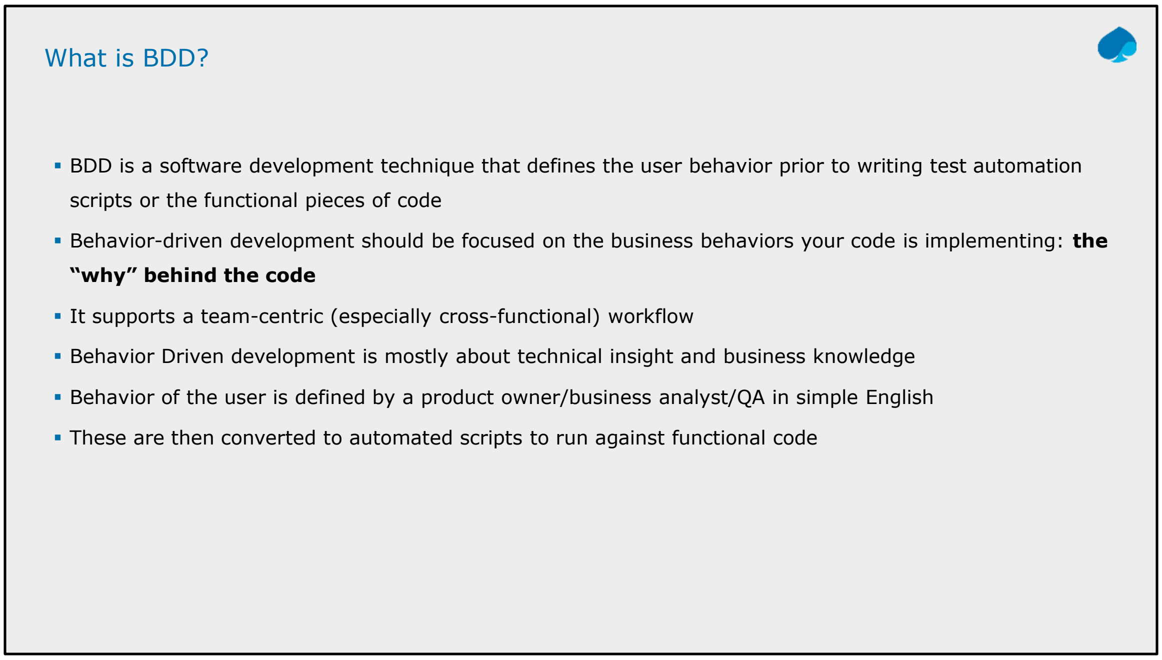
BDD

Lesson 01 – Introduction to BDD

Lesson Objectives

To understand the following topics:

- What is BDD?
- TDD Vs BDD
- Implementing BDD approach
- BDD Tools

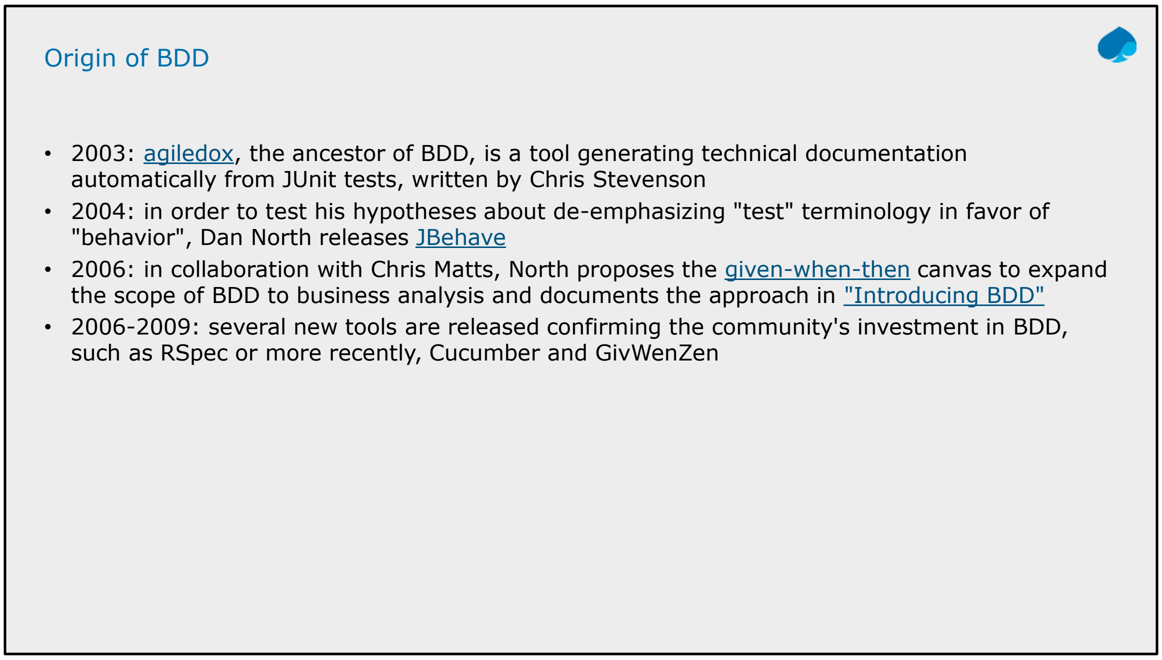


What is BDD?

- BDD is a software development technique that defines the user behavior prior to writing test automation scripts or the functional pieces of code
- Behavior-driven development should be focused on the business behaviors your code is implementing: **the "why" behind the code**
- It supports a team-centric (especially cross-functional) workflow
- Behavior Driven development is mostly about technical insight and business knowledge
- Behavior of the user is defined by a product owner/business analyst/QA in simple English
- These are then converted to automated scripts to run against functional code

This is a development method which has evolved from the Test-driven development process.
In most of the cases, this is achieved with the use of domain specific language.
Domain specific language uses natural English language constructs to define the outcomes from the said behavior.

Why BDD?
BDD addresses the issues like,
Where to start the process?
What to test and what not to test?
What to call the tests?
How to understand why a test fails?

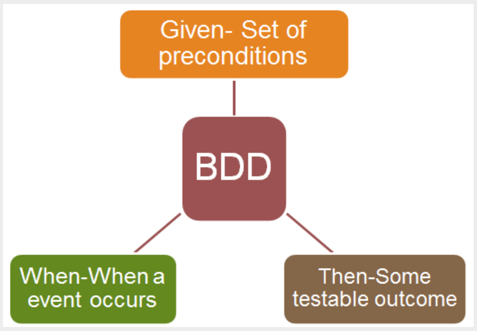


Origin of BDD

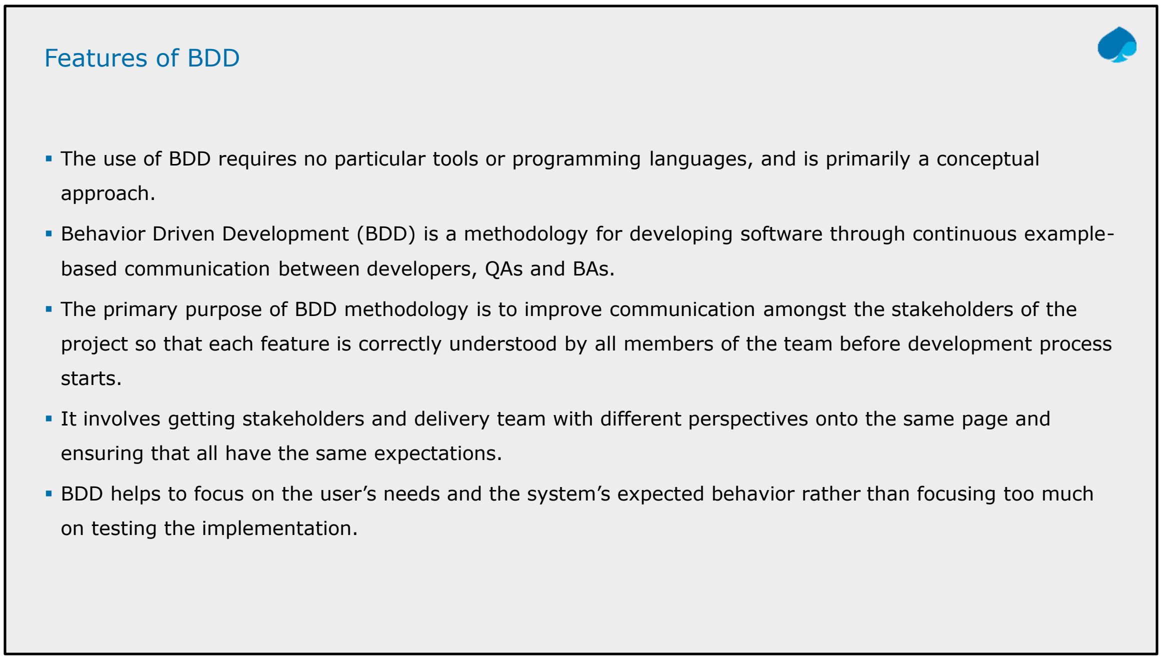
- 2003: agiledox, the ancestor of BDD, is a tool generating technical documentation automatically from JUnit tests, written by Chris Stevenson
- 2004: in order to test his hypotheses about de-emphasizing "test" terminology in favor of "behavior", Dan North releases JBehave
- 2006: in collaboration with Chris Matts, North proposes the given-when-then canvas to expand the scope of BDD to business analysis and documents the approach in "Introducing BDD"
- 2006-2009: several new tools are released confirming the community's investment in BDD, such as RSpec or more recently, Cucumber and GivWenZen

Reference :
https://www.agilealliance.org/glossary/bdd/#q=~(filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'bdd))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

BDD Framework



Most software projects involve teams of several people working collaboratively together, so high-quality communication is critical to their success. As you probably know, good communication isn't just about eloquently describing your ideas to others; you also need to solicit feedback to ensure you've been understood correctly. This is why agile software teams have learned to work in small increments, using the software that's built incrementally as the feedback that says to the stakeholders "Is this what you mean?"

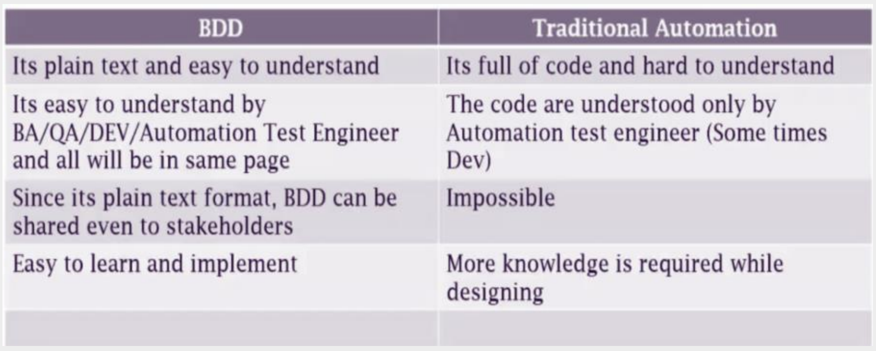


Features of BDD

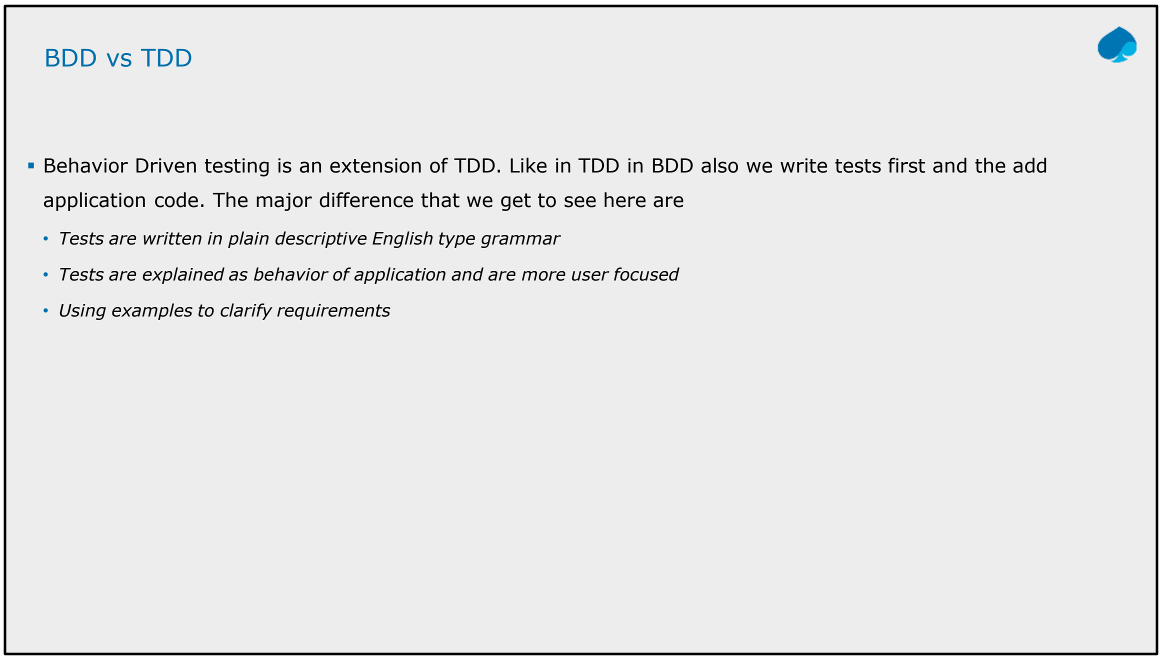
- The use of BDD requires no particular tools or programming languages, and is primarily a conceptual approach.
- Behavior Driven Development (BDD) is a methodology for developing software through continuous example-based communication between developers, QAs and BAs.
- The primary purpose of BDD methodology is to improve communication amongst the stakeholders of the project so that each feature is correctly understood by all members of the team before development process starts.
- It involves getting stakeholders and delivery team with different perspectives onto the same page and ensuring that all have the same expectations.
- BDD helps to focus on the user's needs and the system's expected behavior rather than focusing too much on testing the implementation.

In BDD, Examples are called Scenarios. Scenarios are written in a special format called Gherkin. The scenarios explain how a given feature should behave in different situations with different input parameters. They are called **"Executable Specifications"** because Gherkin is structural and it serves both specification and input into automated tests.

BDD vs Traditional Automation



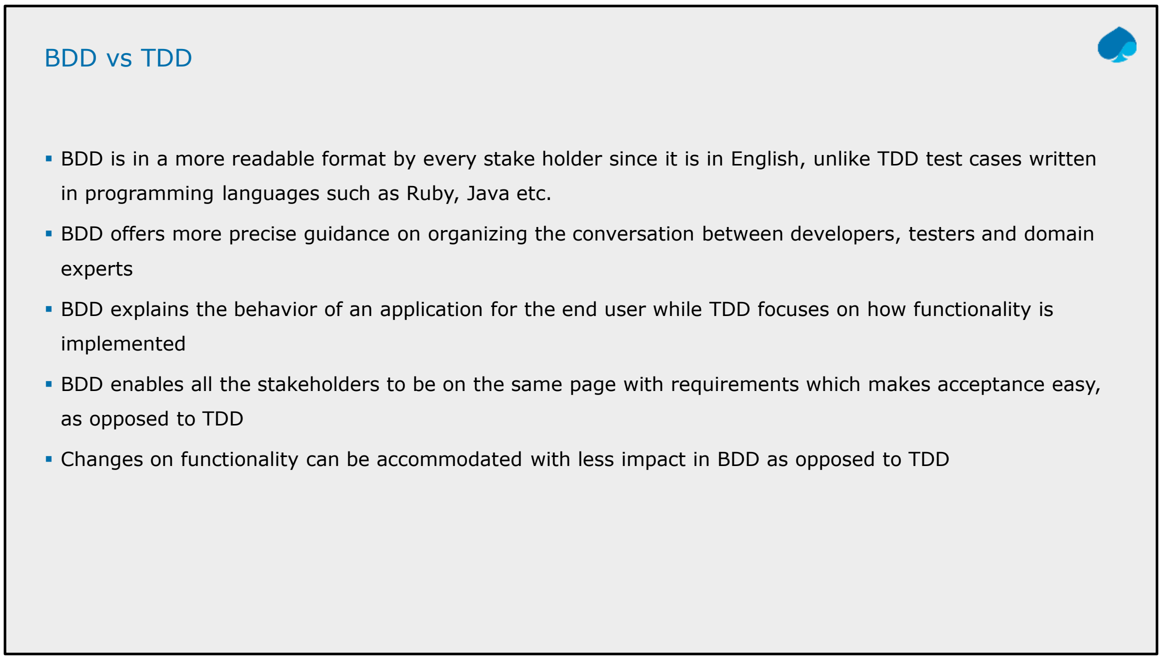
BDD	Traditional Automation
Its plain text and easy to understand	Its full of code and hard to understand
Its easy to understand by BA/QA/DEV/Automation Test Engineer and all will be in same page	The code are understood only by Automation test engineer (Some times Dev)
Since its plain text format, BDD can be shared even to stakeholders	Impossible
Easy to learn and implement	More knowledge is required while designing



BDD vs TDD

- Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. The major difference that we get to see here are
  - *Tests are written in plain descriptive English type grammar*
  - *Tests are explained as behavior of application and are more user focused*
  - *Using examples to clarify requirements*

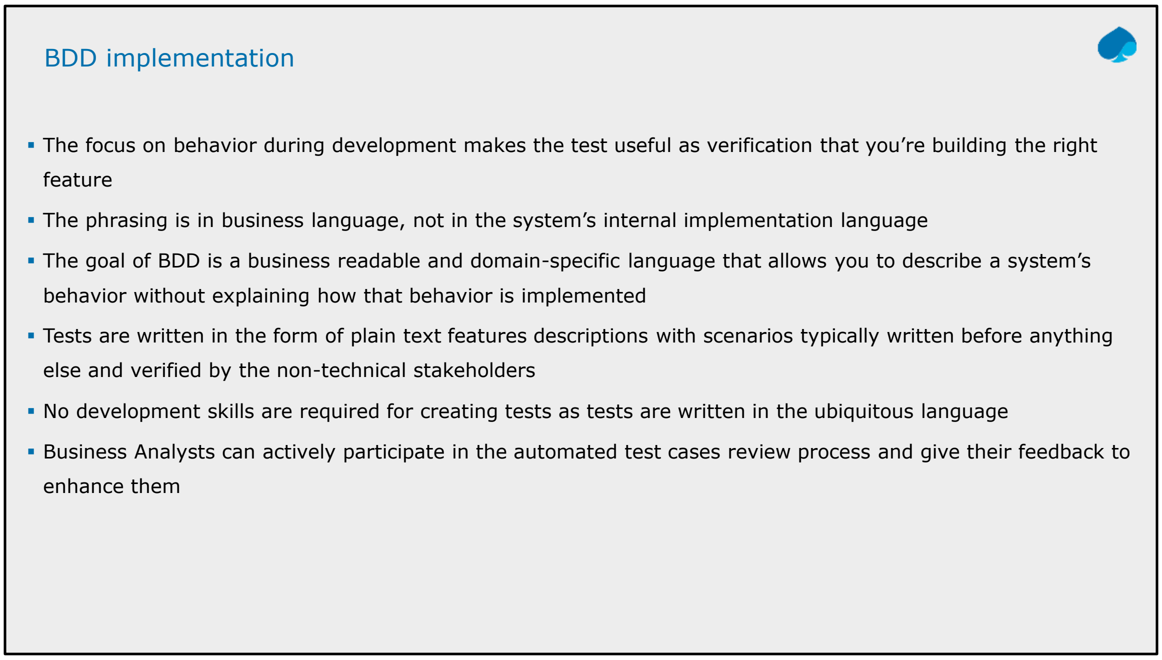
The behavior of the application is the central idea in BDD; it focuses on the customer and pushes developers and testers to walk in the customer’s shoes. If actions do not affect the end-user, BDD might not represent such a scenario very well, in which case TDD better serves the purpose.



BDD vs TDD

- BDD is in a more readable format by every stake holder since it is in English, unlike TDD test cases written in programming languages such as Ruby, Java etc.
- BDD offers more precise guidance on organizing the conversation between developers, testers and domain experts
- BDD explains the behavior of an application for the end user while TDD focuses on how functionality is implemented
- BDD enables all the stakeholders to be on the same page with requirements which makes acceptance easy, as opposed to TDD
- Changes on functionality can be accommodated with less impact in BDD as opposed to TDD

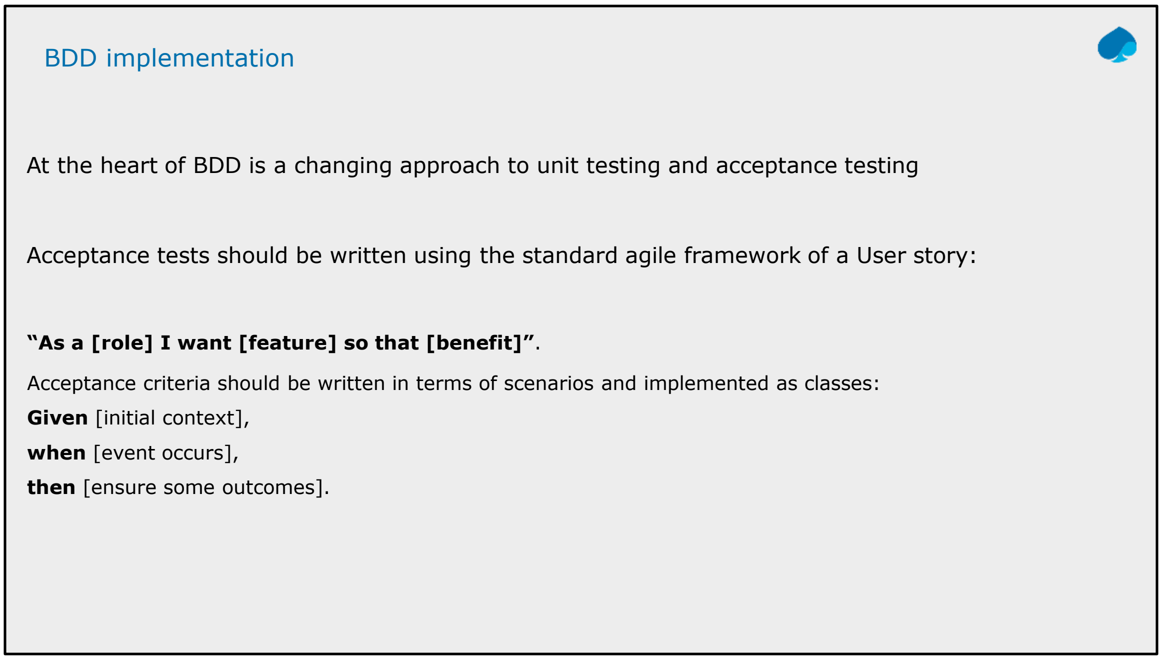
For systems that are driven by actions of the end user such as an ecommerce website or a HR system, BDD acts as a good medium to capture all the user actions.
For systems that have third party API calls, cron jobs, data exports/imports, etc., TDD might be a better solution.



BDD implementation

- The focus on behavior during development makes the test useful as verification that you're building the right feature
- The phrasing is in business language, not in the system's internal implementation language
- The goal of BDD is a business readable and domain-specific language that allows you to describe a system's behavior without explaining how that behavior is implemented
- Tests are written in the form of plain text features descriptions with scenarios typically written before anything else and verified by the non-technical stakeholders
- No development skills are required for creating tests as tests are written in the ubiquitous language
- Business Analysts can actively participate in the automated test cases review process and give their feedback to enhance them

BDD helps to focus on the user's needs and the system's expected behavior rather than focusing too much on testing the implementation. In some cases, the tests written by the manual tester/client/ business analysts are not good enough for automation. So, some rework may be required to make them suitable for automation.



BDD implementation

At the heart of BDD is a changing approach to unit testing and acceptance testing

Acceptance tests should be written using the standard agile framework of a User story:

"As a [role] I want [feature] so that [benefit]".

Acceptance criteria should be written in terms of scenarios and implemented as classes:

Given [initial context],

when [event occurs],

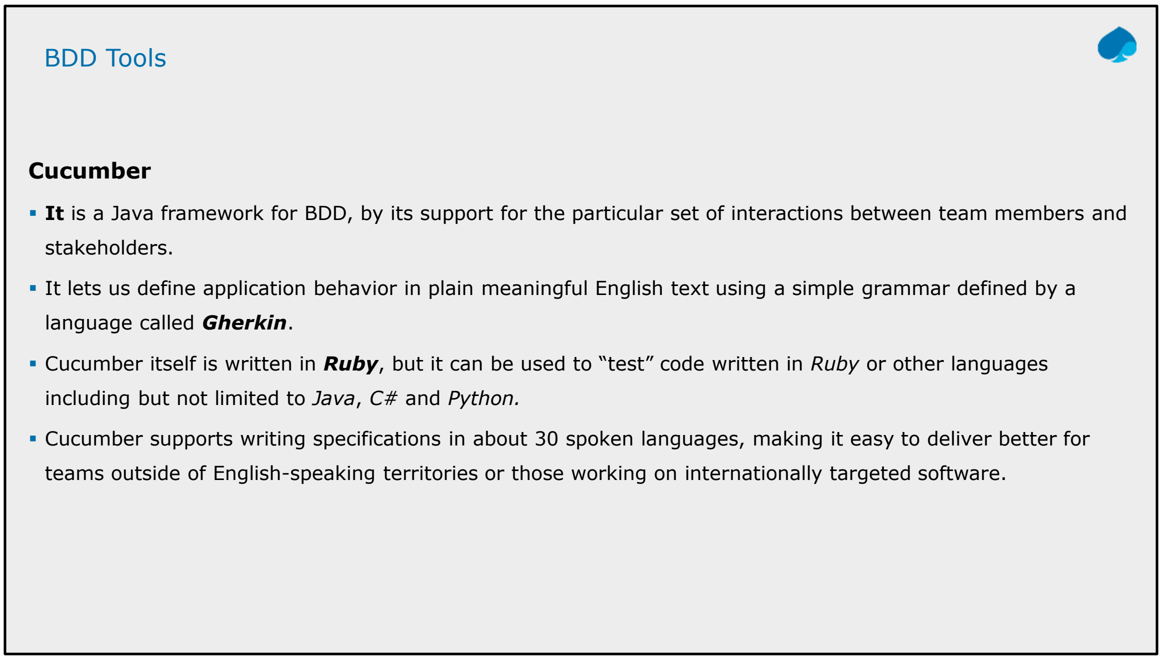
then [ensure some outcomes].

Behavior Driven Development is a worthwhile practice for any software studio to at least evaluate and try, and it has the potential to bring big benefits to your development and software testing programs.

How to start with the Behaviour Driven Development?

Firstly, there's a period of training for your team involved in your requirements documentation — and with no matter, whether you're working Agile or Waterfall or something else, this approach will work.

Secondly, you have to explain the changes to your developers and QA teams. Educate them on what the changes are going to be, and how to read and interpret the new method of documentation. Plan how you will incorporate this into your code and test case documentation.

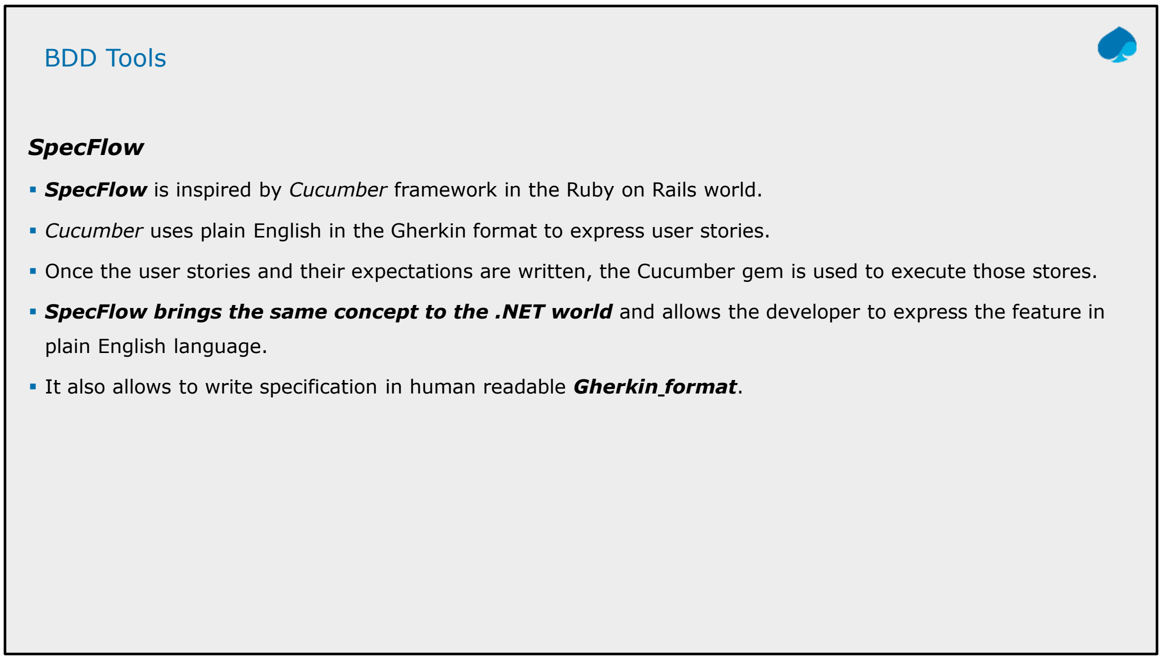


BDD Tools

Cucumber

- **It** is a Java framework for BDD, by its support for the particular set of interactions between team members and stakeholders.
- It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called ***Gherkin***.
- Cucumber itself is written in ***Ruby***, but it can be used to "test" code written in *Ruby* or other languages including but not limited to *Java*, *C#* and *Python.*
- Cucumber supports writing specifications in about 30 spoken languages, making it easy to deliver better for teams outside of English-speaking territories or those working on internationally targeted software.

Features include:
• Integration with all the most popular testing libraries;
• Specifying the behavior looking at the system from the outside;
• Defining executable specifications in different ways like lists, prose and tabular data;
• The plain text files can be stored in any version control system;
• Collaboration and coming up with a good and clear set of Acceptance Criteria.



BDD Tools

SpecFlow

- ***SpecFlow*** is inspired by *Cucumber* framework in the Ruby on Rails world.
- *Cucumber* uses plain English in the Gherkin format to express user stories.
- Once the user stories and their expectations are written, the Cucumber gem is used to execute those stores.
- ***SpecFlow brings the same concept to the .NET world*** and allows the developer to express the feature in plain English language.
- It also allows to write specification in human readable ***Gherkin_format***.

Summary

In this lesson, you have learnt:

- Different definitions of BDD
- How BDD differs from TDD
- Importance of BDD in Agile
- Tools used for implementing BDD

Q.4.getTitle()
Q.5.True
Q.6. sendKeys()

Review Question

Question 1: BDD is a replacement for TDD.

- True
- False

Question 2: BDD uses the syntax of programming language that is used for developing an application.

- True
- False