**Agile in context**

BDD is all about understanding the features that are important from a clients perspective and ahving examples about these features

Agile is a process of implementing guiding principles that allow teams and businesses to rapidly develop products that have value to their business stakeholders. **It aligns software development with the needs of customers while breaking down barriers between traditionallly isolated teams**. Large emphasis what the client wants and what can be done

Agile manifesto:

* **Individuals and interactions** over processes and tools
* **Working software** over comprehensive documentation

User story is during the consultation process, you parse the work to be done into functional increments. It is written by the perspective of an end user. Each user story contribute to the value of the product and help define its purpose. It is usually written by stakeholder, that incule clients, managers, developers or the users them selves. It is not equal to system requirements

Most common user tempalte is the the 5 W’s:

* who – for example as a user
* what – for example what I want to acces via my mobile
* why – because I might not always have a computer
* where
* want

Acceptance Criteria are the conditons that must be met for the product to be acceptable from a user perspective. These will be defined as a set of statements with well-established pass/fail results for both functional and non-functional requirements. Any variations will be taken into account. It helps prevent misscommunication

Well written Acceptance Criteria:

* Acceptance criteria are the basis for the testing systems
* Criteroa are written by clients, product owners, and/or development team
* Development team reviews acceptance criteria before writing tests

Types of Acceptance Criteria

* List-driven or rule-oriented 🡪 most usefull for creating a backlog
* Illustrated or scenario-oriented 🡪translating tasks into a format taht can drive bdd process

**TDD historically**

The TDD process:

1. Write a failing test – knowing the code will fail this test
2. Write the code taht allows for a passing test
3. Refactor the code if necessary
4. Repeat

Where does the TDD fall short:

* In the TDD process, developers want to know what to test, how much to test, and how to understand failing tests –what interface should be tested? how much should be tested?
* Without clear guidelines as to what should be tested and how, confusion and misunderstandings are common
* Presenting TDD in a different light through BDD allows the development process to avoid a number of pitfalls

So, What is BDD?:

* BDD takes principles from both **TDD and Agile and aims to present the needs of both business analysts and developers into a single framework**
* **Business needs are defined in code and are testable**
* Writing test with expressive names, sually in a form of sentance, taht decalred what a test was cheking

**Building the right thing**

The emergence of BDD comes from the failures fo the traditional model of software development

BDD is not for discovering what product to build. Product owners should now waht the end product is

BDD is for building the correct, reliable product that end users will love using

Starting points for BDD:

* Product owners already understand what product they desire
* Some, if not most, acceptance criteria have beed defined
* User stories have already been written

What if we are unclear with our acceptance criteria? – We use Example Mapping:

* Before utilizing a user story in development, you must clarify and confirm acceptance criteria
* Example mapping facilitates the conversation to explore the problem domain
* Example mapping provides a visual representaton of a user story

**Concrete Example**

BDD is about providing concrete examples around feature sets, so everybody understand how to implement a solution

Concrete Examples:

* User stories should be detailed and full of context – we want to discover exactly what the expectations of stakeholders are, and how they use the application in specific situations
* Well-defined examples allow for robust and focused tests – we know what to test adn how to test

Finding our Acceptance Criteria:

* Developers or QA seek clarity from business stakeholders to define acceptance criteria for an application
* This occurs from conversations with stakeholders based on concrete examples dreived from the user story
* The best way to finding is to ask the stakeholder to provide a specific example
* **Names, Context, Situational** awarness what we need

Acceptance criteria vs Scenario:

**Acceptance criteria:**

* Adresses what defines a working system
* Written as pass/fail

**Scenario:**

* Defines the initial conditions for acceptance criteria
* States the trigger of scenario an expected outcome

**Scenario Format:**

* **Scenario:** a user story
* **Given:** someset of inital conditons
* **When:** an event occurs
* **Then:** an outcome is expected