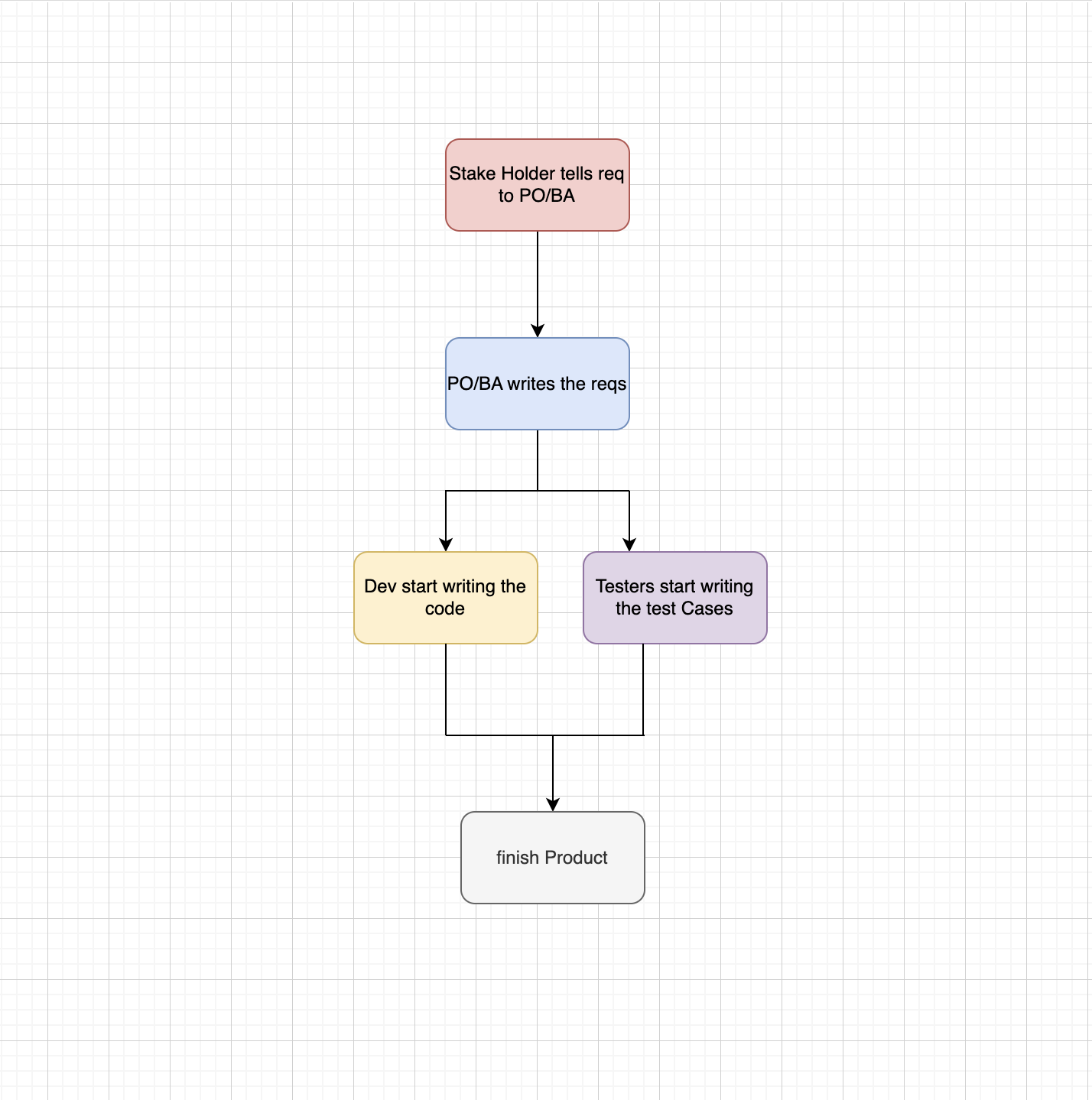
Cucumber

1. Cucumber is testing tool that support Behavior Driven Development (BDD) methodology.

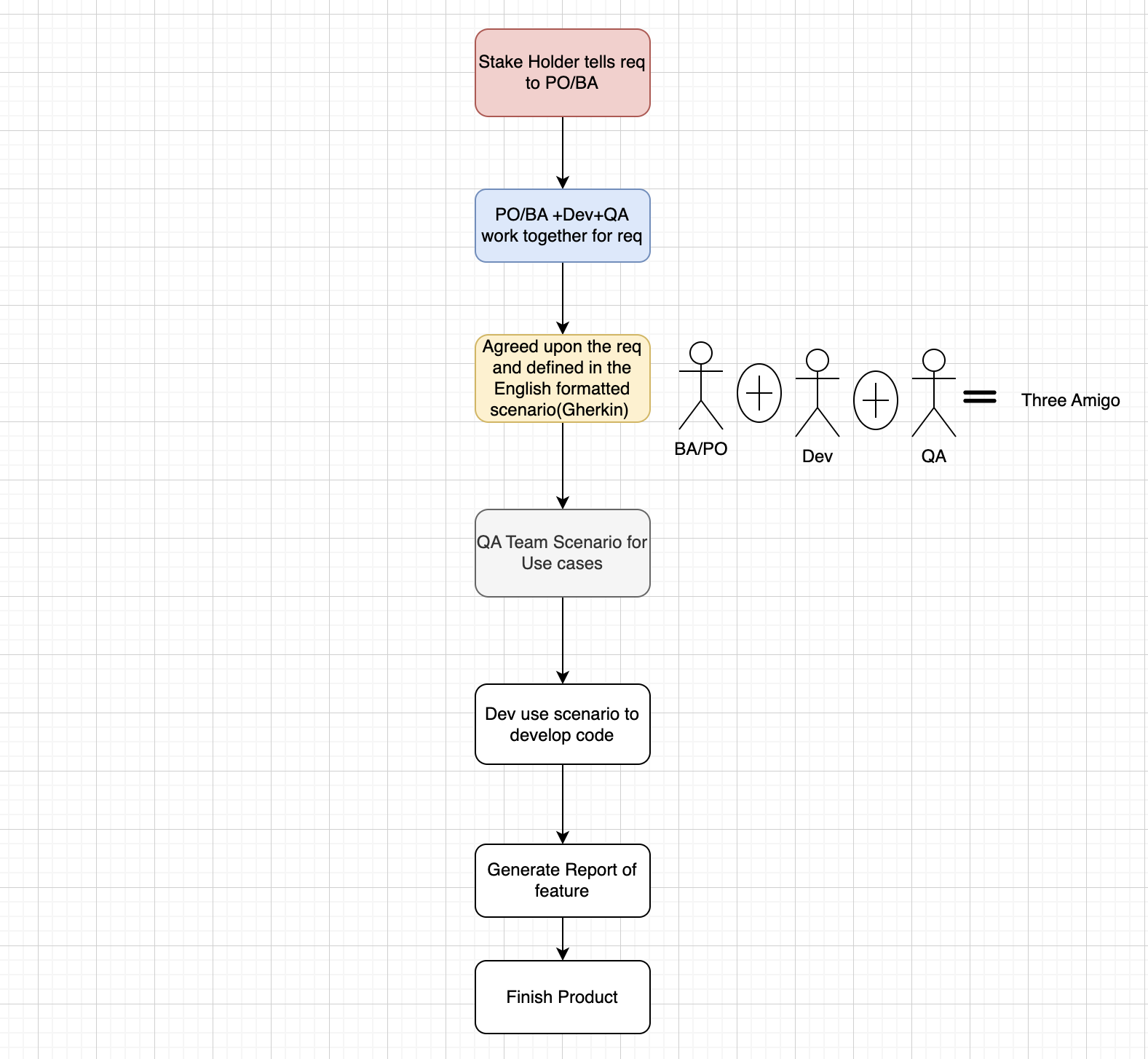
**Traditional Development Process**



**Limitation:-**

1. After getting the requirement from PO/BA, the developer and tester starts works
2. Let us suppose developer develop the code on the basis of their understanding but it didn’t meet the requirement.
3. Let us suppose tester didn’t get the requirement and test cases will not cover
4. Less communication while developing

**Behavior Development Process**

****

**Dependency required to Set up**

**<dependency>**

**<groupId>info.cukes</groupId>**

**<artifactId>cucumber-java</artifactId>**

**<version>1.0.2</version>**

**<scope>test</scope>**

**</dependency>**

**artifactId----cucumber-java. This artifact id is used to import the cucumber api imports**

import io.cucumber.junit.Cucumber;  
import io.cucumber.junit.CucumberOptions;  
  
@CucumberOptions(  
 features = "Feature"  
 ,glue={"stepDefinition"}  
  
)

**<dependency>**

**<groupId>info.cukes</groupId>**

**<artifactId>cucumber-junit</artifactId>**

**<version>1.0.2</version>**

**<scope>test</scope>**

**</dependency>**

import org.junit.runner.RunWith;  
  
@RunWith(Cucumber.class)

**<dependency>**

**<groupId>junit</groupId>**

**<artifactId>junit</artifactId>**

**<version>4.10</version>**

**<scope>test</scope>**

**</dependency>**

**Gherkin:-**

1. Gherkin is a language in which cucumber features files is written.
2. Gherkin is a plain English text language, which helps the tools- Cucumber to interpret and execute the test scripts

**Feature Files**

The file, in which Cucumber tests are written, is known as feature files. It is advisable that there should be a separate feature file, for each feature under test. The extension of the feature file needs to be “.feature”.

A simple feature file consists of the following keywords/parts −

* **Feature** − Name of the feature under test.
* **Description** (optional) − Describe about feature under test.
* **Scenario** − What is the test scenario.
* **Given** − Prerequisite before the test steps get executed.
* **When** − Specific condition which should match in order to execute the next step.
* **Then** − What should happen if the condition mentioned in WHEN is satisfied.

Scenario is a keyword in which we write the test scripts

**Scenario** − Verify Help Functionality.

**Given** user navigates to Facebook and enter the username and password

**When** the user clicks on Help, then the Help page opens.

**Scenario Outline**

**Scenario outline** basically replaces variable/keywords with the value from the table. Each row in the table is considered to be a scenario.

**Scenario** outline: Verify Help Functionality.

**Given** user navigates to Facebook and enter the “<username>” and “<password>”

**When** the user clicks on Help, then the Help page opens.

Examples:

| username | password |

|test. | test |

**The Values that comes in Examples keyword are called as dataTable**

**Cucumber Hooks:-**

Cucumber supports hooks, which are blocks of code that run before or after each scenario. You can define them anywhere in your project or step definition layers, using the methods @Before and @After. Cucumber Hooks allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.

**Cucumber Backgroud:-**

***Background in Cucumber*** is used to define a step or series of steps that are common to all the tests in the feature file. It allows you to add some context to the scenarios for a feature where it is defined. A Background is much like a scenario containing a number of steps. But it runs before each and every scenario were for a feature in which it is defined.

package org.automation.runner;  
  
import io.cucumber.junit.Cucumber;  
import io.cucumber.junit.CucumberOptions;  
import org.junit.runner.RunWith;  
  
@RunWith(Cucumber.class)  
@CucumberOptions(  
 features = "Feature"  
 ,glue={"stepDefinition"}  
  
)  
public class Runner {  
  
}

1. @RunWith(Cucumber.class)---.> Used to run the cucumber from runner file
2. features = "Feature" -----🡪 We need to give the feature directory path
3. glue={"stepDefinition"}---🡪 path of the step definition directory.