Behavior Driven Development (BDD)

Pavel Rabetski
Volvo Group Telematics
Agenda

- Introduction to BDD
- Cucumber BDD framework
- Live demo
- Pros/cons of BDD
- Questions?
Introduction to BDD
Introduction to BDD

```java
public class TestJAccount {

    @Test
    public void testWithdraw(){
        //...
    }

    @Test
    public void testWithdrawWithException(){
        //...
    }
}
```
Introduction to BDD

```java
public class MoneyWithdrawTest {

    @Test
    public void testSuccessWhenEnoughMoneyOnAccount() {
        //...
    }

    @Test
    public void testFailsWhenLackOfMoneyOnAccount() {
        //...
    }
}
```

MoneyWithdraw
- success when enough money on account
- fails when lack of money on account
Introduction to BDD

Behavior Driven Development
Introduction to BDD

I want ...

function X(){
    return Y;
}

If I click this button then ...

Tester

BA

Developer
Introduction to BDD
Introduction to BDD

- Documentation
- BA
- Ubiquitous requirements description
- Tester
- Test scripts
- Developer
- Application code
Introduction to BDD
Introduction to BDD

Behavior Driven Development
(how we write and test requirements)
Introduction to BDD

Story (feature):

As a [role] I want [feature] so that [benefit]

Example: As a customer I want to withdraw money from an ATM so that I don’t have to go to the bank
Introduction to BDD

Story acceptance criteria (scenario):

**Given** [initial context] **when** [event] **then** [outcomes]

Example: **Given** there is enough money on my account **when** I make a withdrawal **then** I get the expected amount of money from the ATM
Cucumber BDD framework
Cucumber BDD framework

Feature files → Cucumber → Test runner (JUnit) → Test scripts
Translation into code → Cucumber → SUT
Feature: Withdraw money
   In order to avoid going to the bank
   As a customer
   I want to withdraw money from an ATM

Scenario: Withdraw less money than the account has
   Given there is enough money on my account
   When I make a withdrawal
   Then I get the expected amount of money from the ATM
   And receipt is printed
Cucumber: parametrization

Gherkin logic

**Scenario:** Withdraw less money ...
  **Given** I have 200 SEK on my account
  **When** I withdraw 100 SEK
  **Then** I get 100 SEK from the ATM

**Scenario:** Withdraw more money ...
  **Given** I have 50 SEK on my account
  **When** I withdraw 100 SEK
  **Then** I get 0 SEK from the ATM

**Scenario:** Withdraw money from the account
  **Given** I have `<balance>` SEK on my account
  **When** I withdraw `<withdraw>` SEK
  **Then** I get `<received>` SEK from the ATM

**Examples:**

<table>
<thead>
<tr>
<th>balance</th>
<th>withdraw</th>
<th>received</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>
Cucumber: annotations

Gherkin logic

@prod
Scenario: Withdraw less money ...
  Given I have 200 SEK on my account
  When I withdraw 100 SEK
  Then I get 100 SEK from the ATM

@test
Scenario: Withdraw more money ...
  Given I have 50 SEK on my account
  When I withdraw 100 SEK
  Then I get 0 SEK from the ATM
BDD for complex systems
Live demo: OEM portal
Live demo: ATM
Pros/cons of BDD

- Level of abstraction for steps?
- Becomes complex for complex systems
- Lack of tool support
P.S. BDD in a context of FSM

**Given** I have 200 SEK on my account **when** I withdraw 100 SEK **then** I get 100 SEK from the ATM

- **Given**
  - initial
  - $x \leq \text{balance}$
  - give $x$ money
  - $\{\text{balance} := \text{balance} - x\}$

- **When**
  - $x \leq \text{balance}$
  - give $x$ money
  - $\{\text{balance} := \text{balance} - x\}$

- **Then**
  - success
  - error

- **x** - amount of money to withdraw
- balance - amount of money on the account
Questions?
References

Cucumber framework:
https://cucumber.io/

Gherkin language:
http://docs.behat.org/en/latest/guides/1.gherkin.html

Dan North about BDD:
http://dannorth.net/introducing-bdd/

How BDD can be misused:
https://cucumber.io/blog/2014/03/03/the-worlds-most-misunderstood-collaboration-tool