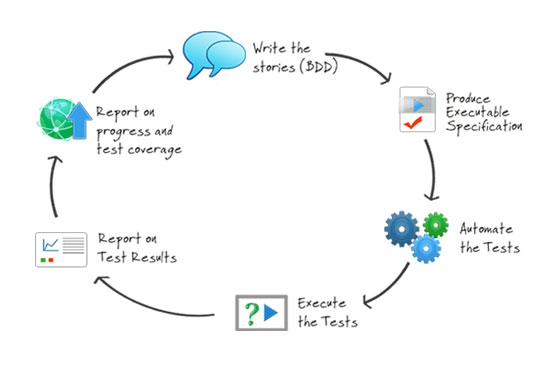


# Overview

**Cucumber / Selenium**

Business and IT don't always understand each other. Too often misunderstandings between business and IT result in the costly failure of IT projects. With this in mind Cucumber was developed as a tool to support human collaboration between business and IT.

Cucumber uses executable specifications to encourage close collaboration. This helps teams to keep the business goal in mind at all times. With Cucumber you can merge specification and test documentation into one cohesive whole, allowing your team to maintain one single source of truth. Because these executable specifications are automatically tested by Cucumber, your single source of truth is always up-to-date.



Cucumber supports testers when designing test cases. To automate these test cases several languages can be used. Cucumber also works well with Browser Automation tools such as Selenium Webdriver.

**Selenium**

Selenium automates browsers and is used for automating web applications for testing purposes. Selenium offers testers and developers full access to the properties of objects and the underlying tests, via a scripting environment and integrated debugging options.

Selenium consists of many parts. If you want to create robust, browser-based regression automation suites and tests then Selenium Webdriver is most appropriate. With Selenium Webdriver you can also scale and distribute scripts across many environments.

## Strengths

**Supports BDD**

Those familiar with Behavior Driven Development (BDD) recognize in Cucumber an excellent open source tool that supports this practice.

**All in one place**

With Cucumber / Selenium you can automate at the UI level. Automation at the unit or API level can also be implemented using Cucumber. This means all tests, regardless of the level at which they are implemented, can be implemented in one tool.

**Maintainable test scripts**

Many teams seem to prefer UI level automation. This despite the huge cost of maintaining UI level tests compared with the cost of maintaining API or unit tests. To lessen the maintenance of UI testing, when designing UI level functional tests, you can try describing the test and the automation at three levels: business rule, UI workflow, technical implementation.

When using Cucumber combined with Selenium you can implement these three levels for better maintenance.

**Early start**

Executable specifications can and should be written before the functionality is implemented. By starting early teams get most return on investment from their test automation.

**Supported by a large community**

Cucumber and Selenium are both open source tools with a large community, online resources and mailing lists.

# Tooling

**Cucumber**

Cucumber supports over a dozen different software platforms. Every Cucumber implementation provides the same overall functionality, but they also have their own installation procedure and platform-specific functionality. See <https://cucumber.io/docs> for all Cucumber implementations and framework implementations.

Also, IDE’s such as Intellij offer several plugins for Cucumber support.

**Selenium**

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

## Automation process

**Write a feature file**

Test automation in Cucumber starts with writing a feature file. A feature normally consists of several (test)scenario’s and each scenario consists of several steps.

Feature: Refund item

Scenario: Jeff returns a faulty microwave

Given Jeff has bought a microwave for $100

And he has a receipt

When he returns the microwave

Then Jeff should be refunded $100

Above example shows a feature “Refund item” with one scenario “Jeff returns a faulty microwave”. The scenario consists of four steps each starting with a key word (Given, And, When, Then).

##### Implementing the steps

Next the steps are implemented. Assuming we use Java to implement the steps, the Java code will look something like this.

public class MyStepdefs {

@Given("Jeff has bought a microwave for $(\d+)")

public void Jeff\_has\_bought\_a\_microwave\_for(int amount) {

// implementation can be plain java

// or selenium

driver.findElement(By.name("test")).sendKeys("This is an example\n");

driver.findElement(By.name("button")).click();// etc

}

}

Cucumber uses an annotation (highlighted) to match the step from the feature file with the function implementing the step in the Java class. The name of the class and the function can be as the developer sees fit. Selenium code can be used within the function to automate interaction with the browser.

**Running scenario’s**

There are several ways to run scenarios with Cucumber, for example the JUnit runner, a command line runner and several third party runners.

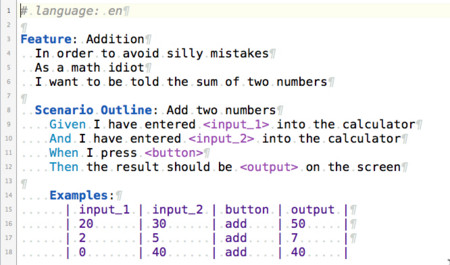
**Reporting test results**

Cucumber can report results in several different formats, using *formatter plugins*

## Features

##### Feature files using GherkiN

Cucumber executes your feature files. As shown in the example below, feature files in Gherkin are easy to read so they can be shared between IT and business. Data tables can be used to execute a scenario with different inputs.



##### organizing tests

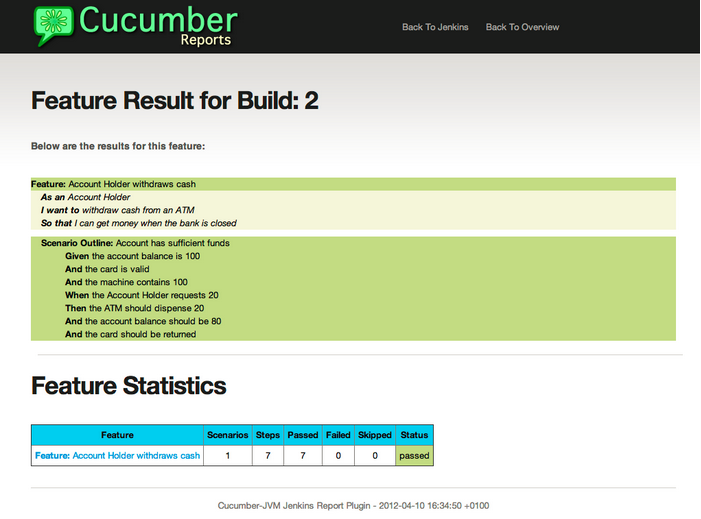
Feature files are placed in a directory structure and together form a feature tree.

Tags can be used to group features based on all kinds of categories. Cucumber can include or exclude tests with certain tags when running the tests.

## Reporting test results

Cucumber can report results in several formats, using formatter plug-ins.

Not supported option by Shared Services: The output from Cucumber can be used to present test results in Jenkins or Hudson depending of the preference of the project.



# HOW IS Cucumber / Selenium USED AT Capgemini?

## Tool deployment

Cucumber and Selenium are chosen as one of Capgemini’s test automation industrial tools. We support the Java implementation of Cucumber and Selenium Webdriver. We can help with creating Cucumber, Selenium projects in Eclipse and IntelliJ.

## Application in ATaaS (Automated Testing as a service)

In the context of industrialisation, Capgemini has developed a range of services to assist and support the projects in process and tools implementation.

In this context a team of experts assists projects using test automation.

The main services provided by the center of expertise are:

* Advise on the feasibility of automation.
* Support with installation.
* Coaching teams in the use of BDD.
* Support in writing the first feature files and step implementations. We help teams to start up. (For teams who want to have their feature files implemented by us, we offer the service Testing BOX. )