**SpecFlow** is a testing framework which supports Behaviour Driven Development (BDD). It lets us define application behaviour in plain meaningful English text using a simple grammar defined by a language called Gherkin.

*SpecFlow Introduction*

* [*Test Driven Development (TDD)*](http://toolsqa.com/cucumber/test-driven-development-tdd/)
* [*SpecFlow & Behavior Driven Development*](http://toolsqa.com/cucumber/cucumber-behavior-driven-development-bdd/)
* [*Gherkin – Business Driven Development*](http://toolsqa.com/cucumber/gherkin-business-driven-development-bdd-language/)

## **Test Driven Development**

***TDD*** is an iterative development process. Each iteration starts with a set of tests written for a new piece of functionality. These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.

Any failures in the test run are marked and more Application code is written/re-factored to make these tests pass. Once application code is added/re-factored the tests are run again. This cycle keeps on happening till all the tests pass. Once all the tests pass we can be sure that all the features for which tests were written have been developed.

## **Drawbacks of TDD**

1. Developer can consider it as a waste of time
2. The test can be targeted on verification of classes and methods and not on what the code really should do
3. Test become part of the maintenance overhead of a project
4. Rewrite the test when requirements change

## **Behavior Driven Development**

**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

This difference brings in the need to have a language which can define, in an understandable format.

**Features of BDD**

1. Shifting from thinking in “tests” to thinking in “behavior”
2. Collaboration between Business stakeholders, Business Analysts, QA Team and developers
3. Ubiquitous language, it is easy to describe
4. Driven by Business Value
5. Extends Test Driven Development (TDD) by utilizing natural language that non technical stakeholders can understand
6. BDD frameworks such as Cucumber or JBehave are an enabler, acting a “bridge” between Business & Technical Language

BDD is popular and can be utilised for ***Unit level*** test cases and for***UI level*** test cases. Tools like ***RSpec***(for Ruby) or in .NET something like ***MSpec***or ***SpecUnit*** is popular for Unit Testing following BDD approach.  Alternatively, you can write BDD-style specifications about***UI interactions***. Assuming you’re building a web application, you’ll probably use a browser automation library like ***WatiR/WatiN or Selenium***, and script it either using one of the frameworks I just mentioned, or a given/when/then tool such as ***Cucumber (for Ruby)*** or***SpecFlow (for .NET)***.

## **BDD Tools Cucumber & SpecFlow**

### *****What is Cucumber?*****

***Cucumber*** is a testing framework which supports ***Behavior Driven Development (BDD).***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.

### *****What is 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*](http://toolsqa.com/cucumber/gherkin-business-driven-development-bdd-language/).

## **Why BDD Framework?**

Let’s assume there is a requirement from a client for an E-Commerce website to increase the sales of the product with implementing some new features on the website. The only challenge of the development team is to convert the client idea in to something that actually delivers the benefits to client.

The original idea is awesome. But the only challenge here is that the person who is developing the idea is not the same person who has this idea. If the person who has the idea happens to be a talented software developer, then we might be in luck: the idea could be turned into working software without ever needing to be explained to anyone else. Now the idea needs to be communicated and has to travel from Business Owners (Client) to the development teams or many other people.

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?”