BDD

Behavior Driven Development

By:- Bereket Woldesilasie

Agenda

Introduction

What is BDD

Key features of BDD

Cucumber

Demo

Advantage and Disadvantage of Bdd

Conclusion

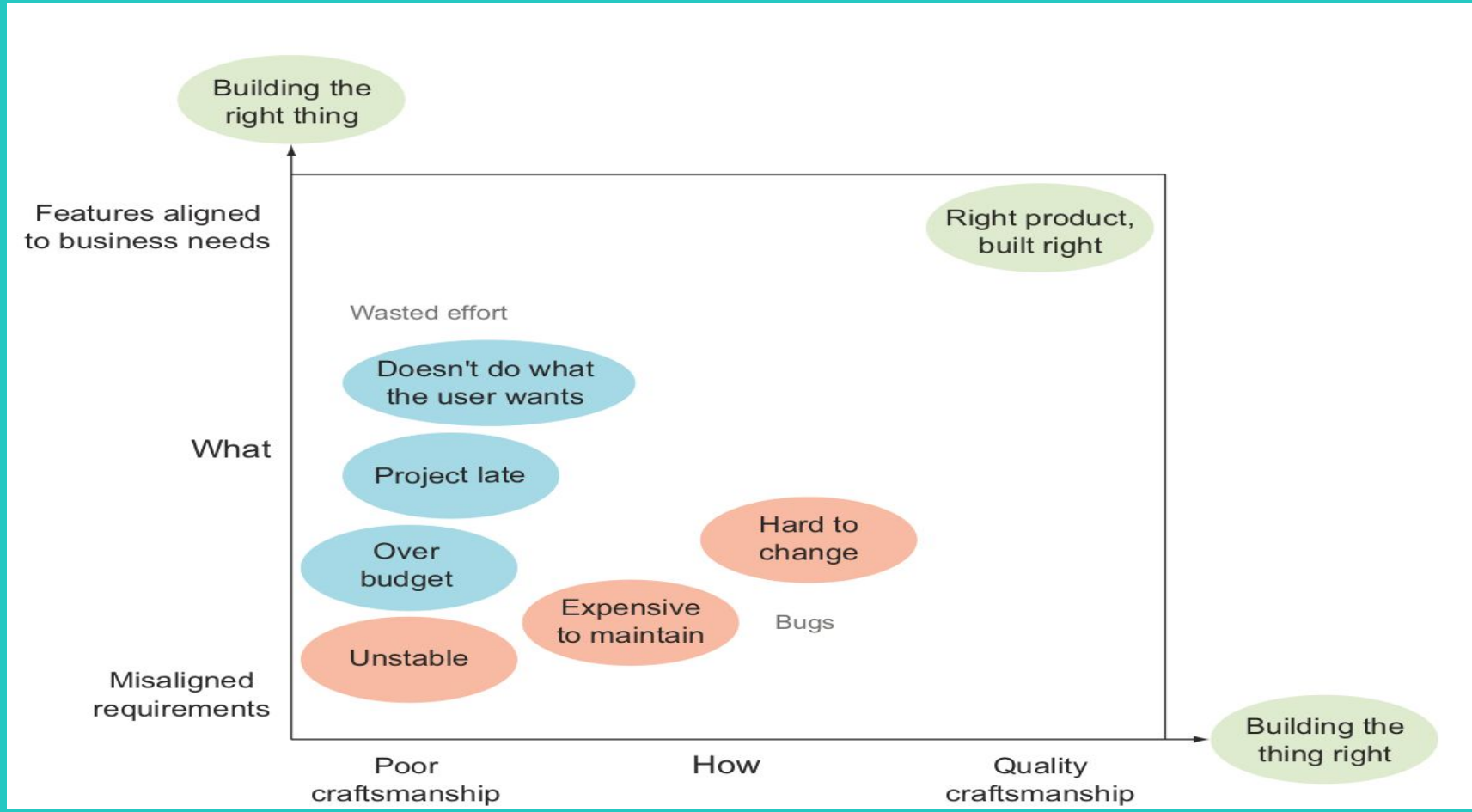
Introduction

The most crucial aspect of building software is to create software that delivers tangible value to its users.

In 2012, the U.S. Air Force decided to ditch a major software project that had already cost over $1 billion USD. The Expeditionary Combat Support System(ECSS) was designed to modernize and streamline supply chain management in order to save billions of dollars and meet new legislative requirements. The Air Force estimated that an additional $1.1 billion USD would be required to deliver just a quarter of the original scope.

Software projects fail for many reasons, but the most significant causes fall into two broad categories:

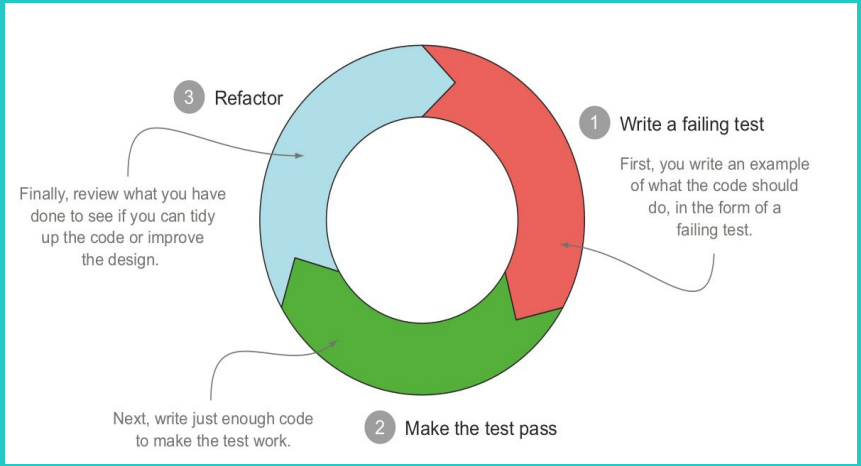
- Not building the software right
- Not building the right software


Building the right thing
Features aligned to business needs
Right product, built right
Wasted effort
Doesn't do what the user wants
What
Project late
Hard to change
Over budget
Expensive to maintain
Bugs
Unstable
Misaligned requirements
Building the thing right
Poor craftsmanship
How
Quality craftsmanship

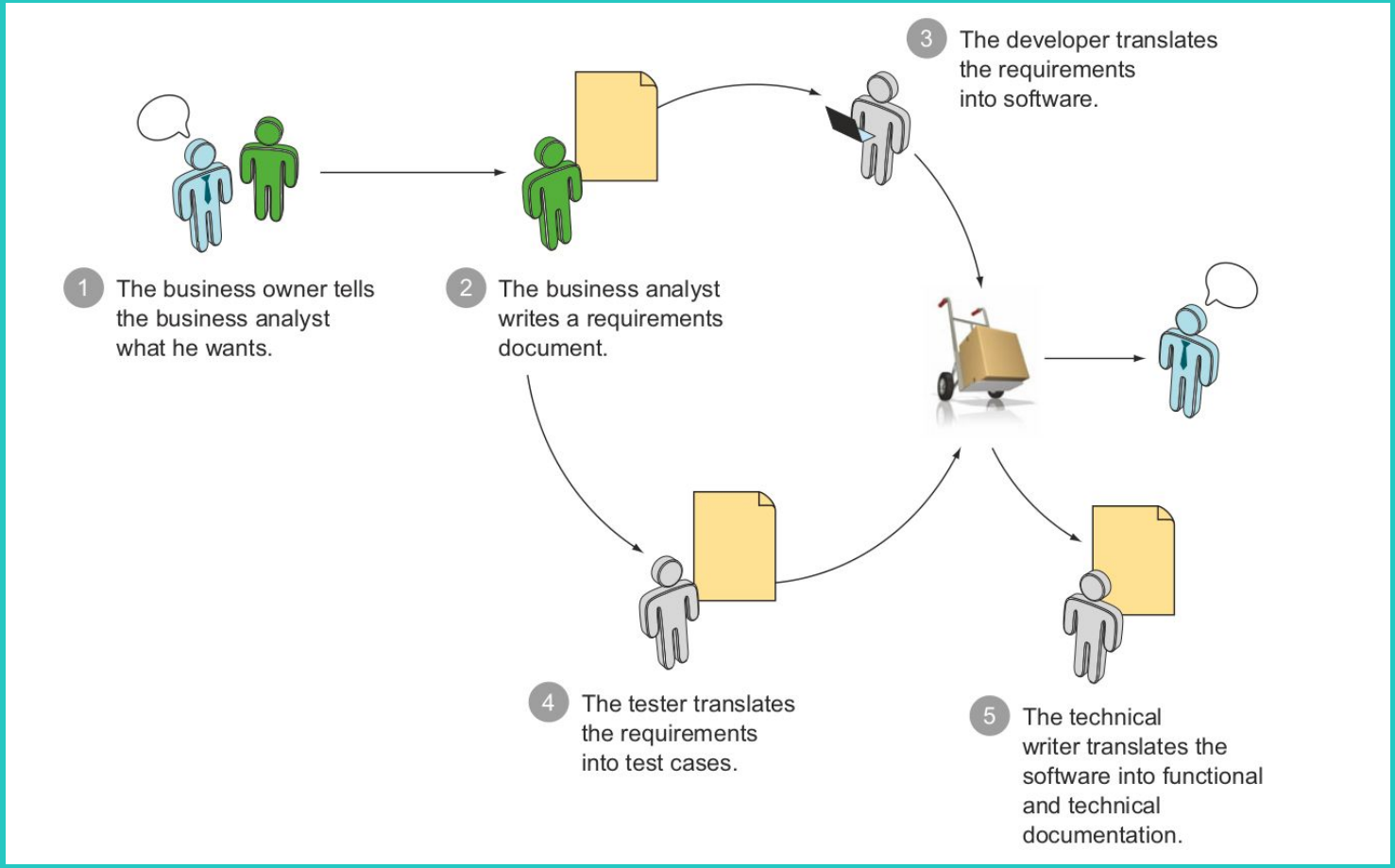
What is BDD

BDD stands for **Behavior-Driven Development**.It is an approach in software development that emphasizes the behavior of an application for business needs.

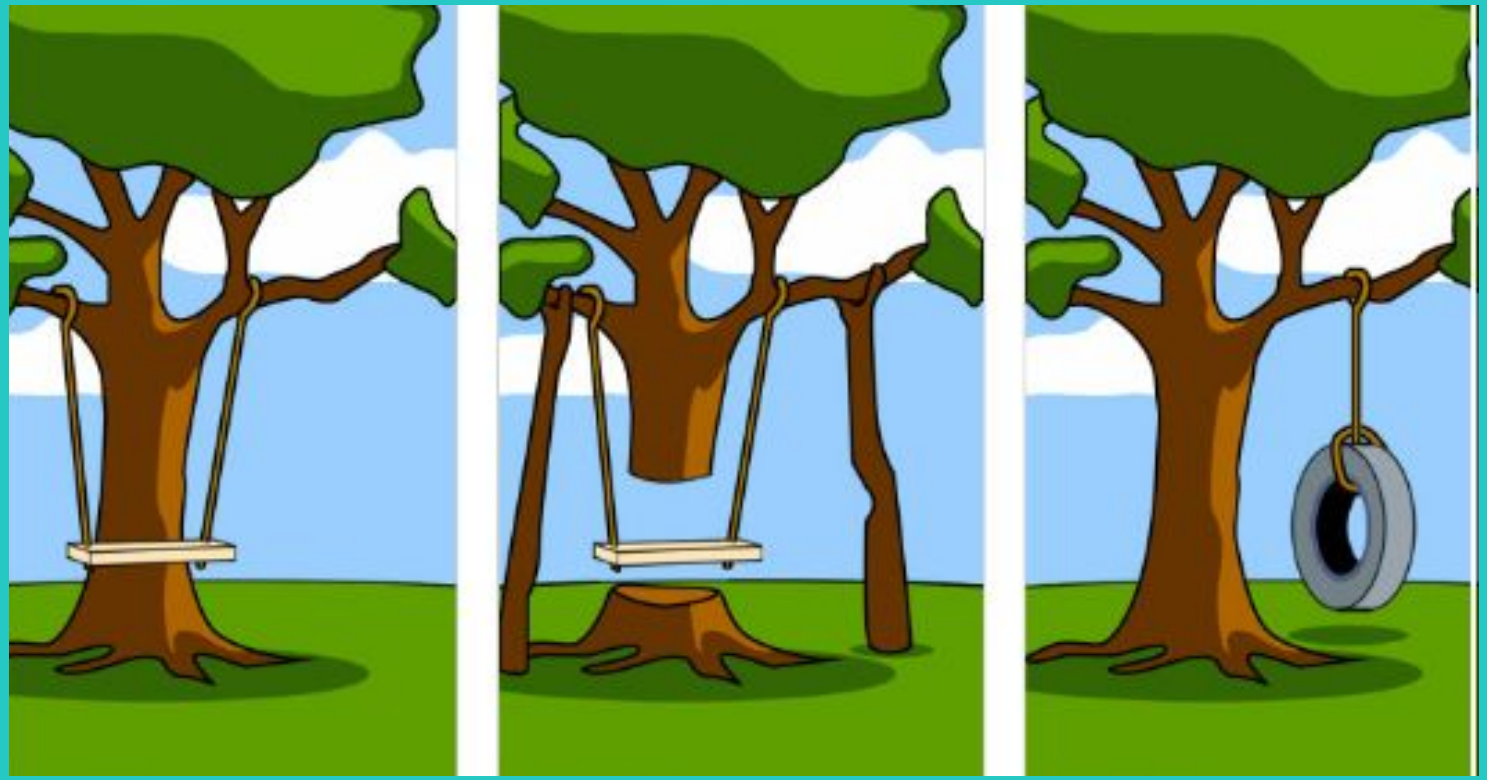
It was first introduced by Dan North in 2003 as a response to the shortcomings of traditional development practices like TDD



Test-Driven Development relies on a simple, three-phase cycle.



The traditional development process



What user need

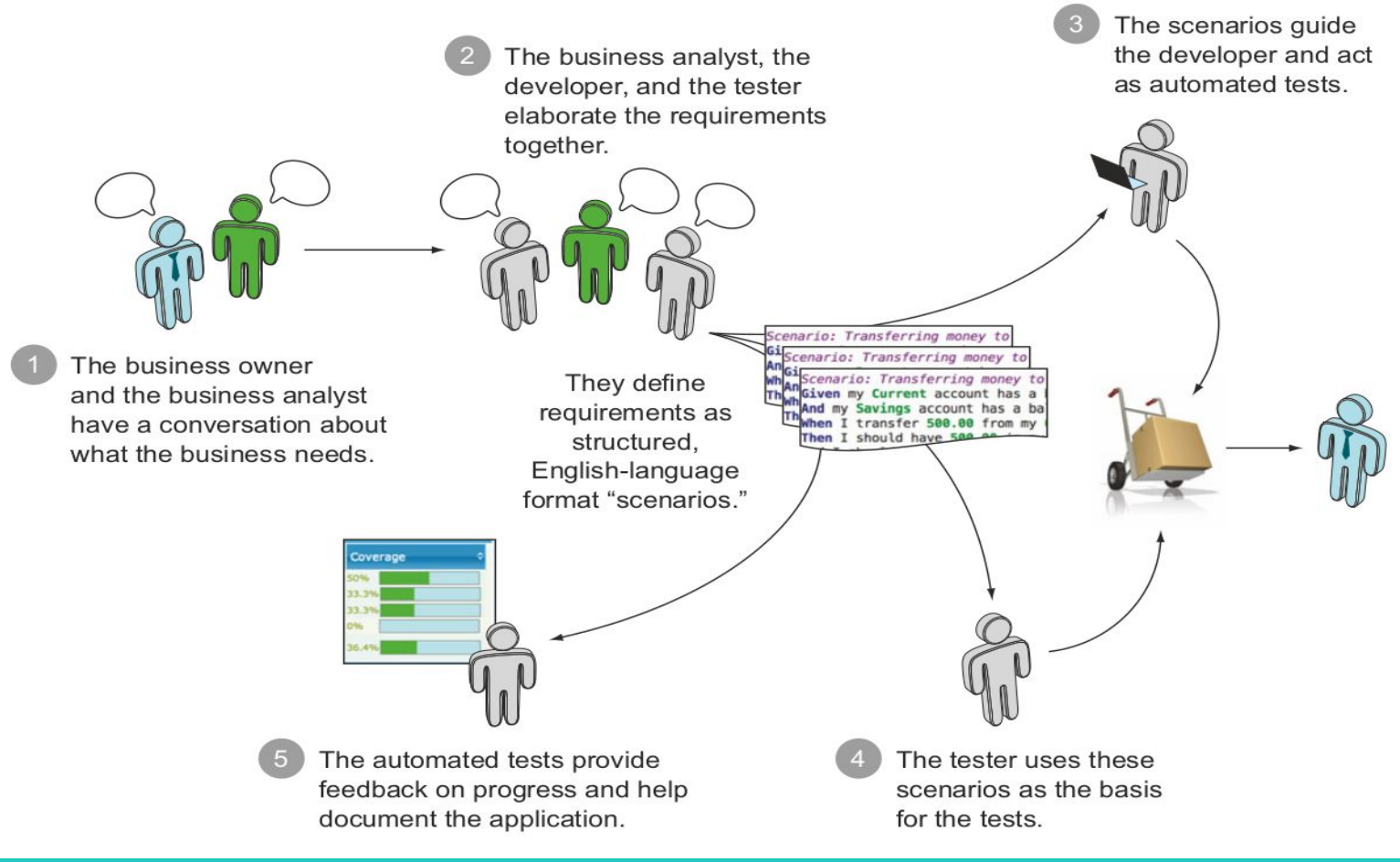
What developer understood

What tester understood

BDD is a collaborative approach to software development that emphasizes communication, collaboration, and shared understanding among all stakeholders.

It encourages the use of a common language, called the ubiquitous language, to describe the desired behavior of the software in a way that is understandable to both technical and non-technical team members.

It encourages developers to think in terms of the desired behavior of the software, rather than just implementing features.


1 The business owner and the business analyst have a conversation about what the business needs.
2 The business analyst, the developer, and the tester elaborate the requirements together.
They define requirements as structured, English-language format “scenarios.”
3 The scenarios guide the developer and act as automated tests.
Scenario: Transferring money to
Given my Current account has a
And my Savings account has a ba
When I transfer 500.00 from my
Then I should have
Coverage
50%
33.3%
33.3%
0%
36.4%
4 The tester uses these scenarios as the basis for the tests.
5 The automated tests provide feedback on progress and help document the application.

Key Features of BDD

1 Collaboration and Communication: BDD emphasizes collaboration and communication among all stakeholders involved in the software development process. It encourages active participation from business analysts, developers, testers, and other team members to ensure a shared understanding of the desired behavior of the software.

2 Ubiquitous Language: BDD promotes the use of a common language, often referred to as the "ubiquitous language," that is shared and understood by both technical and non-technical team members. This language helps to bridge the gap between business requirements and technical implementation, fostering clearer communication and reducing misunderstandings.

cont..

3 Specification by Example: BDD encourages the use of concrete examples and scenarios to describe the desired behavior of the software. These examples serve as living documentation that can be understood by both technical and non-technical stakeholders. By focusing on specific examples, BDD helps to clarify requirements and provides a common reference for discussion and validation.

4 Test Automation: BDD promotes the automation of tests based on the specified behavior. The scenarios and examples defined in BDD are often written in a format that can be executed as tests. By automating these tests, BDD ensures that the software's behavior is verified continuously, allowing for faster feedback and early detection of issues.

Cucumber

Cucumber is an open-source software tool used for Behavior-Driven Development (BDD). It provides a framework for writing and executing automated tests in a human-readable format.

Cucumber uses a plain-text syntax called **Gherkin** to express the behavior of the system in terms of scenarios and steps.

Gherkin

Gherkin is a structured language that is easy to understand by both technical and non-technical stakeholders. It allows for the creation of feature files that describe the functionality of the software in a business-readable format.

Gherkin uses a set of special **keywords** to give structure and meaning to executable specifications.

Each keyword is translated to many spoken languages.

Keywords

Feature : The purpose of the `Feature` keyword is to provide a high-level description of a software feature, and to group related scenarios.

Scenario: Describes a specific test scenario, typically written as "Scenario: <scenario-name>". It represents a particular use case or interaction with the system.

Given: Specifies the preconditions or initial state of the system for a scenario, written as "Given <precondition>". It sets up the necessary context before the action takes place.

When: Represents the specific action or event that is being performed in the scenario, written as "When <action>". It captures the user's interaction or system event.

Then: Defines the expected outcome or behavior that should result from the action, written as "Then <expected-outcome>". It describes the expected state or response after the action is performed.

And, But: These keywords are used to add additional steps within a scenario. "And" is used when the step adds to the previous step's context, while "But" is used to contrast or contradict the previous step.

Scenario Outline: Allows the definition of a scenario template that can be reused with different inputs or data sets. It uses placeholders (e.g., "<placeholder>") to represent variables that are filled in with concrete values in the examples table.

Examples: Provides a tabular structure to define multiple sets of inputs and expected outcomes for a scenario outline. Each row represents a different combination of inputs and expected outcomes.

```
Feature: Is it friday

  Feature Description
  Scenario: Is it friday
  Given today is Friday
  When I ask some whether it is friday yet
  Then I should be told Friday
```

Step Definition

Step definitions connect Gherkin steps to programming code. A step definition carries out the action that should be performed by the step. So step definitions hard-wire the specification to the implementation.

Advantage of BDD

- Improved Collaboration
- Greater Clarity
- Reduced Costs
- Enhanced Test Coverage
- Faster Feedback

Disadvantage of BDD

- BDD requires high business engagement and collaboration
- BDD doesn’t work well in a silo
- Add extra complexity
- Time and Effort for developing feature files,writing scenarios

Conclusion

Overall ,Behaviour-Driven Development provides a structural framework for effective collaboration,communication and Quality assurance,resulting in software tha better aligned with user needs and business objective.

End!