Behavior Driven Development

Venkatesh-Prasad Ranganath
Kansas State University

What’s in the name?

- Behavior-Driven Development
- Specification by Example
- Story Testing
- Example-Driven Development
- Agile Acceptance Testing
- Acceptance Test-Driven Development
- TDD with requirements??

Gherkin Language

- A line-oriented language that uses indentation to define structure.
- Feature is the top-level construct (test suite)
  - One feature per file
- Each feature is composed of scenarios (tests)
- Each scenario is composed of steps (test logic)
  - Having *Given…When…Then…* structure
- One step per line
- https://github.com/cucumber/cucumber/wiki/Gherkin

Feature

```
Feature: Transfer money between accounts
  To manage my money efficiently
  As a bank client
  I want to transfer funds between accts
```

Scenario

```
Scenario: Transfer money to savings
  Given my current account has $200
  And my savings account has $1000
  When I transfer $100 from my current
account to savings account
  Then I should have $100 in my current
account
  And I should have $1100 in my savings
account
```

Scenario

Different type of steps

- Given describes the context / state in which the scenario starts
- When describes the action(s) being performed as part of the scenario
- Then describes the expected outcome(s) of the performed actions in the scenario
- And/But extend the above steps

Mapping Steps to Code (in *behave* tool)

```
Scenario: Transfer money to savings
  Given my current account has $200
```

```
  @given('my current account has $200')
  def current_account_setup(context):
      context.current = account.Account(200)
```

Mapping Steps to Code (in *behave* tool)

```
Scenario: Transfer money to savings
  Given my current account has $200
```

```
@given('my current account has ${amt}')
def current_account_setup(context, amt):
    context.current = account.Account(int(amt))
```

Mapping Steps to Code (in *behave* tool)

```
Scenario Outline: Transfer money to savings
  Given my current account has $<init1>
  …
 Examples:
  | init1 | init2 | withdraw | balance1 | balance2 |
  | 200   | 1000  | 100      | 100      | 1100     |
  | 500   | 1000  | 200      | 300      | 1200     |
```

```
@given('my current account has ${amt}’)
def current_account_setup(context, amt):
    context.current = account.Account(int(amt))
```

Features w/ Scenarios

```
Feature: Transfer money between accounts
  To manage my money efficiently
  As a bank client
  I want to transfer funds between accts

  Scenario: Transfer money to savings account
    Given my current account has $200
    And my savings account has $1000
    When I transfer $100 from my current account
to savings account
    Then I should have $100 in my current account
    And I should have $1100 in my savings account
```

Using *behave* tool

- Install via `pip install behave`
- Feature files go in `features` folder
- Steps files go in `features/steps` folder
- Implementation can be placed `src` folder
- Execute `behave` in the folder containing `features` folder

Key Aspects

- Driven by business goals
- Collaboratively specification (dev team + users)
  - Uniform understanding of requirements
- Example-based illustrations
  - Each scenario serves as an example of how the corresponding feature is used or works
- Executable specifications
  - Each scenario can be executed using the mapping of steps to code
- Automatic and frequent validation
  - Spec execution exercises the implementation
  - Tight coupling between specs and implementation