**Name: Franco Daniel Aldunate Cordero**

**BDD and BDT**

**When and why BDD was defined**

The term Behavioral Driven Development or BDD was introduced by Dan North in early 2000’s.

BDD was defined motivated by the arguing over the principles of TDD and the term “Test”. Developers didn’t want to take responsibility of writing test and testers were complaining with the fact that developers are not qualified to write good tests. Then the term “Behavior” was introduced, so instead of just writing tests, “examples” would be written, which means to specify what the code needs to do ahead of time and make that code executable. In other words to describe the behavior of the code before it actually exists and then make the application code behave in that way.

**The most important aspects of BDD**

The BDD approach can be divided into two main parts.

1. The practice of using examples written in ubiquitous language to illustrate behaviors, which means to write examples describing the behavior of the software.
2. The practice of using those examples as the basis of automated tests. As well as checking functionality for the user, which ensures that the system works as defined by the business throughout the project lifetime.

There are many benefits of using a BDD in the software development process:

* All development work can be traced back directly to business objectives.
* Software development meets user need. Satisfied users = good business.
* Efficient prioritization - business-critical features are delivered first.
* All parties have a shared understanding of the project and can be involved in the communication.
* A shared language ensures everyone (technical or not) has thorough visibility into the project’s progression.
* Resulting software design that matches existing and supports upcoming business needs.
* Improved quality code reducing costs of maintenance and minimizing project risk.

**Why is useful and when could be applied.**

When launching a new digital project, for example a web or desktop application, there can be disconnection between the following:

* The business being truly able to define the desired outcomes
* The developer’s understanding of what needs to be built
* The business’ understanding of the technical challenges their requirements may present

BDD can help a business and its technical team to deliver software that fulfills business goals. Business analysts, developers and testers can share a common language in terms of the behavior expected for the software. Instead of using terms for a specific field like business, BDD allows to use a language understandable for technical and non-technical project stakeholders. It helps to drive the developing process focusing the main goals of the business.

BDD is a process designed to aid the management and the delivery of software development projects by improving communication between engineers and business professionals. In so doing, BDD ensures all development projects remain focused on delivering what the business actually needs while meeting all requirements of the user.

The core premise of BDD is to gather business experts, developers and testers in one room to discuss and reach a common understanding of features of the product, their domain, scope, problems, etc.

BDD can be applied when there is an agreement to gather all participants of the development process, there must be structural or organizational capabilities on the team to apply BDD, this means to have the team in a geographic area which allows the interaction between them, or to have the resources to reach a good communication. Not have organizational boundaries like offshore testing, or participants splitted in different organizations, or a social division between roles like business analysts and testers.

**Which type of teams could implement BDD?**

BDD can be implemented by teams compounded by business analysts, developers and testers who:

* Have good communication and resources to reach it.
* Focus on collaboration to understand and define behavior of features that deliver business value.
* Have training on BDD approach
* Have commitment to follow a BDD approach

**Structure of a feature**

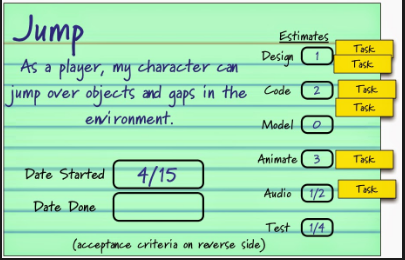
A Feature is the ‘What’ that a Stakeholder (the User or Customer) requires from the Product Owner, it means a capability that provides value to users. The Product Owner encapsulates these desires as Feature in the form of a User Stories in a ‘Wish List’, named the Product Backlog.

Structure:

The names should be short, snappy and instantly recognizable. Example: “Search for Customer”.

* Name
* Size [Optional]
* Priority

Example:



**Structure of a user story**

**User stories** are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

* The title should describe an activity
* The narrative should include a role, a feature and a benefit
* The scenario title should say what’s different
* The scenario should be described in terms of Givens, Events and Outcomes
* The givens should define all of, and no more than, the required context

Structure:

Title (one line describing the story)

Narrative:

As a [role]

I want [feature]

So that [benefit]

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title

Given [context]

  And [some more context]...

When  [event]

Then  [outcome]

  And [another outcome]...

Example:

Story: Account Holder withdraws cash

As an Account Holder

I want to withdraw cash from an ATM

So that I can get money when the bank is closed

Scenario 1: Account has sufficient funds

Given the account balance is \$100

 And the card is valid

 And the machine contains enough money

When the Account Holder requests \$20

Then the ATM should dispense \$20

 And the account balance should be \$80

 And the card should be returned

**Structure of a scenario**

A BDD scenario is a written description of a product’s behavior from one or more users’ perspectives. Scenarios are designed to reduce the cost of translation and make it easier for your engineers to understand the requirements and for your QA (if you have one) to test it properly.

Structure:

**'Given,** describes the initial context for the example  
**'When'** describes the action that the actor in the system or stakeholder performs  
**'Then'** describes the expected outcome of that action

Example:

Scenario: Account has insufficient funds

Given the account balance is \$10

 And the card is valid

 And the machine contains enough money

When the Account Holder requests \$20

Then the ATM should not dispense any money

 And the ATM should say there are insufficient funds

 And the account balance should be \$20

 And the card should be returned

**Differences between BDD and BDT**

The main differences between BDD and BDT are:

1. **Language:** BDT targets developers, so they write their specification according to their understanding of the feature, based on programming goals like write a test for the outputs of a function, write the code and make it pass, then refactor the code.

Instead BDD targets all participants of the development process, generates a testable documentation of specifications focusing on the behavior of the feature, this means that all participants can share knowledge and understanding about the specification.

1. **Artifacts:** As said before, BDT handles artifacts for developers, but BDD handles artifacts available and understandable by all the development process participants.
2. **Communication:** BDT is about developers writing their tests based on their understanding of the feature. Instead BDD is about discussion, agreement, sharing, understanding, and commitment between business analysts (product owner), developers and testers.
3. **Goal:** BDT goal is to implement code that satisfies tests, but BDD goal is to implement code that satisfies expected behavior of the feature and provides business value.