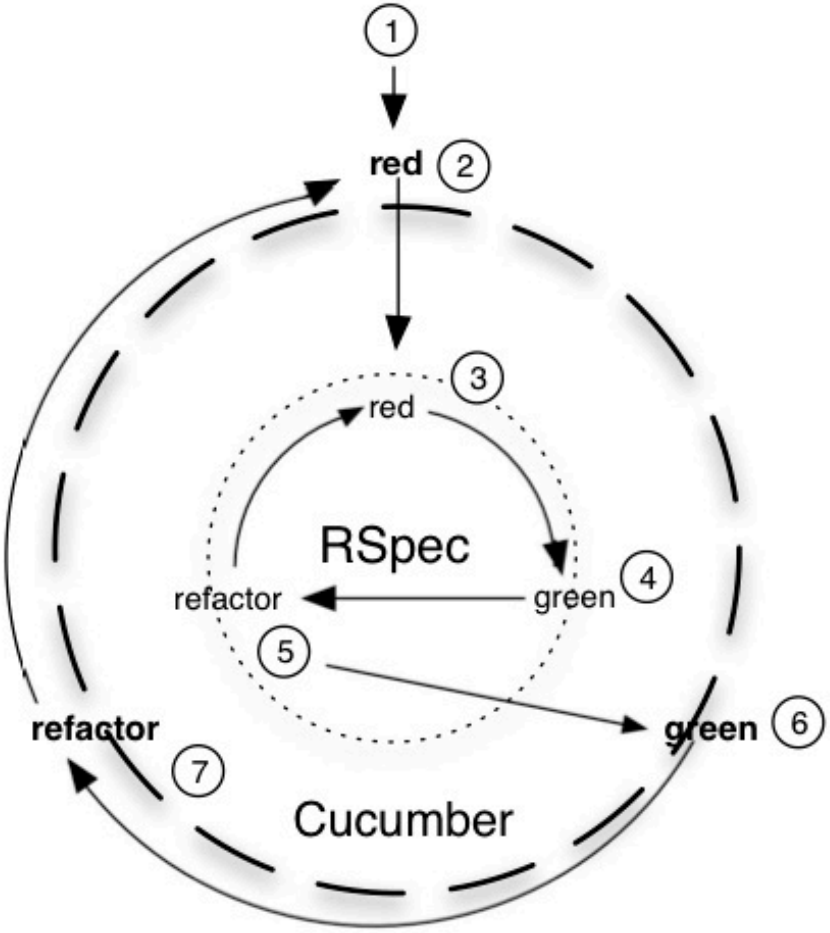
BDD Workflow

- Formalize the requirements with scenarios and examples
- See: https://github.com/effkay/bdd-presentation/blob/master/EXAMPLE_APPLICATION/features/beta_signup_page.feature for an example cucumber feature

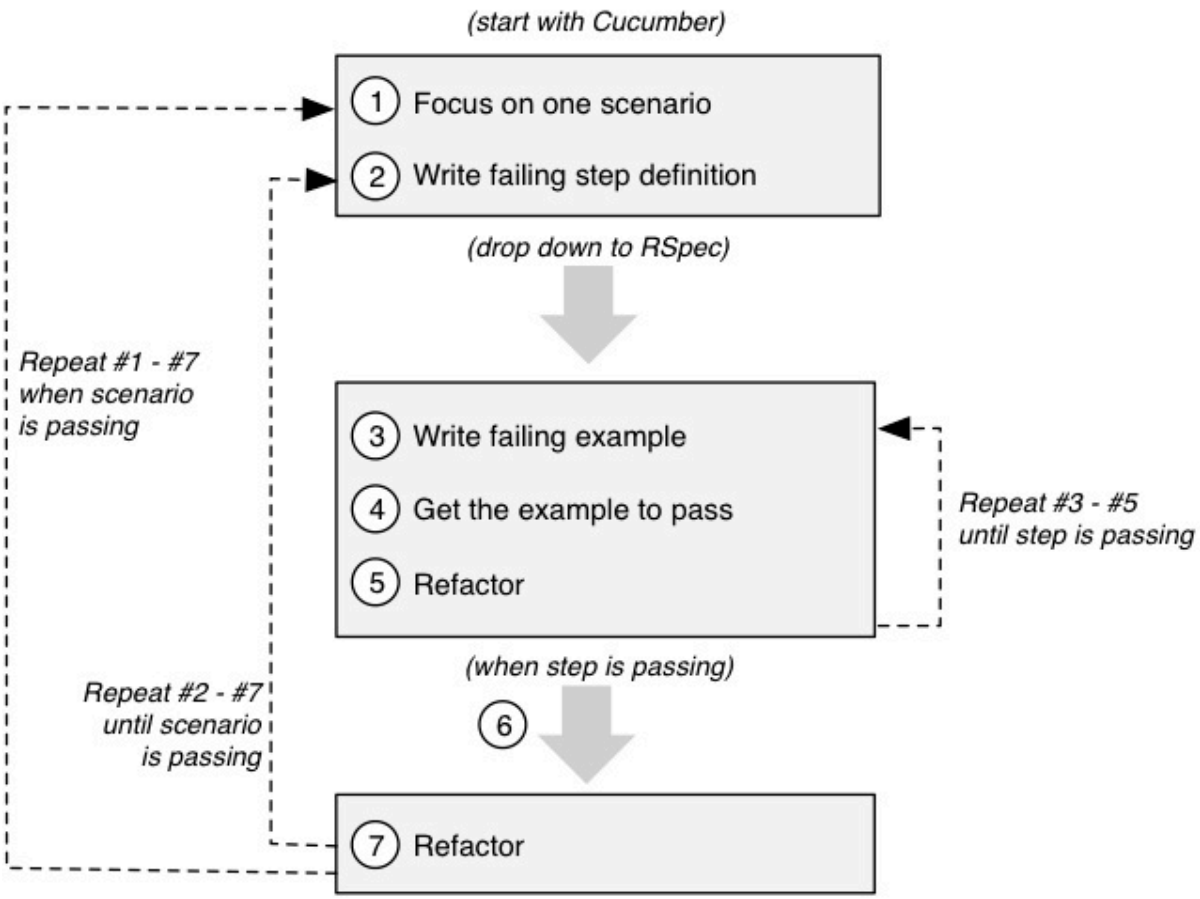
BDD Workflow: BDD Cycle

- Outside in development workflow
- Run tests all the time, automatically, while coding

BDD Cycle


1
red
2
3
red
RSpec
refactor
green
4
5
refactor
green
6
7
Cucumber

BDD Cycle


(start with Cucumber)
1 Focus on one scenario
2 Write failing step definition
(drop down to RSpec)
3 Write failing example
4 Get the example to pass
5 Refactor
(when step is passing)
6
7 Refactor
Repeat #1 - #7 when scenario is passing
Repeat #2 - #7 until scenario is passing
Repeat #3 - #5 until step is passing

The added value of BDD

- Acceptance criterias of a user story can be formalized
- One clear definition of what is necessary for the user story to be complete
- Automated testing of the acceptance criteria

The added value of BDD

- Tracebility of any code back to specific stakeholders need
- Clear path for developers on TDD/BDD best-practices and workflow

Criticisim

- BDD is just a fancy word for TDD?
- Tools add a new layer of abstraction/ complexity and mean more work
- Tools might be to focused on web development

Demo Time!

- Find the demo application code on: https://github.com/effkay/bdd-presentation/tree/master/EXAMPLE_APPLICATION

Conclusion

- BDD helps when you get stuck with TDD (workflow)
- Improves inter-project communication trough clearer and formalized acceptance tests
- Improves developer involvement due to traceability of business value
- Once you try it, you either hate it or never code without it

Questions?

- The RSpec Book (ISBN: 978-1-93435-637-1)
- http://dannorth/introducing-bdd
- http://behaviour-driven.org
- http://forums.pragprog.com/forums/95/topics/3035
- All resources on: https://github.com/effkay/bdd-presentation