SOFTWARE TEST

Assoc. Prof. Dr. Mehmet Akif Çifçi

What is Software Testing?

- Software testing is a process performed to verify the quality and functionality of a software, and includes a set of activities to determine that the software meets the intended requirements, produces correct and expected results, and operates consistently.
- This is an important part of the software development process and is an inspection process to assess whether the software is ready for release. It is also used to detect bugs, increase software reliability, improve usability and ensure customer satisfaction.

What is Software Testing?

- Testing can be performed using different methods, techniques and tools. Testers/engineers create test cases, automated tests and manual tests to test different functions and interfaces of the software. In this process, the correctness and stability of the software is evaluated using predicted inputs and expected outputs. It can also be aimed at identifying potential problems such as bugs, error messages, performance issues and security vulnerabilities.

What is Software Testing?

- Software testing is important to deliver robust and reliable software to the user. Early detection and correction of defects can reduce costs and ensure the successful operation of the software. Furthermore, software testing provides a continuous feedback mechanism to improve the quality of the software development process and create a reliable product.

Types of Software Testing

Software test types can be categorized under two main headings. These are

- Manual Test
- Automation Test

Manual Test

- Manual testing is a software testing process in which test cases are executed manually without the use of an automated tool. All test cases are executed manually by the tester with the end user in mind. Test case reports are also generated manually.

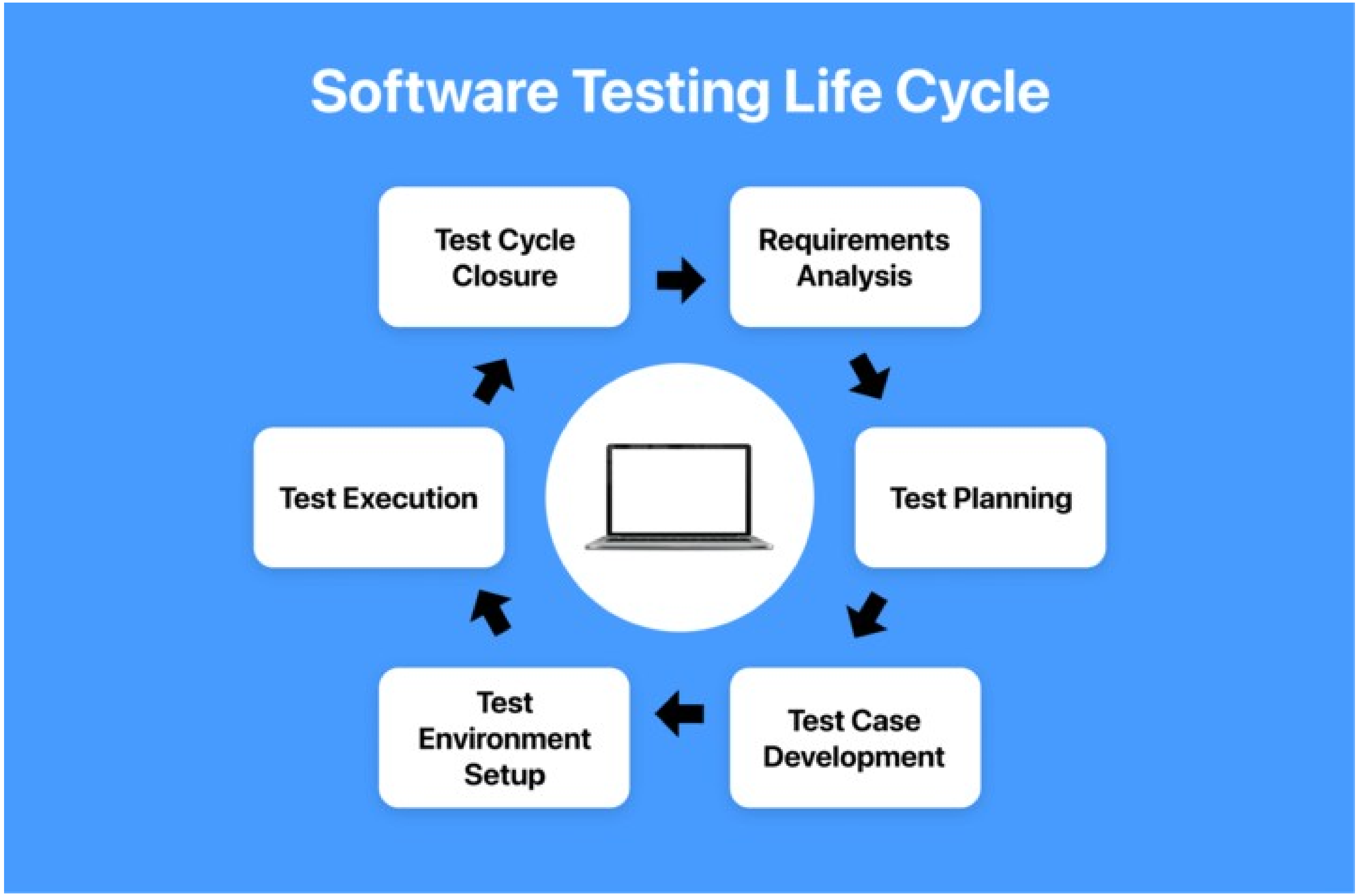
- Manual Testing is one of the most fundamental testing processes as it can find both visible and hidden bugs of the software. Manual testing is mandatory for every newly developed software before automation testing.

Automation Test

- Automation testing is a software testing method that uses specialized software tools and then compares actual test results with predicted or expected results.
- Automation testing can be performed by writing scripts or using any automation testing tool. Test automation is used to facilitate repetitive tasks and other testing tasks that are time-consuming and difficult to perform manually.

Software Testing Life Cycle

- The software test life cycle (STLC) is a good method for the stages in the testing process. The phases of the STLC are as follows:
  - Requirements,
  - Test Planning,
  - Test Case Development,
  - Testbed Setup,
  - Test Implementation (Running the Test),
  - Test Reporting


Software Testing Life Cycle
Test Cycle Closure
Requirements Analysis
Test Planning
Test Case Development
Test Environment Setup
Test Execution

Requirements

- The Requirements Phase is the first step in the Software Testing Life Cycle (STLC). In this phase, the test team tries to understand the requirements for what to test. The activities of the whole team involved in testing in this phase include brainstorming for requirements analysis and the process of identifying test requirements. The requirements phase helps to determine the test scope.

Test Planning

- Test Planning is the most productive phase of the software testing lifecycle where all test phases are defined. In this phase, the manager of the test team calculates the estimated effort and cost for the test effort. The test team then analyzes the risks involved and defines timelines and test environments to create a strategy.

Test Scenario Development

- The test case development phase starts after the test planning phase is completed. In this phase, the test team notes detailed test cases and prepares the necessary test data. All necessary automation scripts are also created at this stage. Test cases are written by the test team in stages. After the necessary reviews and some changes are made, the test team creates the test data on the basis of the prerequisites with the approval of the test cases.

Test Environment Setup

- In the test environment, the conditions for an efficient product test are defined and usually completed together with test case development. The test team does not have to be very active here. The project owner and developers provide the test environment. The tester's task here is to perform smoke tests and check whether the requirements for the environment are ready.

Test Implementation (Running the Test)

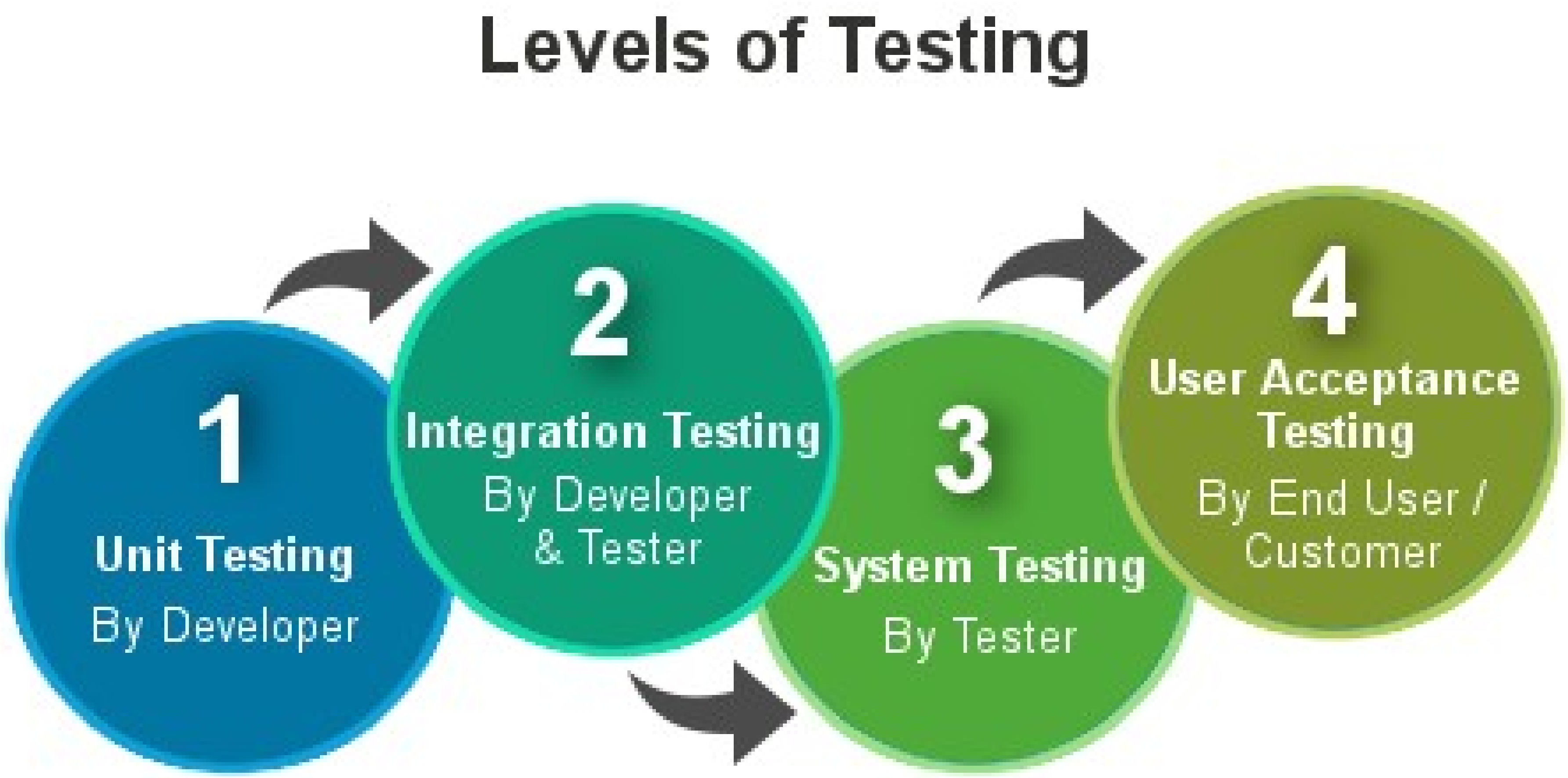
- At this stage, the QA team runs tests according to test plans and prepared test cases and documents the results. They identify bugs and provide feedback to the development team to fix them. In the following process, the corrected errors are checked again by the testers. These tests verify that user requirements are met.

Test Reporting

- It is important to report the test data obtained after the test and analyze the errors. Test plans are adhered to and evaluated by taking into account the specified test pass and fail criteria. After completing the test cycle, test closure report and test measurements are prepared.

Software Test Levels

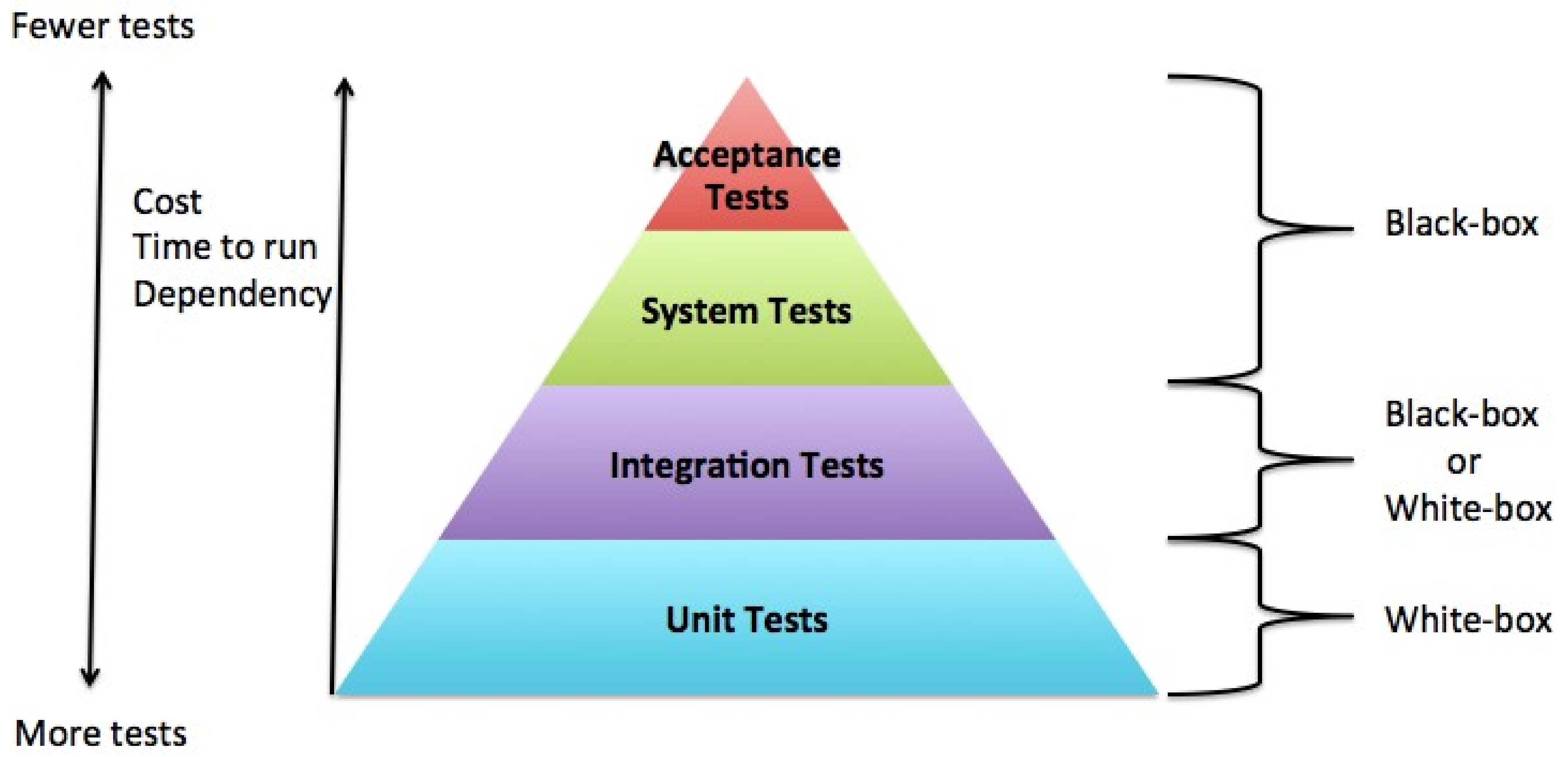
- Testing is an evaluation process to find out whether a system or its components meet specified requirements. This process consists of technical levels, starting with software development professionals and extending to the end user. The testing levels are mainly to identify missing areas and avoid duplication between SDLC phases. The software development life cycle (SDLC) consists of several phases: requirements analysis, design, development, testing and maintenance. Each phase also goes through testing. Therefore, there are various levels of testing.


Levels of Testing
1
Unit Testing
By Developer
2
Integration Testing
By Developer & Tester
3
System Testing
By Tester
4
User Acceptance Testing
By End User / Customer

Software Test Levels

There are mainly four test levels. These are

- Unit testing
- Integration testing
- System testing
- Acceptance testing


Fewer tests
Cost
Time to run
Dependency
More tests
Acceptance Tests
System Tests
Integration Tests
Unit Tests
Black-box
Black-box or White-box
White-box

Unit Testing

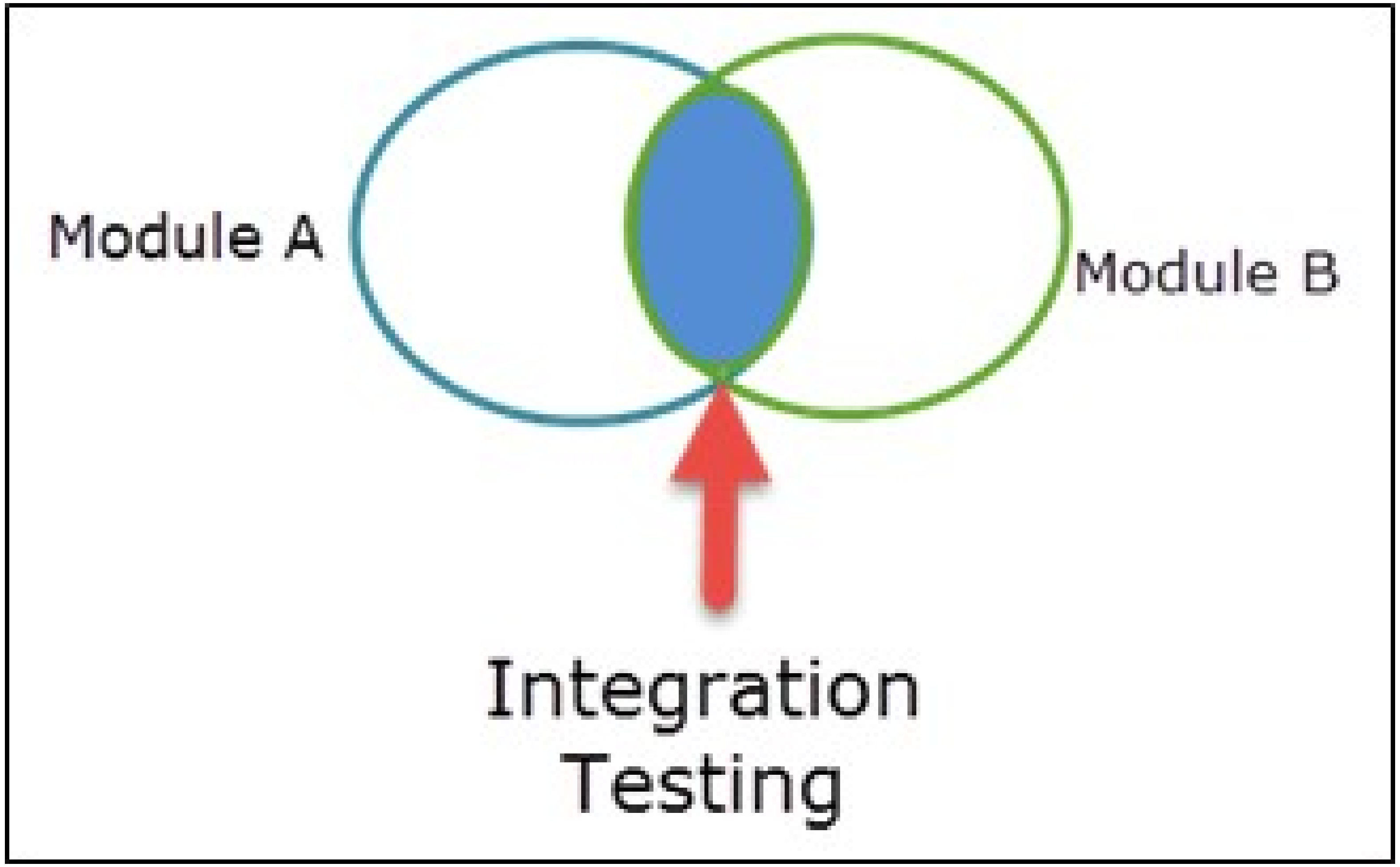
- Unit or component testing is the most basic type of testing. Unit testing isolates each component of the software (modules, programs, classes, objects, etc.) and checks its functioning. It is also done to prove that each component is correct in terms of fulfillment of requirements and functionality.
- Unit testing is usually done by software developers with the help of a simulator or development environment. Errors that arise are corrected immediately without being recorded. This type of testing should be done at the earliest stages of the software development process and the software should be delivered to the test team in this way.

Unit Testing

- Unit testing is usually done by software developers with the help of a simulator or development environment. Errors that arise are corrected immediately without being recorded. This type of testing should be done at the earliest stages of the software development process and the software should be delivered to the test team in this way.
- A software developer should see unit testing as part of the software development process. Through unit testing, bugs are detected during the software development process. This saves time and money. The BDD (Behavior Driven Development) approach, which reduces costs in the SDLC cycle, is the biggest proof of the importance of writing unit tests.

Integration Testing

- Integration means joining together. Integration testing can be performed during the integration of the components of a software or during the integration of two different software. Therefore, integration testing can be done at different test levels as unit integration testing and system integration testing.
- Components that software developers test separately during unit testing can cause errors when they are integrated together. Integration testing aims to test whether these different components (units) of the system work correctly together.


Module A
Module B
Integration Testing

System Testing

- System testing is performed on a complete and integrated system. It checks the suitability of the system according to the requirements. It tests the overall interaction of components. It includes load, performance, reliability and security testing.
- System testing is the final test to verify that the system meets the specification. Therefore, both functional and non-functional requirements are evaluated for testing. System testing on functional requirements starts by using black box techniques that are most appropriate for the system to be tested. Then, white box techniques can be used to catch bugs that black box testing does not catch.

Acceptance Testing

- The purpose of acceptance testing is to build confidence in the system, parts of the system or non-functional requirements of the system. The main focus in acceptance testing is not to find bugs, but to show that the system is ready to go live. However, even if the main focus is not on finding bugs, many bugs can be uncovered, ranging from typos to bugs that can cause a major problem in implementation.
- Acceptance testing is generally done by the user or customer. However, other stakeholders can also be involved.

Acceptance testing can take place in many stages:

- Alpha Testing: Companies developing packaged software want to get feedback from potential or existing customers before releasing the software for sale. Alpha testing is done in a controlled manner within the company developing the software.
- Operational (Acceptance) Test: Acceptance of the system by system administrators.
- Contractual Acceptance Testing: Contractual acceptance testing is performed according to the terms of the contract for customer-specific software. The acceptance criteria must be determined when the parties accept the contract.

Acceptance testing can take place in many stages:

- Regulatory Acceptance Testing: Regulatory acceptance testing is performed in accordance with government-defined regulations that must be followed, such as legal or safety regulations.
- User Acceptance Test: Verifies that the system is suitable for use by end users.
- Beta (Field) Testing: Beta testing or field testing is carried out in an uncontrolled manner in the customers' or potential customers' own environments.

Test Strategies

- Test-Driven Development
- Behavior-Driven Development
- Acceptance Test Driven Development

Test-Driven Development

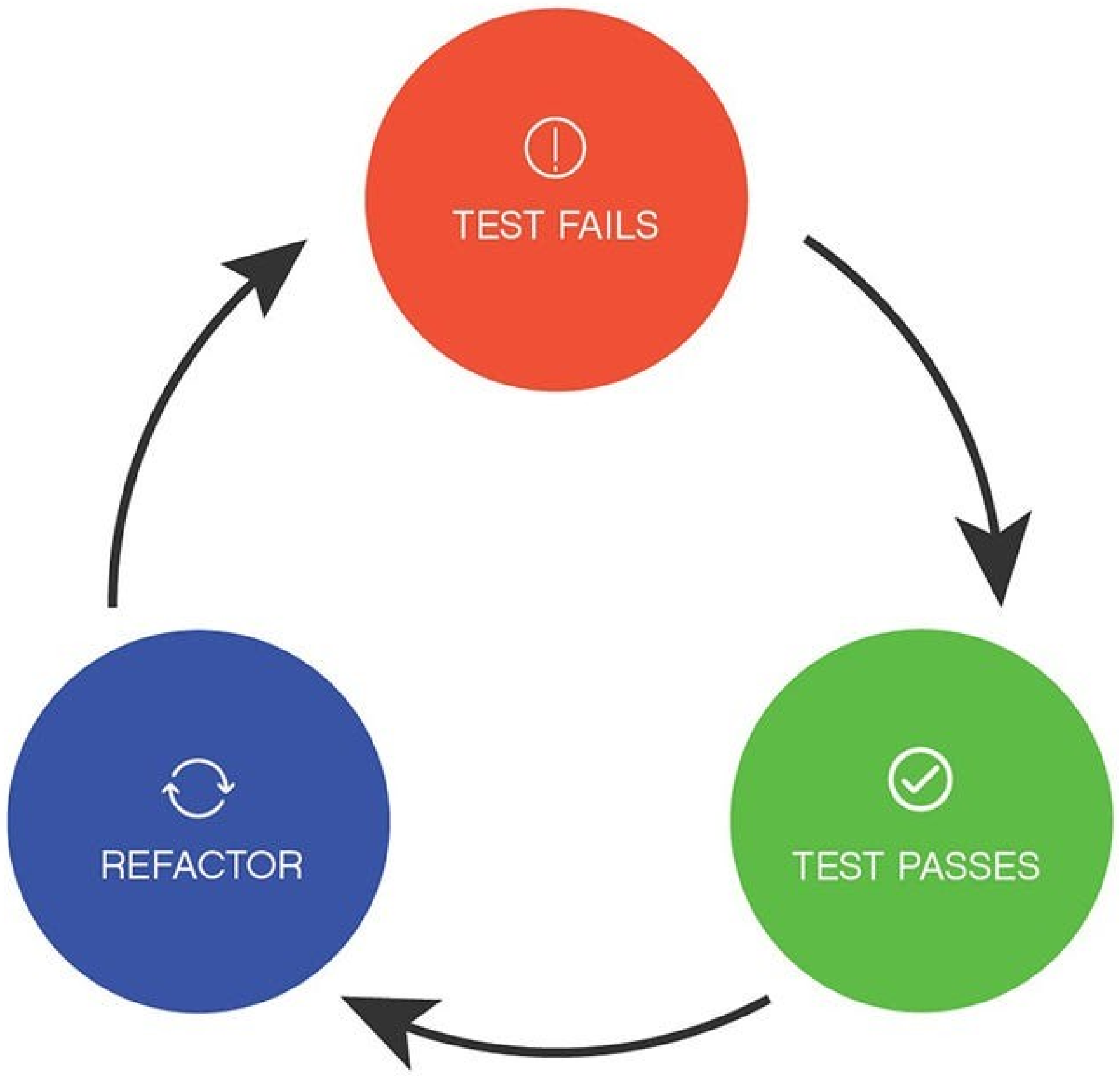
Test-driven development is done by writing Unit Tests to test the code and separate dependencies from the code. The important thing in this strategy is to write the tests before the code is written.

- The TDD process includes the following steps;
- Based on the specified requirements, the software developer writes a test case
- These tests are performed and are expected to fail because they were written before the development of a feature
- Coding is done by the development team to pass the test successfully
- All tests are ensured to be successful
- The code is reviewed and edited again. Improvement or cleanup is done

TDD Advantages

- Errors are easily visible and feedback is immediately available to correct them
- Encourages the development of cleaner and better designs
- The code is testable when written
- As software developers get used to the TDD process, their productivity increases
- The code is easy to understand because it is driven by tests. This allows other team members to easily work on the code in the absence of any team member. This encourages knowledge sharing and collaboration in the team
- Gives the developer the confidence to easily change the broad architecture of the application

TDD Cycle

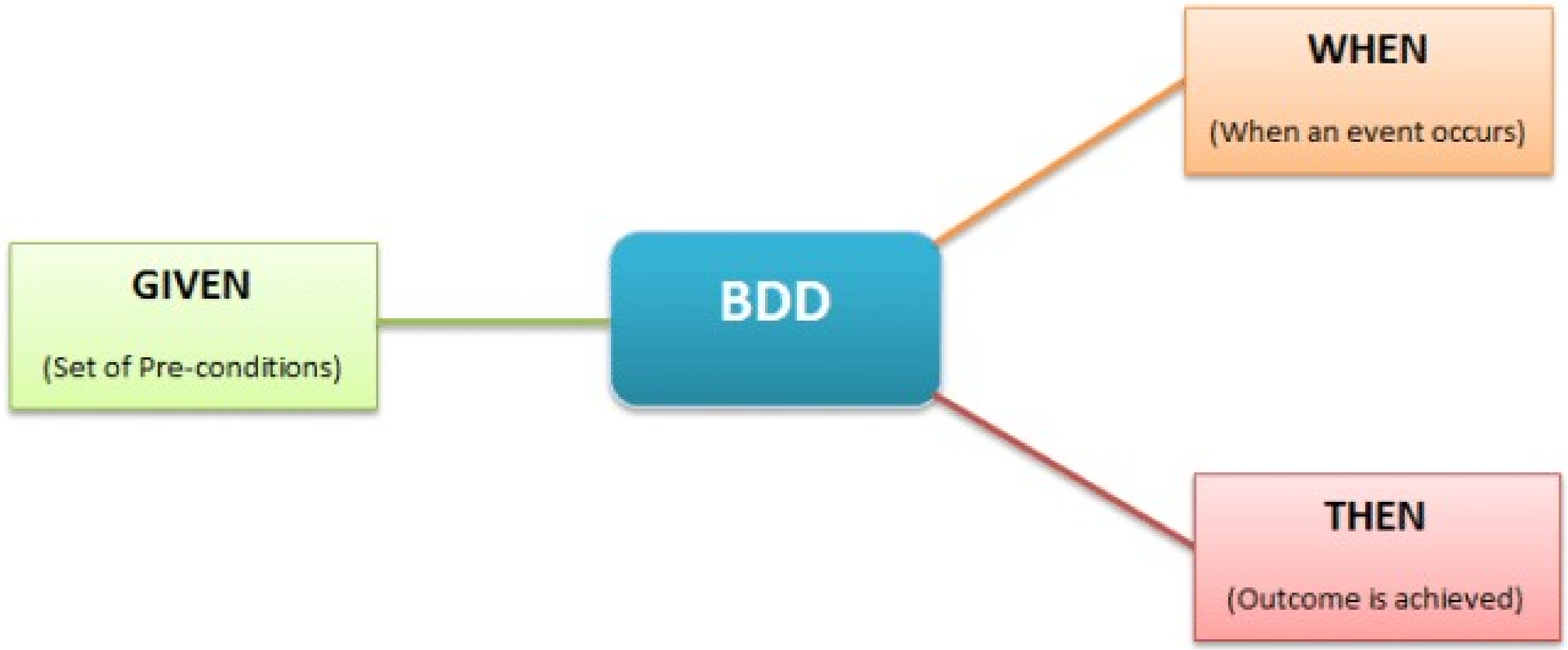


Behavior-Driven Development

- Behavior Driven Development is a derivative of Test Driven Development. Testing, however, is mainly based on system behavior.
- While Test Driven Development asserts some conditions, Behavior Driven Development is written in plain text using real-time instances of actual requirements to describe the behavior of the system. In this way, with these texts, you ensure that you have an up-to-date document of the application.
- Commands such as Given, When, Then are used when writing test cases.

BDD Advantages

- Helps reach a wider audience through the use of non-technical language
- You move to a higher level of abstraction in describing the behavior of the software, so you can write tests with more business functions
- Improves communication between software developers, testers and product owners
- Acceptance criteria can be established before development

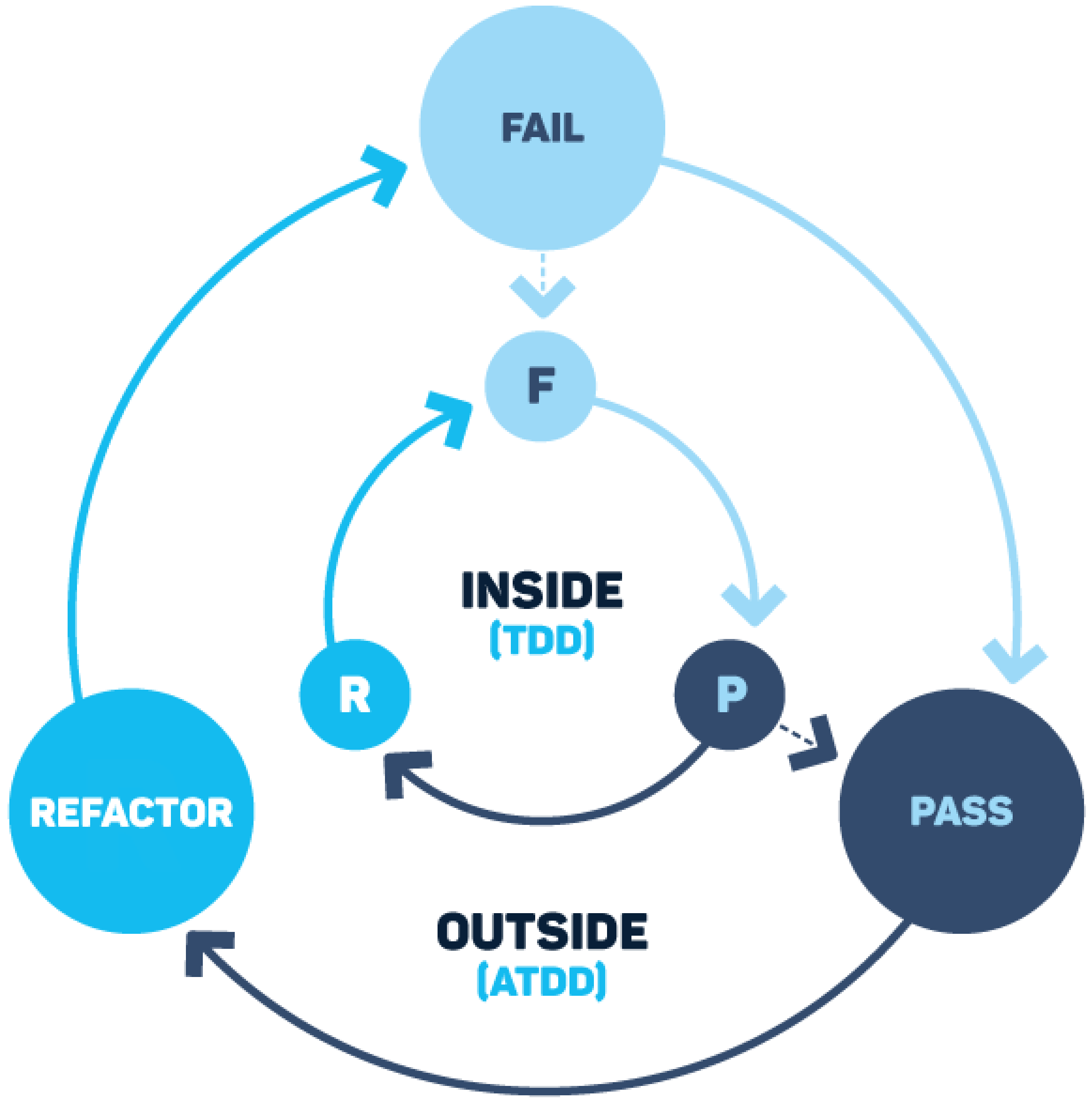

GIVEN
(Set of Pre-conditions)
BDD
WHEN
(When an event occurs)
THEN
(Outcome is achieved)

Acceptance Test Driven Development

- Acceptance Test Driven Development tests, a single acceptance test is written from the user's perspective. It is similar in nature to Behavior Driven Development, but Acceptance Test Driven Development focuses mainly on the code that meets the requirements of the software, with a focus on predicting the functional behavior of the system. Like BDD, the tests are written in plain text, but focus on authoring acceptance. So ATDD turns BDD into an executable test specification, and this specification can be automated.

ATDD Advantages

- Requirements are analyzed very clearly without any ambiguity
- Ensures good communication across the whole team
- Acceptance tests serve as a guide and guide the software to where it needs to get to
- Automation of ATDD can reduce feedback time

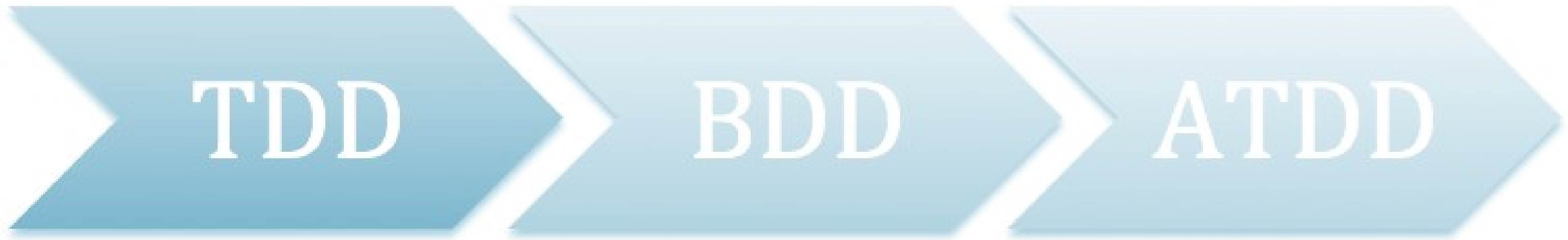

FAIL
F
INSIDE
(TDD)
R
P
REFACTOR
PASS
OUTSIDE
(ATDD)

Differences between Strategies

- TDD is more technical and is written in the same language in which the feature is implemented. For example, if implemented in Java, JUnit tests are written. BDD and ATDD are written as text in plain English.
- The TDD approach focuses on the implementation of a feature. BDD focuses on the behavior of the feature. ATDD focuses on capturing requirements.

Differences between Strategies

- To implement TDD we need to have technical knowledge. BDD and ATDD do not require any technical knowledge. The benefit of BDD and ATDD is that both technical and non-technical people can participate in developing this feature.
- As TDD evolves, it provides space for good design and focuses on the aspect of meeting the requirement. BDD and ATDD focus on a different aspect that is suitable for use.


TDD
BDD
ATDD

Doc. Dr. Mehmet Akif Cifci

- Technical University of Vienna (Austria)
- University of Klaipeda (Lithuania)
- Bandirma Seventeen Eylul University

To Follow and Connect

https://github.com/themanoftalent
https://www.linkedin.com/in/themanoftalent/
https://www.researchgate.net/profile/Mehmet-Akif-Cifci