**When and why use BDD?**

BDD or behavior driven development is a development process that emerged from TDD or test driven development.

This practice is designed to make these more accessible agile practices and effectives for the team work that require start with them. BDD has grown accompanied to growing the agile methodologies and the use test acceptation automatics .

BDD is largely facilitated through the use of a simple domain-specific language (DSL) using natural language constructs (e.g., English-like sentences) that can express the behavior and the expected outcomes.

BDD is considered as an effective technical practice especially when the "problem space" of the business problem to solve is complex.

The more aspect of BDD

* Is an evolution to agile practices
* Oftware development should be managed by both business interests and technical insight
* Solve problem complex.
* The tools serve to add automation to the ubiquitous language that is a central theme of BDD

Which type of teams could implement BDD

All teams that use agile methodologies

Structure of a feature

Feature: Serve coffee

Coffee should not be served until paid for

Coffee should not be served until the button has been pressed

If there is no coffee left then money should be refunded

Scenario: Buy last coffee

Given there are 1 coffees left in the machine

And I have deposited 1$

When I press the coffee button

Then I should be served a coffee

[TestFixture]

public class When\_the\_login\_handler\_is\_given\_a\_login\_and\_password

{

constant string login = "jdoe";

constant string password = "password";

static LoginValidator loginValidator;

context c = () => loginValidator = an<ILoginValidator>;

because b = () => sut.Validate(login, password);

it should\_validate\_the\_data\_with\_a\_LoginValidator =

() => loginValidator.was\_told\_to(x => x.DoValidation(login, password));

}

Structure of a user History

**Story**: Returns go to stock

**In order to** keep track of stock

**As a** store owner

**I want to** add items back to stock when they're returned.

**Scenario 1:** Refunded items should be returned to stock

**Given** that a customer previously bought a black sweater from me

**And** I have three black sweaters in stock.

**When** he returns the black sweater for a refund

**Then** I should have four black sweaters in stock.

**Scenario 2:** Replaced items should be returned to stock

**Given** that a customer previously bought a blue garment from me

**And** I have two blue garments in stock

**And** three black garments in stock.

**When** he returns the blue garment for a replacement in black

**Then** I should have three blue garments in stock

**And** two black garments in stock.

Structure of the Scenario

**private** Game game;

**private** StringRenderer renderer;

@Given("a $width by $height game")

**public** void theGameIsRunning(int width, int height) {

game = **new** Game(width, height);

renderer = **new** StringRenderer();

game.setObserver(renderer);

}

@When("I toggle the cell at ($column, $row)")

**public** void iToggleTheCellAt(int column, int row) {

game.toggleCellAt(column, row);

}

@Then("the grid should look like $grid")

**public** void theGridShouldLookLike(String grid) {

assertThat(renderer.asString(), equalTo(grid));

Differences BDT vs BDD

Behavior-driven development (BDD) is an agile testing methodology. BDT - Behavior driven testing, usually come after /during BDD (Behavior Driven development) but is not a must, it can be organized with traditional testing as well.