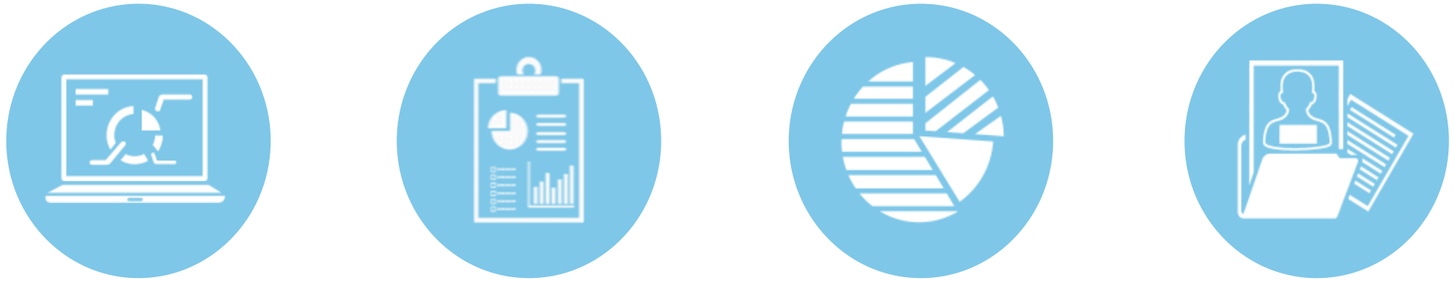
DevOps Certification Training

Lesson 04: Software and Automation Testing Frameworks



Learning Objectives

By the end of this lesson, you will be able to:

- Describe the benefits in traditional and agile approach of software testing
- Describe the levels and approaches of software testing
- Demonstrate test-driven development framework with JUnit
- Demonstrate behavior-driven development framework with cucumber

Software and Automation Testing Frameworks

Software Testing Overview

Software Testing

Software testing is an investigation conducted to share information about the quality of the product and the service under test to the organization.

Is it meeting the requirements that guided its design and development?

Is it responding correctly to the inputs?

Is it functioning within an acceptable time?

Is it working properly in its intended environments?

Is it achieving the results?

Source:Wikipedia

Traditional vs. Agile Testing

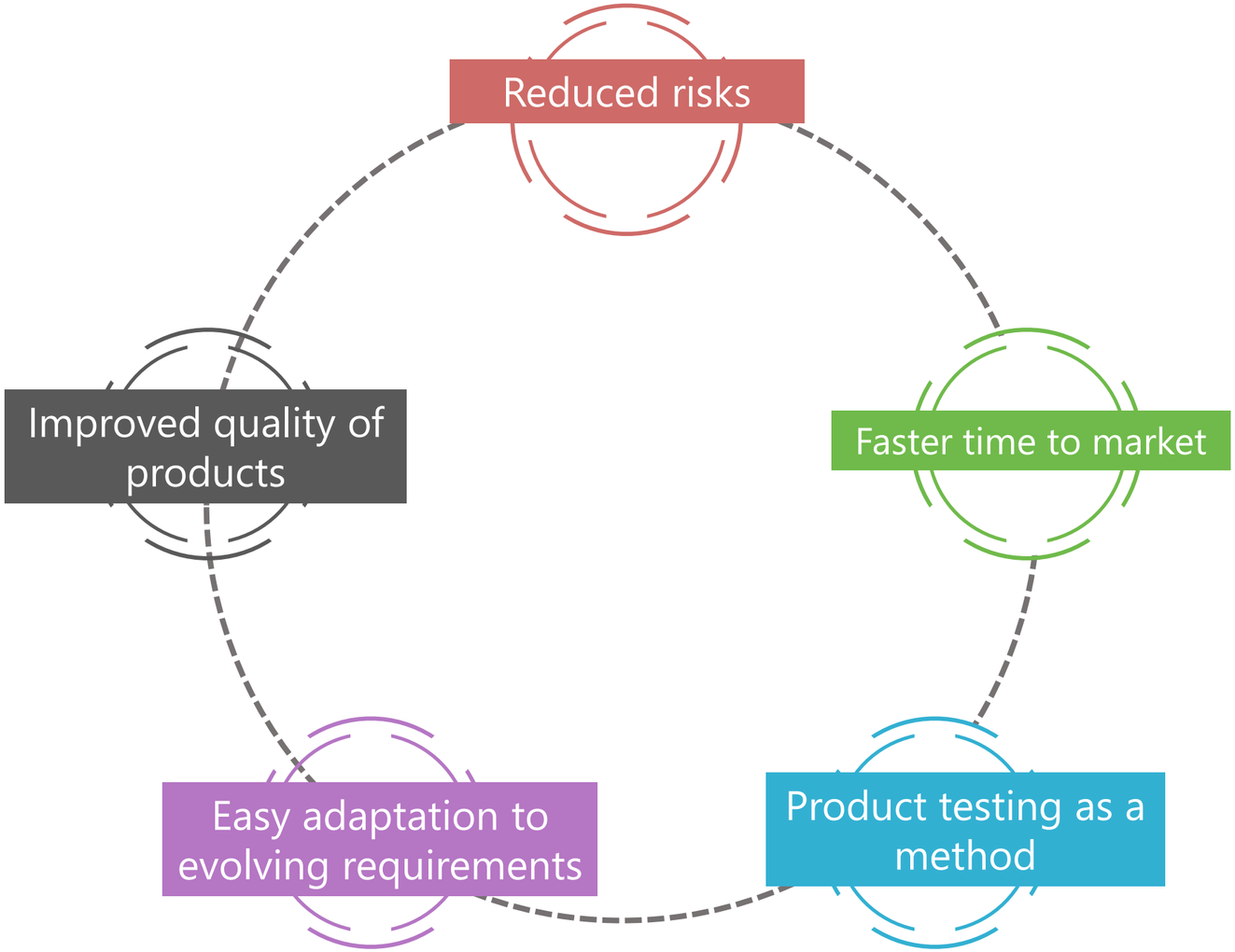
Traditional Testing

- Follows predictive model with a step-by-step phased approach
- Certifies the quality of product
- Produces the final system with all the required features
- Unit testing for each module followed by integration and system testing
- No feedback until test is completed
- Modifications are implemented in the next release

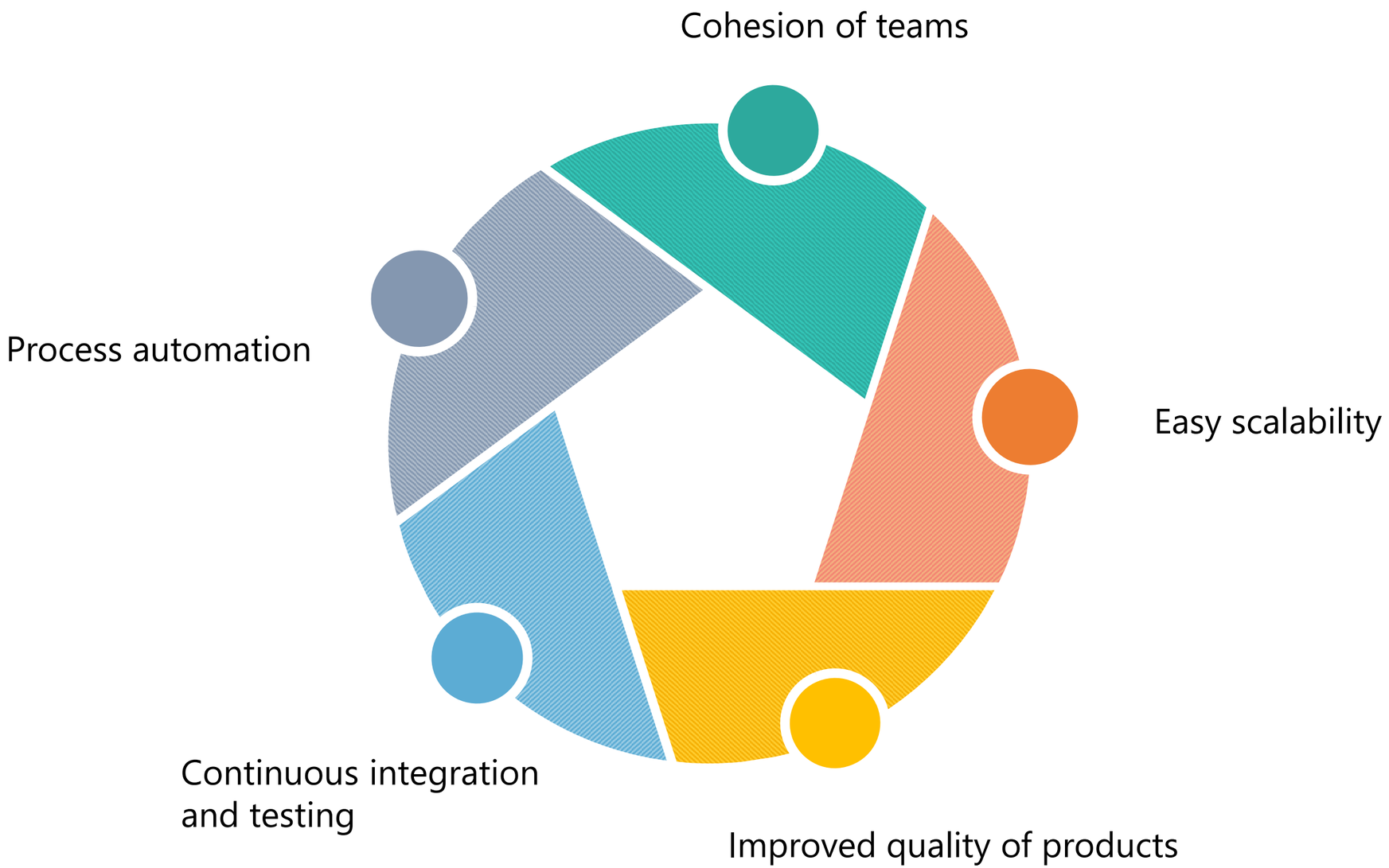
Agile Testing

- Follows adaptive model with iterative approach
- Certifies the quality and fast delivery of the product with minimal functionalities
- Produces working versions of the final system that have a subset of the features
- Continuous testing and regression across all iterations
- At the end of every sprint, short ongoing feedback cycles are produced
- Modifications are implemented in the next sprint of the test cycle

Agile Testing Advantages



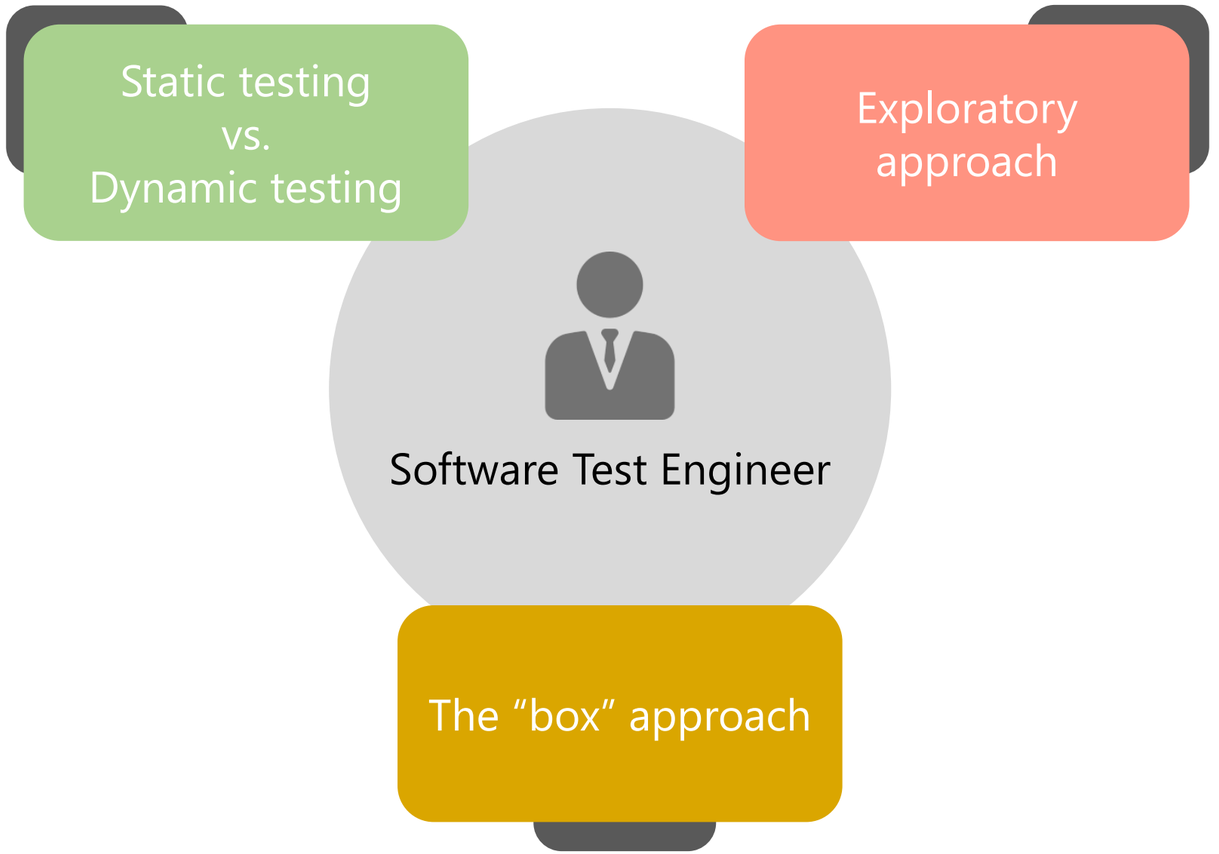
Advantages of Using Agile with DevOps Testing



Software and Automation Testing Frameworks

Testing Levels, Approaches, and Automation Tools

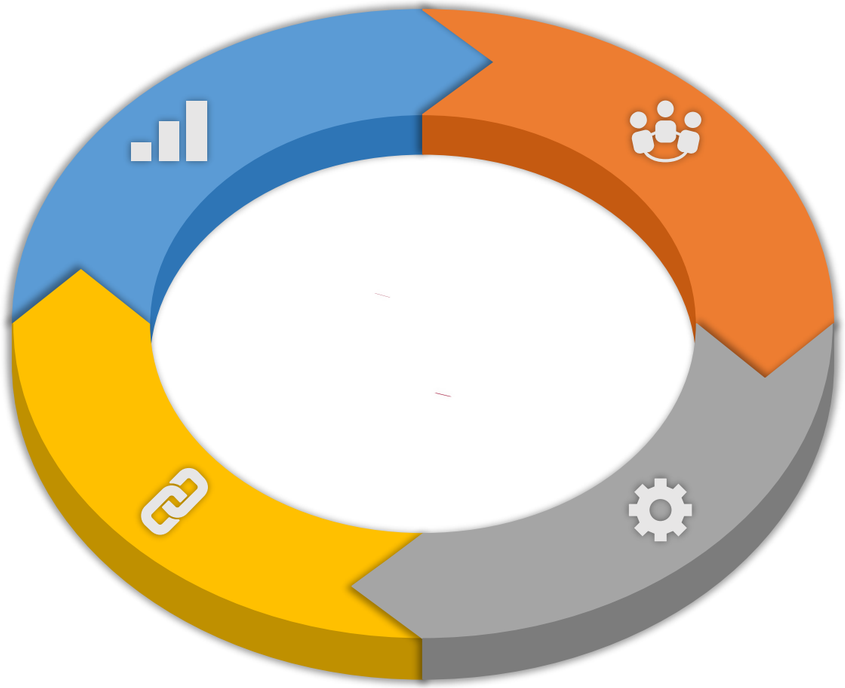
Software Testing Approaches



Software Testing Levels

Unit Testing

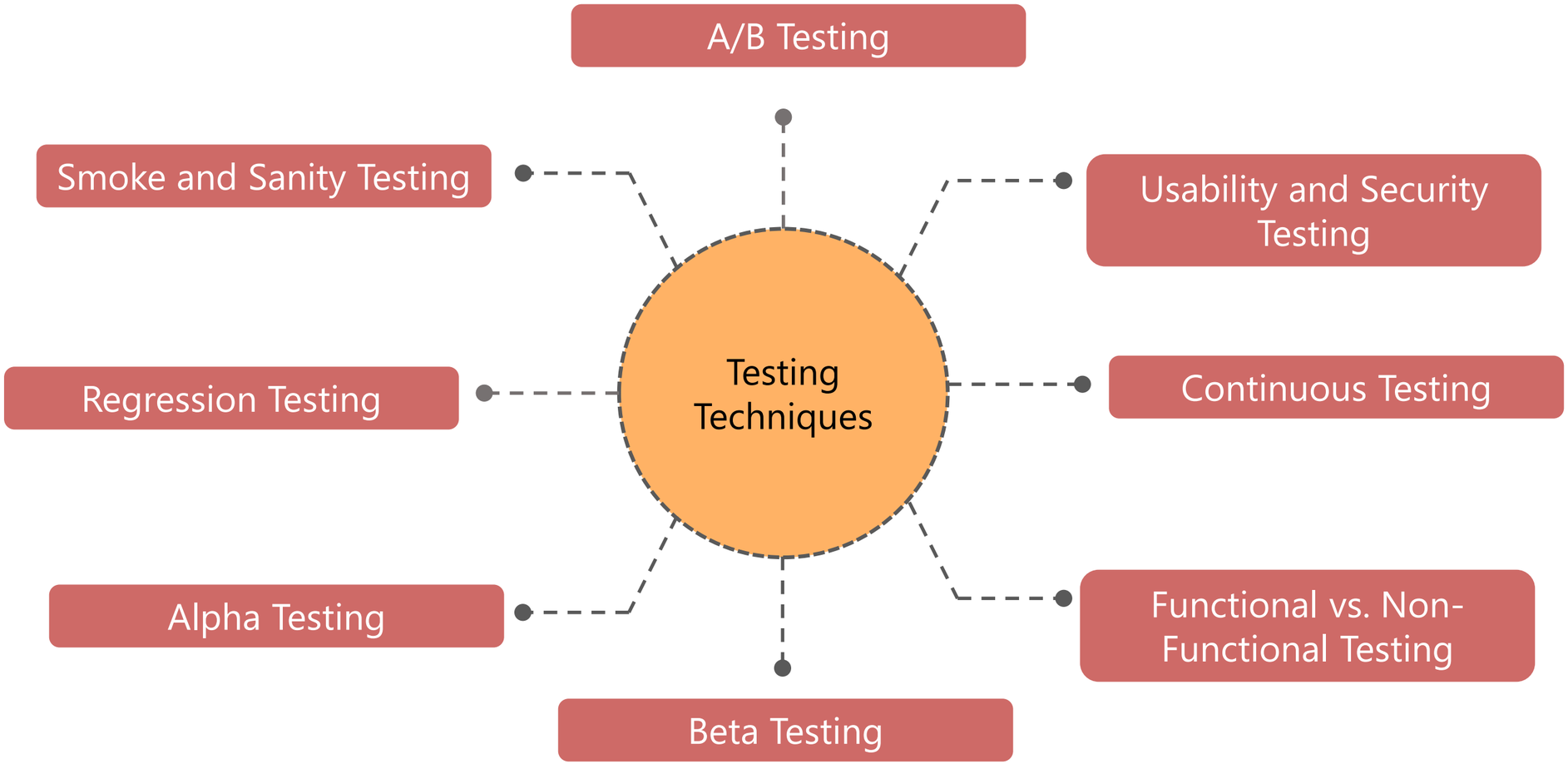
Operational Acceptance Testing



Integration Testing

System Testing

Popular Software Testing Techniques



Automation Testing Framework

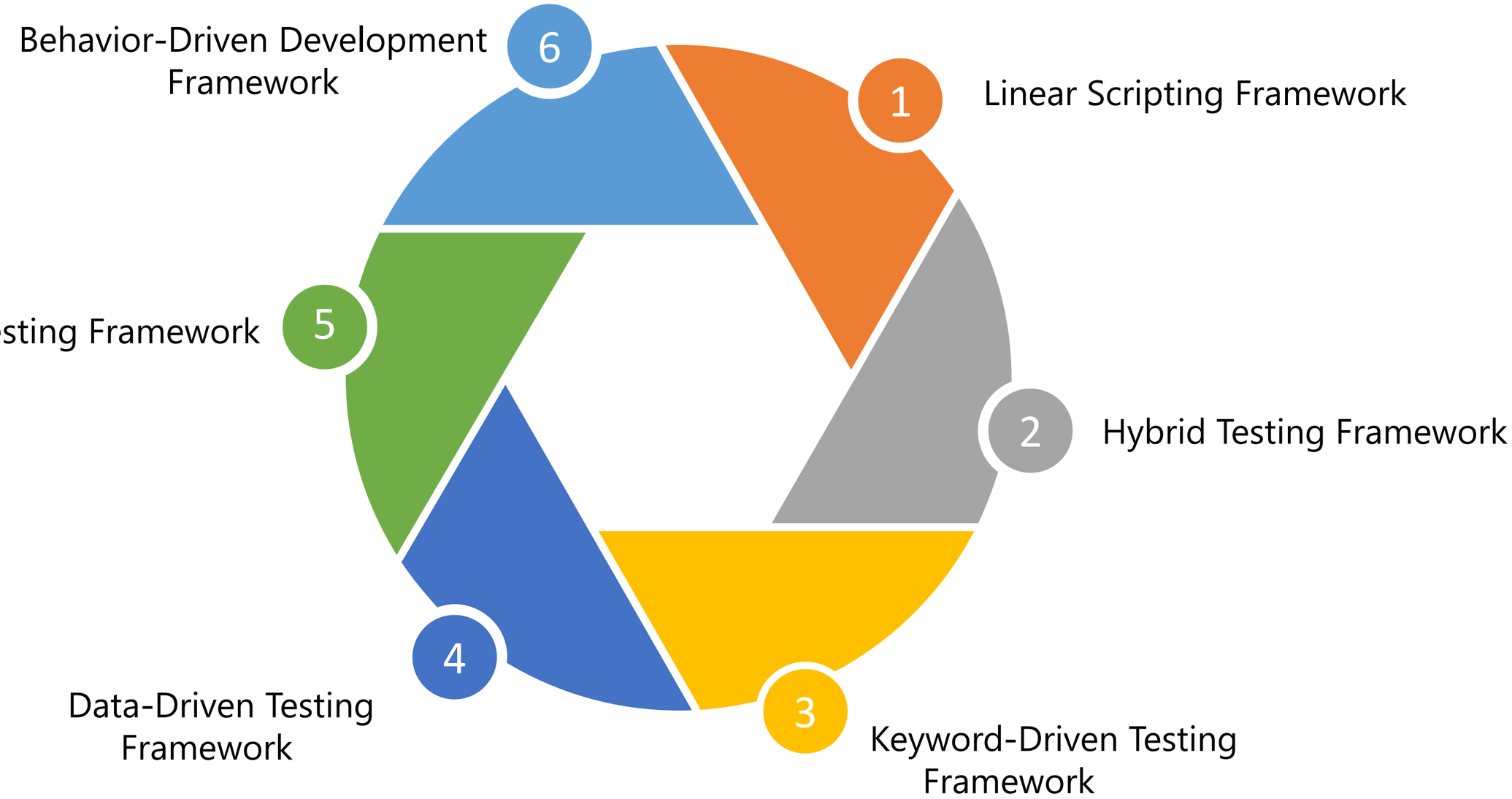
Test-driven development teams rely on automated testing. A testing framework is used to write test cases and a continuous integration software will run these test cases automatically. A well-developed testing framework or an integrated development environment (IDE) can be used to run the test scripts.

TestNG

Jenkins

JUnit 5

Automation Testing Framework (Contd.)



Popular Testing Tools

SoapUI is an open-source web service testing application for service-oriented architectures and RESTful APIs.

Robot Framework is a keyword-driven test automation framework that uses tabular test data syntax.

Popular Testing Tools (Contd.)

RSpec is a domain specific testing tool to test Ruby code. It is a behavior-driven development framework that is used in the production applications.

Eggplant Functional is a black-box GUI test automation tool which uses intelligent image recognition algorithms to monitor the display being tested. It is used for mobile testing, cross-platform testing, and performance testing.

Popular Testing Tools (Contd.)

Mocha is a JavaScript test framework for Node.js programs. Its features are browser support, asynchronous testing, test coverage reports, and use of any assertion library.

Jasmine is an open-source testing framework for JavaScript. It supports asynchronous testing, use of 'spies' for implementing test doubles, testing of front-end code through a front-end extension.

Software and Automation Testing Frameworks

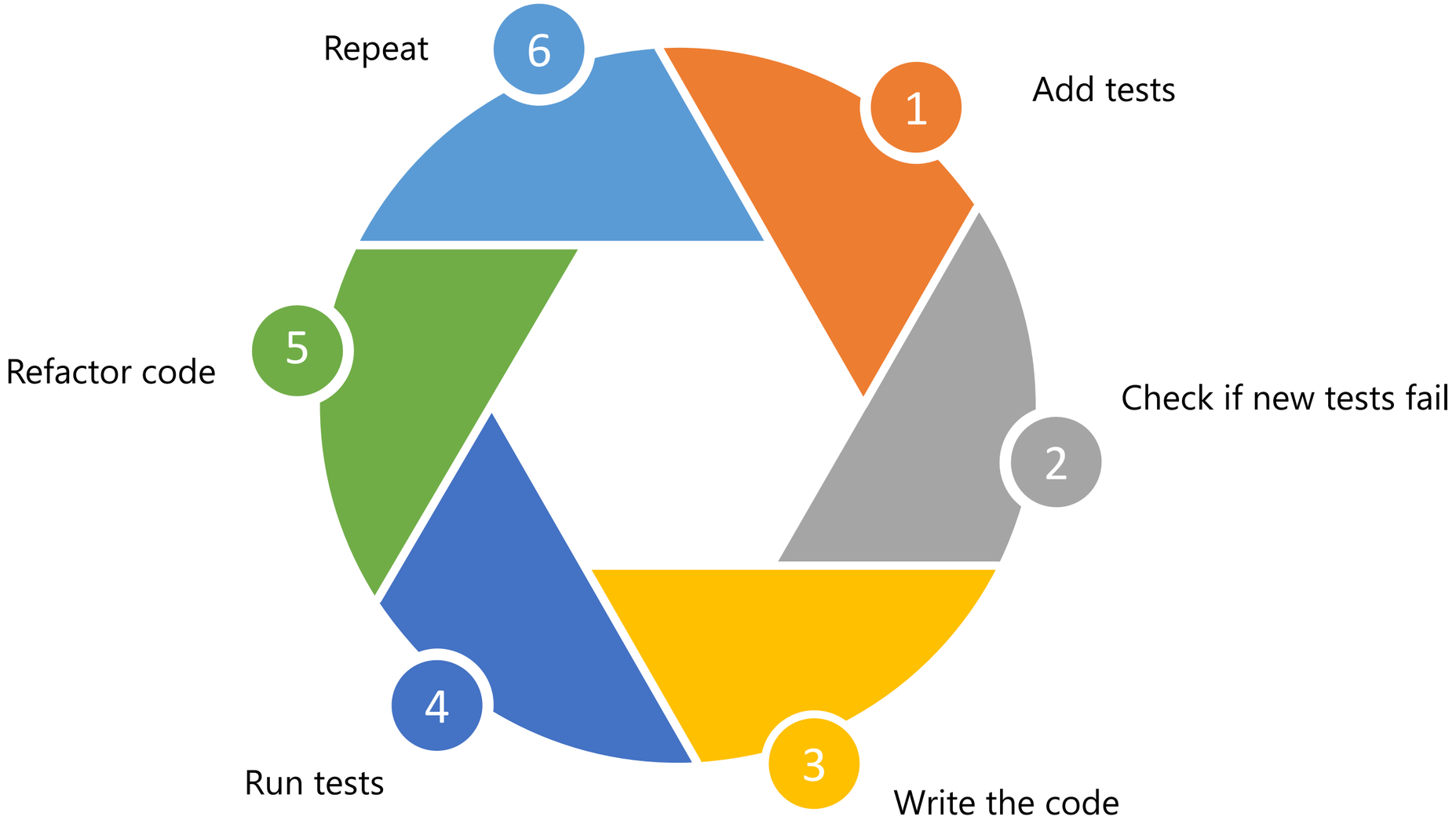
Test-Driven Development Approaches and JUnit 5

Overview of Test-Driven Development

Test-driven development is a software development process that depends on the repetition of short development cycle. Requirements are converted into test cases, then the software is improved to pass new tests. Developers apply the concept to improve and debug legacy code developed in older techniques.

Source:Wikipedia

Test-Driven Development Cycle



Test-Driven Development Advantages

- Fast Feedback
- Creation of solid code
- Improved Quality
- Creation of detailed specification
- Clear Interface
- Increase in Programmer's productivity
- Less rework
- Maintainable and Flexible
- Less bugs
- Less debugging
- Faster Time to market
- Simplification of codes
- Fast identification of errors
- Less Development cost
- Adaptability

General Approach

Test-driven development is related to the test-first programming concept of extreme programming. Recently, it has created general interest and is applied with minor semantic changes in the approach.

1. Add a check replaces add a test
2. Run all checks replaces run all tests
3. Do the work replaces write some code
4. Run all checks replaces run tests
5. Clean up the work replaces refactor code
6. Repeat

Overview of JUnit

Junit is a unit testing framework for the Java programming language. It is found that 10,000 Java projects used JUnit as an external library. Each library of the JUnit was used by 30.7% of the projects. It is the first option for developers while testing the Java applications.

Advantages of JUnit

The server requirement is eliminated to test web applications

Hierarchy of the program code can be tested as a single unit or multiple units

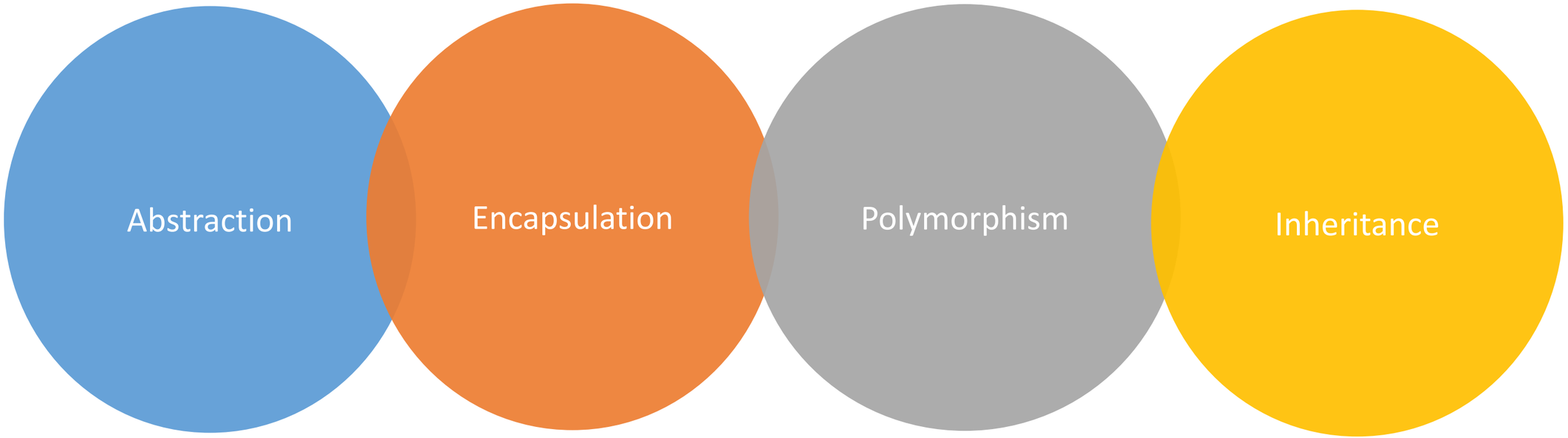
Supports test assertions and immediate test reporting

Simple framework for writing automated tests

Availability of a text-based command line and graphical test reporting mechanism

Java: A Programming Language

Java is an object-oriented programming language. It is intended to let the developers “write once, run anywhere”. Java applications are compiled to bytecode that can run on any Java virtual machine (JVM).



Java Program Template

A Java program is a collection of objects, classes, methods, and instance variables.

Syntax of Java program

```
Packages;
Access_Modifiers keyword Classname {

    Global variables;
    Access_modifiers keyword datatype functionName(Argument(s)){

        //statements and expressions;

    }
}
```

Junit 5 with Java

Software required:

- Java SE (Java 8 or higher versions)
- Eclipse IDE
- JUnit library

Assisted Practice

Duration: 30 mins

Test-Driven Development Approach with JUnit 5

Problem Statement: You are given a project to demonstrate the test-driven development approach using JUnit 5 and Java as a programming language.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice: Guidelines to Demonstrate the Usage of Junit 5 with Java

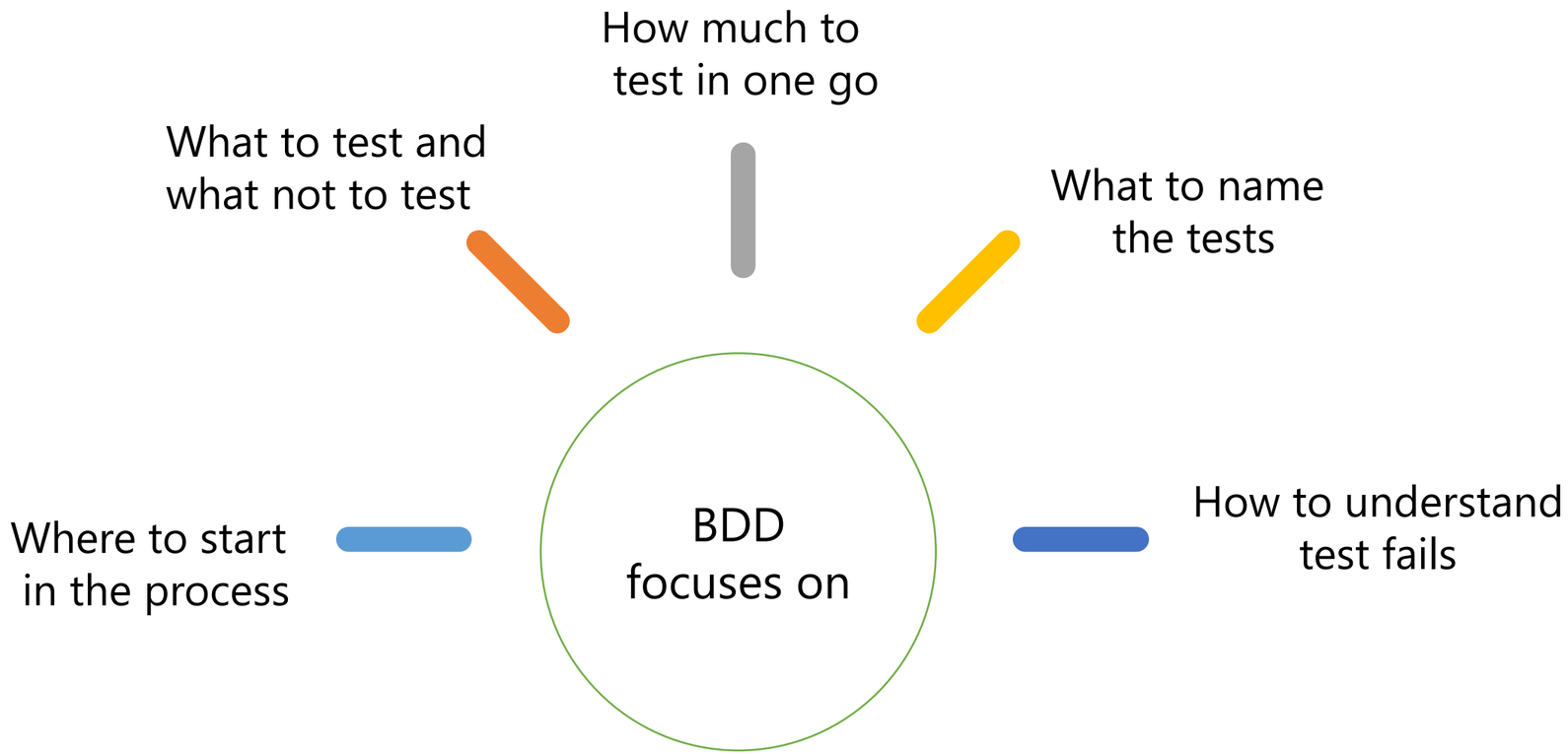
1. Download Java 8+ version and set up the environment variables.
2. Download eclipse from the official eclipse site.
3. Create a new Java project and refer the guide document, 4.1-JUnit demonstration with Java.
4. Create 2 classes to add and subtract numbers.
5. Generate the JUnit test files for each classes (Add the files in a different source directory for better indentation).
6. Confirm eclipse to add JUnit 5 library to the build path.
7. Confirm that the test cases are failed and the error logs are available in “Failure Trace” section in eclipse.
8. In the TestSrc folder, remove the comments (3 and 5) and uncheck the comments (1, 2, 4, 6).
9. Re-run the test cases and confirm that the tests are successful without any error or warning logs in the Failure Trace.

Software and Automation Testing Frameworks

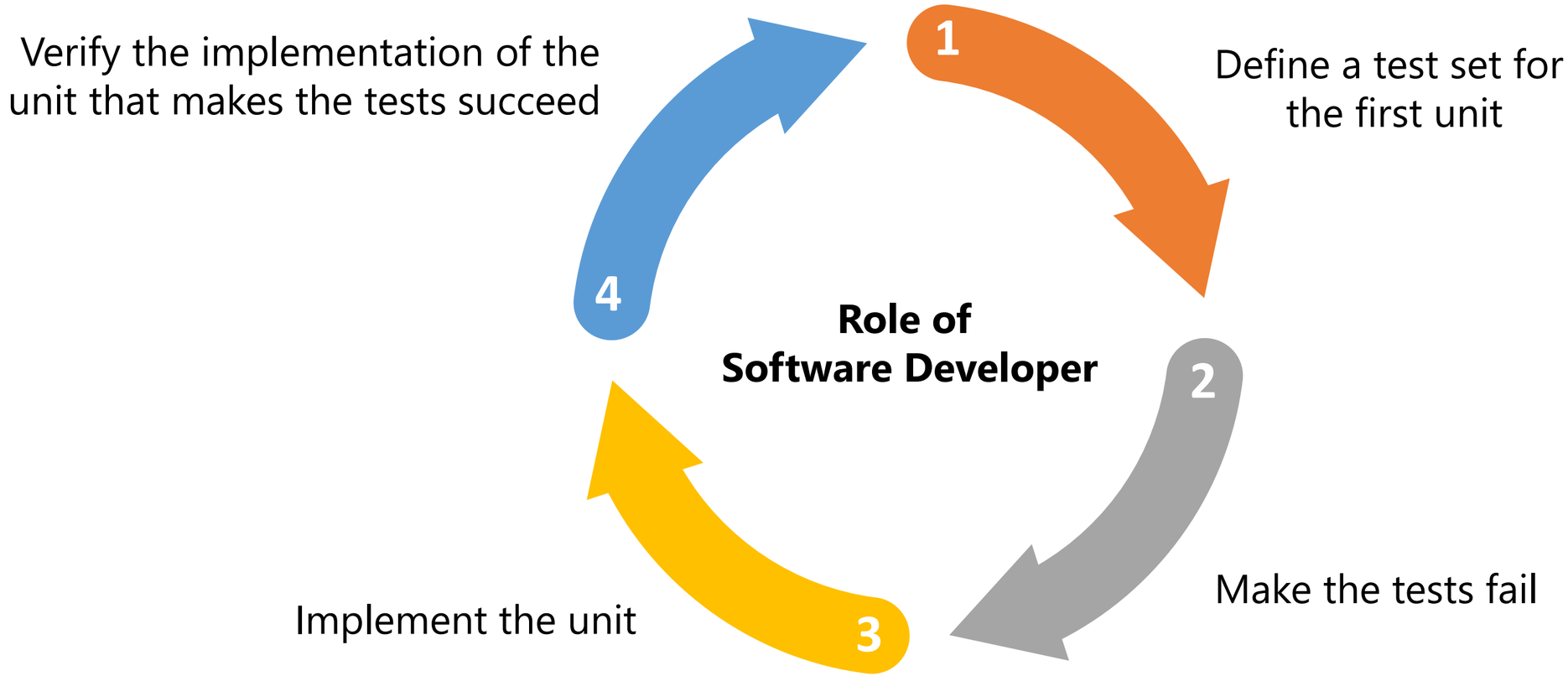
Behavior-Driven Development Principles, Cucumber, and its Applications

Overview of Behavior-Driven Development

Behavior-driven development is an extension of test-driven development.



Behavior Driven-Development Principles

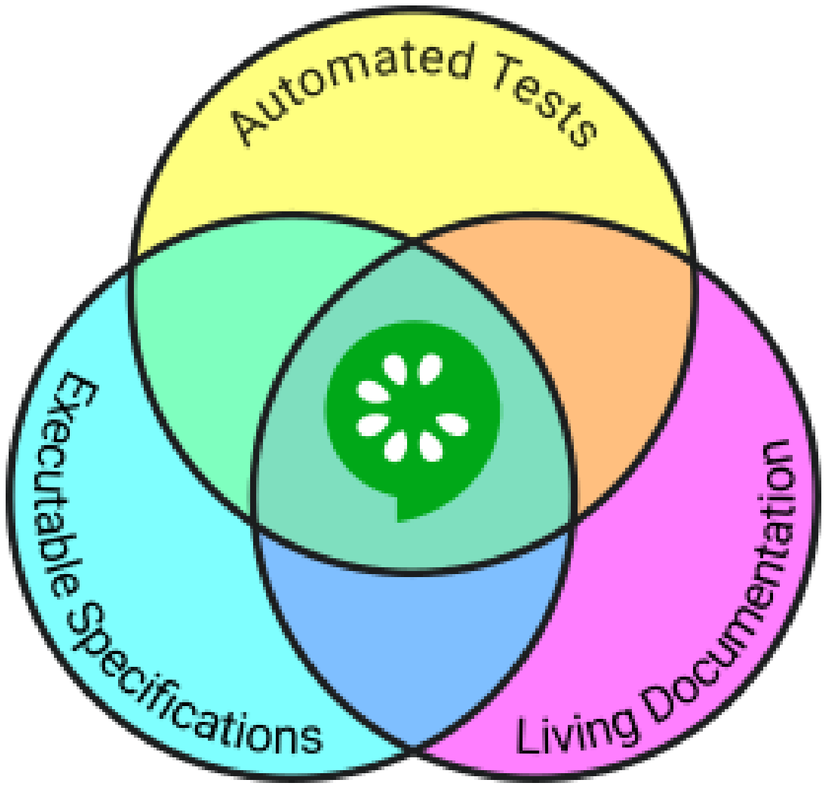


Overview of Cucumber

Cucumber is a software tool for testing other software by running automated acceptance tests written in a behavior-driven development approach. It allows the execution of feature documentation written in business-facing text. Gherkin language is used to define test cases. It is designed for non-technical, human readable, and collectively describes use cases related to software system.

Gherkin: A Language for Cucumber

Gherkin is a set of grammar rules that structures the plain text for cucumber to understand. It serves multiple purposes: Unambiguous executable specification, automated testing, and documentation about the system's actual behavior.



Gherkin Keywords

Each line of the script must start with a Gherkin keyword followed by any relevant text.

The primary keywords are:

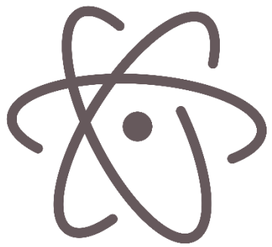
- Feature
- Scenario
- Given, When, Then, And, But (Steps)
- Background
- Scenario Outline
- Examples

The secondary keywords are:

- “ “ “ (Doc Strings)
- | (Data Tables)
- @ (Tags)
- # (Comments)

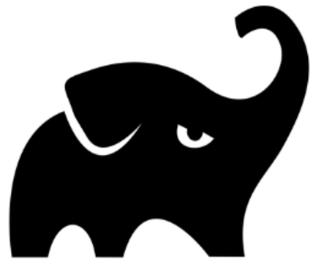
Tools for Cucumber

Supported Editors



Supported IDEs

Supported Build Tools

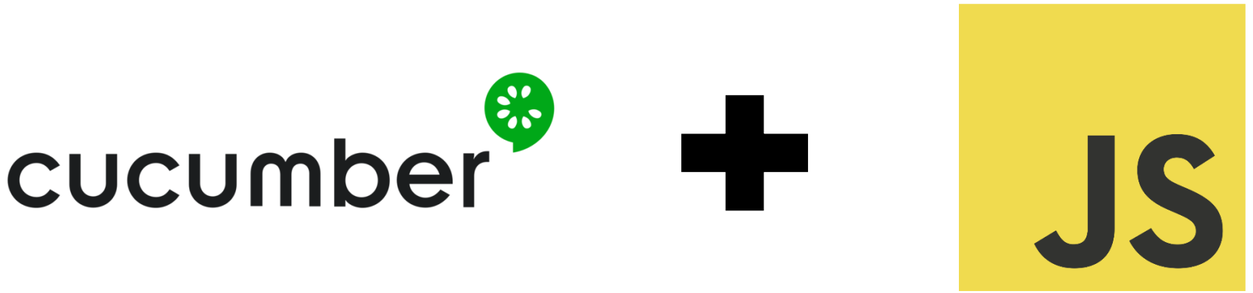


Cucumber with Java

Software required:

- Java SE (Java 9 and higher versions are not supported)
- Maven – Version 3.3.1 or higher
- IntelliJ IDEA Cucumber for Java plugin
- Cucumber Eclipse (An alternative for IntelliJ)

Cucumber with JavaScript



Software required:

- Node.js – A JavaScript runtime engine
- Text Editor – Atom, Visual Studio Code, or TextMate (For Mac Users Only)

Cucumber with Ruby

Software required:

- Ruby
- Bundler
- Text Editor

Cucumber with Kotlin

Software required:

- Java SE (Java 9 and higher versions are not supported)
- Maven – Version 3.3.1 or higher
- IntelliJ IDEA Cucumber for Java plugin and IntelliJ IDEA Kotlin plugin
- Cucumber Eclipse and Kotlin Eclipse (An alternative for IntelliJ)

Assisted Practice

Duration: 45 mins

Behavior-Driven Development Approach with Cucumber

Problem Statement: You are given a project to demonstrate the behavior-driven development approach using cucumber and JavaScript as a language.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice: Guidelines to Demonstrate the Usage of Cucumber

1. Install node.js and npm from the node.js official website.
2. Open the Visual Studio Code terminal and check the version of node and npm.
3. Create an empty cucumber project folder.
4. Add cucumber as a development dependency.
5. Create a folder “feature” and a sub-folder “steps” in it.
6. Add the codes in the respective files. (Codes are available in the guide Document, 4.2 – Cucumber installation).
7. Verify cucumber installation.
8. Follow the guide document, 4.3 – Cucumber application with JavaScript
9. Confirm that all the scenarios and steps are passed.

Key Takeaways

You are now able to:

- Describe the benefits in traditional and agile approach of software testing
- Describe the levels and approaches of software testing
- Demonstrate test-driven development framework with JUnit
- Demonstrate behavior-driven development framework with cucumber

Knowledge Check

Knowledge Check 1	_____ **set of JUnit tests focuses on test-driven development.**

a. Grey-box

b. Sanity

c. Structure-based

d. All of the above

Knowledge Check 1	_____ **set of JUnit tests focuses on test-driven development.**

a. Grey-box

b. Sanity

c. Structure-based

d. All of the above

The correct answer is **c. Structure-based**

Structure-based testing, tests the structure of the code and is a set of JUnit tests focused on test-driven development.

Knowledge Check 2	**White-box testing is carried out to test the _____ components.**

a. Dynamic

b. Static

c. Final

d. All the above

Knowledge Check

2

White-box testing is carried out to test the _____ components.

a. Dynamic

b. Static

c. Final

d. All the above

The correct answer is **a. Dynamic**

White-box testing, tests the dynamic components and the internal structure of the code.

Knowledge Check

3

Which of the following is triggered by migrating the existing software?

a. Maintenance Testing

b. Sanity Testing

c. White-box Testing

d. None of the above

Knowledge Check

3

Which of the following is triggered by migrating the existing software?

a. Maintenance Testing

b. Sanity Testing

c. White-box Testing

d. None of the above

The correct answer is **a. Maintenance Testing**

Maintenance testing is in charge of the testing on the already deployed software.

Knowledge Check 4	**Cucumber software is written in _____ form.**

a. C

b. Java

c. C++

d. Ruby

Knowledge Check

4

Cucumber software is written in _____ form.

a. C

b. Java

c. C++

d. Ruby

The correct answer is **d. Ruby**

Knowledge Check

5

What is feature file in Cucumber?

a. Feature

b. Scenario

c. Scenario Outline

d. All of the above

Knowledge Check

5

What is feature file in Cucumber?

a. Feature

b. Scenario

c. Scenario Outline

d. All of the above

The correct answer is **d. All of the above**

Feature file is an entry point to all the cucumber tests. It is used to describe the tests in Gherkin language.

Lesson-End Project

Behavior-Driven Development Approach with Java and Kotlin

Duration: 90 mins

Problem Statement:

Create a directory named **Lesson-04** and perform the following:

- Install cucumber on IntelliJ for Java and Visual Studio Code for Kotlin.
- Execute the 3 scenarios: Undefined, Failed, and Passed for Java.
- Execute the 3 scenarios: Undefined, Failed, and Passed for Kotlin.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You