The Agile software methodology is nowadays becoming popular among software development teams in the fast-moving market. However, this popular trend also imposes testing teams to manage and maintain their test cases and test scripts following changing requirements. Thus, one should choose an appropriate testing method right from the beginning to implement any Agile software project smoothly. In this article, we will be covering the features of Cucumber BDD along with the below topics:

Cucumber and its outstanding features
It helps in improving communication.
It is an automated acceptance testing tool.
All the testers can take part in an automation test using Cucumber BDD.

Cucumber and its outstanding features

As yet, there have been many successful Agile software projects. Thanks to the Behavior-Driven Development (*BDD*) method using the Cucumber tool. So, what is Cucumber?

A cucumber is a tool used to run automated acceptance tests created in a BDD format. One of its extraordinary features of the tool is the ability to carry out plain-text functional descriptions (*written in the language called **Gherkin***) as automated tests.

Let's take a look at the below example:

```
Feature: Update password

 Scenario: Admin user can update the user password

Given I am in the HR system with an Admin account
When I update password of another user
Then I receive a message for updating password successfully
And user password is updated to the new password
```

This amazing feature of ***Behavior-Driven Development (BDD)*** approach with the advantages as below:

Writing BDD tests in an omnipresent language, a language whose structure is built around the domain model and widely used by all team members comprising of developers, testers, BAs, and customers.
Connecting technical with non-technical members of a software team.
Allowing direct interaction with the developer's code, but we write BDD tests in a language that can also be made out by business stakeholders.

Last but not least, acceptance tests can execute automatically, while business stakeholders manually perform it.

1. Cucumber helps improve communication

Cucumber assists in improving communication between technical and non-technical members of the same project. Let's have a look at the requirement below and its automation tests:

```
As an Admin User,
I would like to change the password of other user's accounts.
Feature: Update password
 Scenario: Admin user can update the user password
   Given I am in the HR system with an Admin account
   When I update password of another user
   Then I receive a message for updating password successfully
   And user's password is updated to the new password
```

With TestNG, the above test scenario can implement as below:

```
@test
public void testAdminUserCanUpdateUserAccountPassword() {
 // create users
 User userAdmin = new User(UserRole.ADMIN, username, password);
 User user = new User(UserRole.VIEWER, user_username, user_password);

 // use Admin user to update another user password
  String message = userAdmin.updatePassword(user, user_new_password);


// verify password changed
  Assert.assertEquals(message, "Password changed successfully");
  Assert.assertEquals(user.getPassword(), user_new_password);
}
```

We can write the same test case using Cucumber:

```
Feature: Update password
 Scenario: Admin user can update the user password
   Given I am in the HR system with an Admin account
   When I update password of another user
   Then I receive a message for updating password successfully
   And user's password is updated to the new password
```

Both automation test scripts above execute well to complete the test automatically. But do all testers of your team make out these tests? Is there a possibility of other business analysts and other stakeholders use these tests again at the acceptance testing (*AT*) stage?

The automation test with TestNG may be tough for most manual testers and BAs to catch up. Moreover, it is not possible to use this test again for AT. As a result, based on these flaws mentioned before, this can not be considered as a suitable method.

In contrast, we develop/ create the automation test using Cucumber in a business domain language or in natural language, which all members of the software project team can easily make out. Communication is vital for any development team, especially in the Agile team. There are usually lot of continuous chats, discussions, or even arguments happening among developers and testers to figure out what the correct behavior of a feature is. By using Cucumber, the developers can develop the same feature specification now for testing by testers. It is a powerful tool because it can help lower the risk of misunderstanding as well as the communication breakdown.

2. Cucumber is an Automated Acceptance Testing Tool

The acceptance test is generally carried out by *BAs/*customers to make sure that the development team has built specific features. A common activity in this testing stage is verifying the system against the original requirements with specific, real data from production. Cucumber testing not only follows the requirements as its test scenarios but also helps *BAs* or Product Manager to adjust test data quickly. Here is a demonstration with a little adjustment:

```
As an Admin User,
I would like to change the password of other user's accounts.
Feature: Update password
 Scenario: Admin user can update the user password
   Given I am in the HR system with an Admin account
   When I update password of another user
   Then I receive a message for updating password successfully
   And user's password is updated to the new password
```

We write the automation test in the Cucumber framework:

```
Scenario Outline: Verify Updating user password feature
 Given I am in the HR system with "<account type>" account
 And there is another user with "<old password>" password
 When I update password of the user to "<new_password>"
 Then I got the message "<message>"
 And the user password should be "<final_password>"
```

```
Examples:
|account type  |old password |new password |message                |final password |
|Admin         |$Test123 |@Test123     |Password changed.. |@Test123       |
|Viewer        |$Test123 |@Test123     |Invalid right access.. |$Test123       |
```

3. All testers can take part in automation test with Cucumber BDD

In addition to improving communication within the members of the same testing team, Cucumber also helps leverage tester's skills efficiently. The expertise gap always exists in every organization. In other words, some testers have great technical expertise in programming utilizing automated testing, while others are performing manual testing with limited programming skills in the same team. Thanks to Cucumber, all testers, irrespective of their skill levels, can participate in the process of performing automation tests.

Let's take a look at the above example:

> *Any tester who knows the business logic and workflow can write feature files, add more scenarios, and test datasets.*
> *Moreover, any tester who has a basic knowledge of programming and know-how to create objects, access properties, call methods, can generate step definitions.*
> *Any tester with a higher programming skill level can take part in the process of making a framework, define data source connection, and so on.*

There are still a few important issues when implementing Cucumber:

Cucumber helps run test scenarios mentioned in a plain text file using business domain knowledge. Thus, the usage of languages and the perception of the one who creates the test might directly influence the test scenarios, leading to the risk of misunderstanding. We should present the Test scenarios clearly, and their implementation should perform accurately for each step. For example, when you want to verify the Search feature on Google, the test should be:

```
Scenario: performing a search on google
Given I am on "www.google.com" site
When I search for "Cucumber and BDD"
Then ...
```

We incorporate the below steps to have the following test:

```
Scenario: performing a search on google
When I search for "Cucumber and BDD"
```

```
Then ...
```

The stages of the Cucumber tool perform in an ordinary language. One can use them again in various test scenarios. It helps reduce the effort to create tests. However, maintaining the test to be both readable and reusable is a big challenge. If the test is written at a very high level for any stakeholders to make out, we can reuse a few steps (*bold*). Both the above scripts are right; however, the second one is not apparent. Because it does too much more than expected: opening Google's website and searching with the specified text. Say, you want to extend the test to search more texts, you may repeat the above step. Consequently, we open the Google site twice. If you do not follow the requirement stringently, the Cucumber testing tool will cause misunderstanding sooner or later and be so difficult to maintain when being extended.

```
Feature: Update password

 Scenario: Admin user can update the user password
   Given I am in the HR system with an Admin account
   When I update password of another user
   Then I receive a message for updating password successfully
   And user's password is updated to the new password

 Scenario: Viewer user cannot update the user password
   Given I am in the HR system with a Viewer account
    When I update password of another user
    Then I receive a message for not able to update the user password
    And user's password remains the same
```

Contrarily, if the test is generic and if we can reuse to verify updating the user's Last Name, non-technical stakeholders will have difficulty in catching up. Additionally, they can not perform Acceptance Tests.

```
Scenario: Admin user can update user password:
 Given I am in the "$System.HR_Page" with "admin@test.com" username
and "$Test123" password
 And there is another user in "$System.HR_Page" with "user@test.com"
username and "$Test123" password
 When I update "$UserTemplate.Password" of "user@test.com" user to"@Test123"
 And I save the response message as "response message"
 Then "$response message" should be "Password changed successfully"
 And the  "user@test.com" user's "$UserTemplate.Password" should be"@Test123"
```

During the testing process, you have to adjust test scenarios regularly. We do it until they reach an entirely acceptable balance where all members can understand and reuse.

```
Scenario: Verify Updating user password feature
 Given I am in the HR system with "Admin" account
 And there is another user with "$Test123" password
 When I update password of the user to "@Test123"
 Then I got the message "Password changed successfully."
 And the user password should be "@Test123"
```

Or with some more test data:

```
Scenario Outline: Verify Updating user password feature
 Given I am in the HR system with "<account type>" account
 And there is another user with "<old password>" password
 When I update password of the user to "<new_password>"
 Then I got the message "<message>"
 And the user password should be "<final_password>"

Examples:
|account type |old password |new password |message           |final password |
|Admin        |$Test123 |@Test123     |Password changed.. |@Test123  |
|Viewer       |$Test123 |@Test123     |Invalid right access.. |$Test123  |
```

Important notes for the testing team who wants to get started with Cucumber

> *Consider automation tests as essential as a real project. The code should follow coding practice, convention, etc.*
> *One should also consider an appropriate editor tool. This editor should help debug and edit feature files in standard text format. Aptana (free editor), RubyMine (commercial editor), and Katalon Studio are suitable options that completely support BDD-based Cucumber.*
> *Last but not least, make feature files an actual "**communication**" layer where you can store received test data and format test data. It does not contain Domain business logic.*

Cucumber is one of the most powerful tools. It offers us the real communication layer on top of a robust testing framework. The tool can help run automation tests on a wide-ranging testing needs from the backend to the frontend. Moreover, Cucumber creates deep connections among members of the testing team, which we hardly find in other testing frameworks. With many years of automation testing experience, I recommend that Cucumber for Web UI and Web service testing should implement in a way that it helps in successful Agile software project operation.