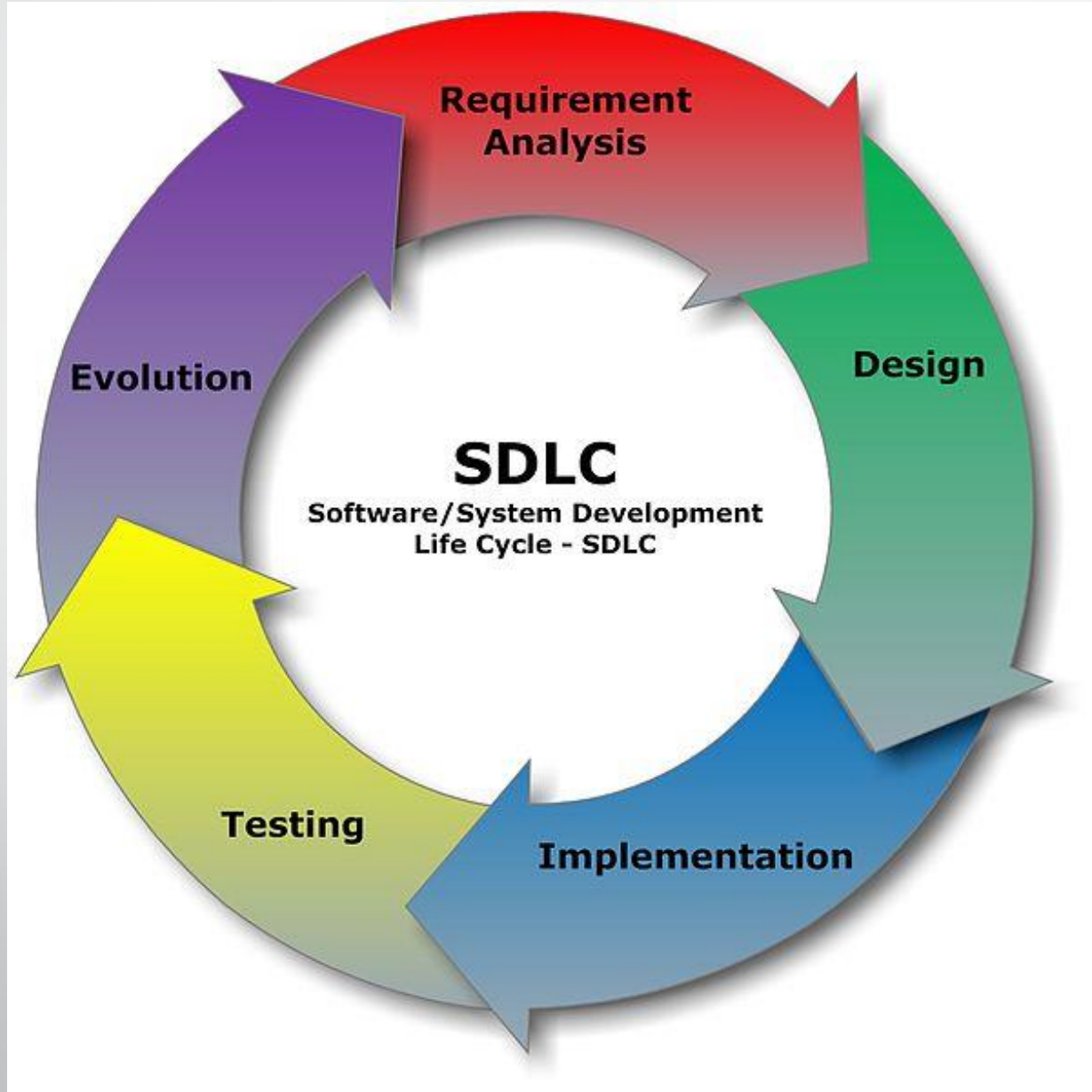
JS Unit testing Techniques

Software /system development life cycle (SDLC)



What is Testing?

- Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.
- Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.
- A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Why is Software Testing Important?

- Testing is important as it uncovers a defect before it is delivered to the customer ensuring the quality of software.
- So that the defects or bugs can be identified in the early stages of development; as later the stage in which bug is identified, more is the cost to rectify it.
- Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.
  - In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
  - China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live

Who does Testing?

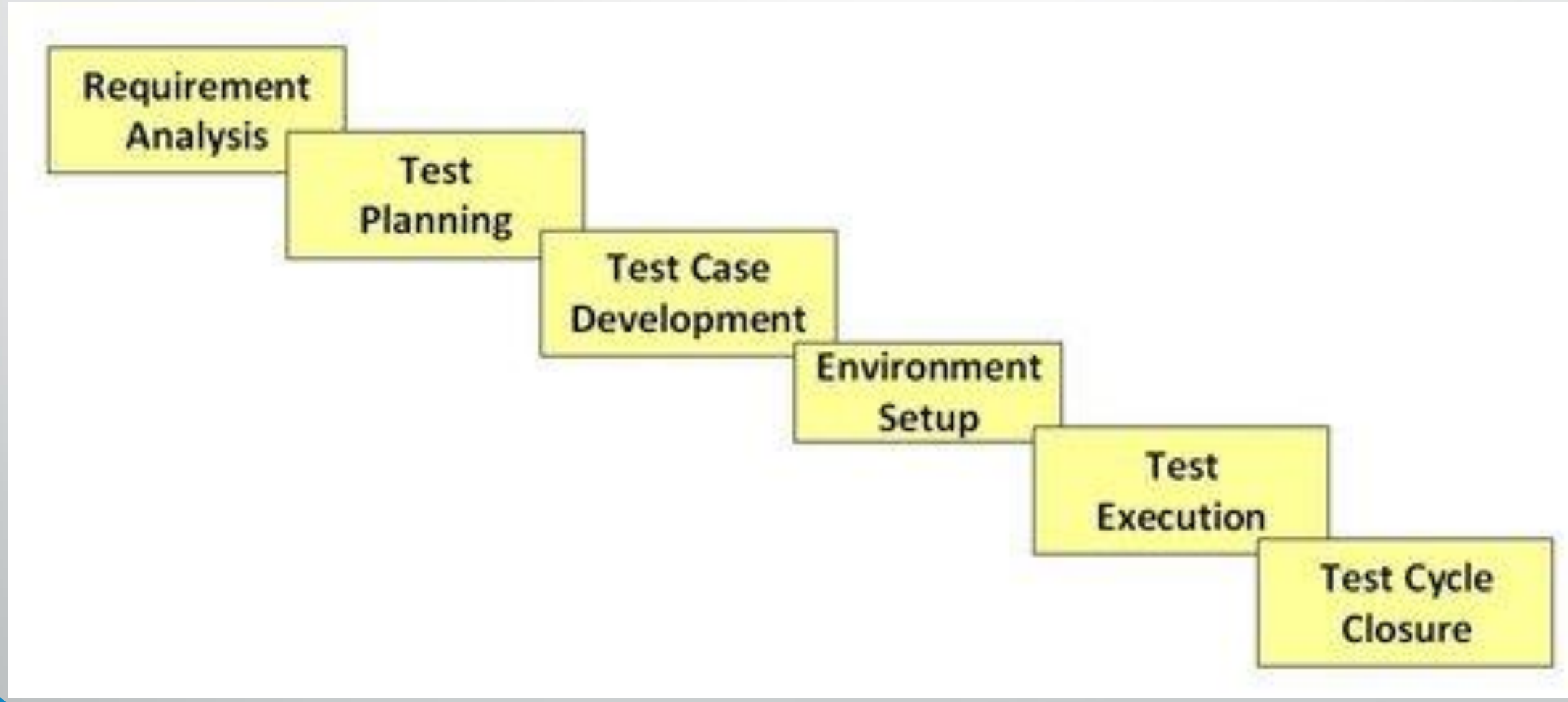
- In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements
- developers also conduct testing which is called **Unit Testing**.
- the following professionals are involved in testing a system within their respective capacities –
  - Software Tester
  - Software Developer
  - Project Lead/Manager
  - End User

STLC
Software Testing Life Cycle

What is Software Testing Life Cycle (STLC)?

- SOFTWARE TESTING LIFE CYCLE(STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met
- It consists of a series of activities carried out methodologically to help certify your software product

Different Phases of the STLC



Requirement Analysis

- During this phase, test team studies the **requirements** from a testing point of view to identify the testable **requirements**.
- The team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) **to understand the requirements in detail.**
- Requirements could be either **Functional** (defining what the software must do) or **Non Functional** (defining system performance /security availability)

- **Activities**
  - Identify types of tests to be performed.
  - Gather details about testing priorities and focus.
  - Prepare Requirement Traceability Matrix (RTM).
  - Identify test environment details where testing is supposed to be carried out.
  - Automation feasibility analysis (if required).
- **Deliverables**
  - RTM
  - Automation feasibility report. (if applicable)

Test Planning

- Typically, in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan. In this phase Test Strategy is also determined.
- **Activities**
  - Preparation of test plan/strategy document for various types of testing
  - Test tool selection
  - Test effort estimation
  - Resource planning and determining roles and responsibilities.
- Deliverables
  - Test plan /strategy document.
  - Effort estimation document.

Test Case Development

- This phase involves the creation, verification and rework of test cases & test scripts. Test data, is identified/created and is reviewed and then reworked as well.
- Activities
  - Create test cases, automation scripts (if applicable)
  - Review and baseline test cases and scripts
  - Create test data (If Test Environment is available)
- Deliverables
  - Test cases/scripts
  - Test data

Test Environment Setup

- This phase involves the creation of a test environment closely simulating the real-world environment.
- **Activities**
  - Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
  - Setup test Environment and test data
  - Perform smoke test on the build
- **Deliverables**
  - Environment ready with test data set up
  - Smoke Test Results.

Test Execution

- During this phase, the testers will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.
- **Activities**
  - Execute tests as per plan
  - Document test results, and log defects for failed cases
  - Map defects to test cases in RTM
  - Retest the Defect fixes
  - Track the defects to closure
- **Deliverables**
  - Completed RTM with the execution status
  - Test cases updated with results
  - Defect reports

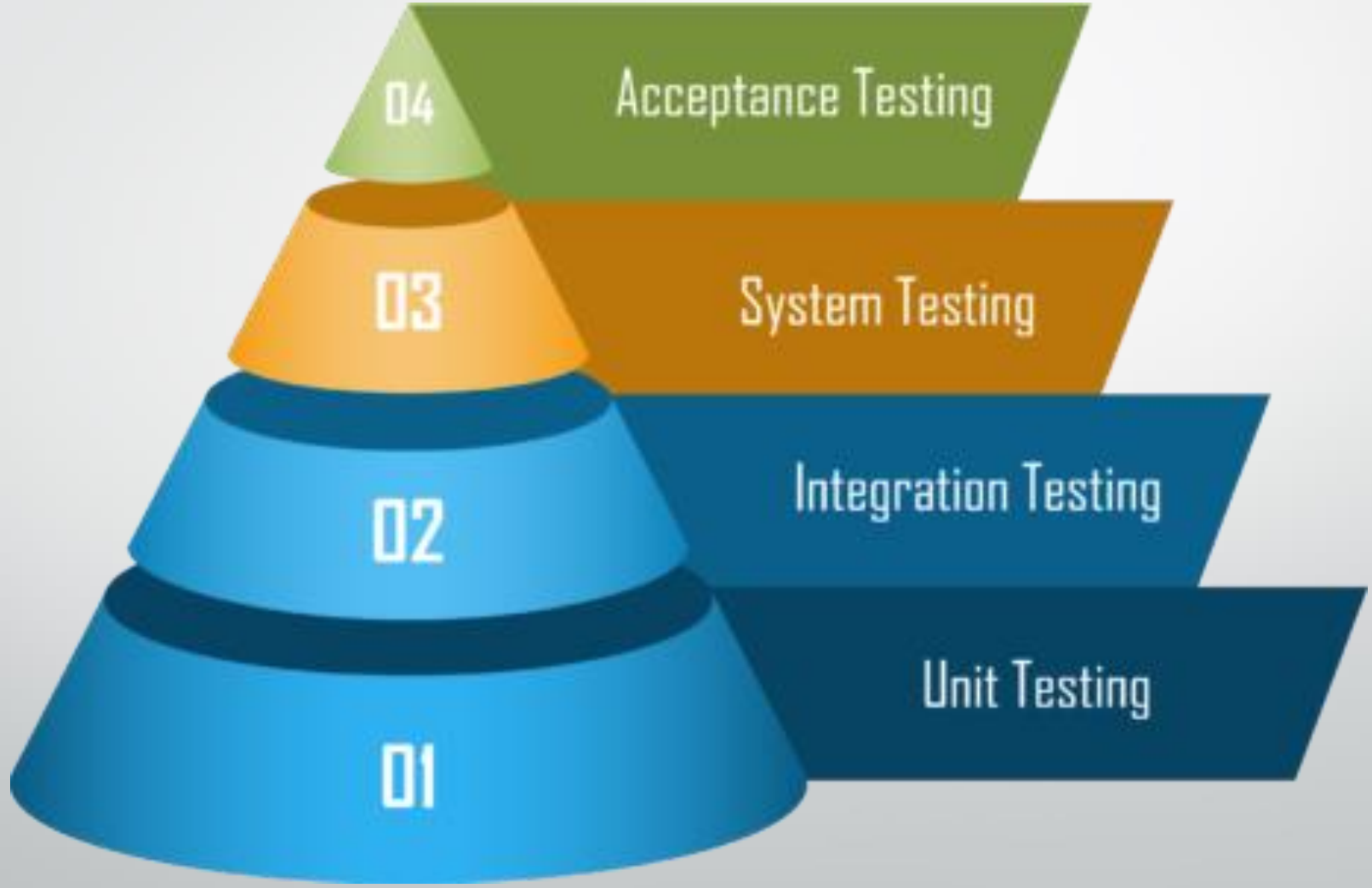
Test Cycle Closure

- Testing team will meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in the future, taking lessons from the current test cycle
- Activities
  - Evaluate cycle completion criteria based on Time, Test coverage, Cost,Software, Critical Business Objectives, Quality
  - Prepare test metrics based on the above parameters.
  - Document the learning out of the project
  - Prepare Test closure report
  - Qualitative and quantitative reporting of quality of the work product to the customer.
  - Test result analysis to find out the defect distribution by type and severity.
- **Deliverables**
  - Test Closure report
  - Test metrics

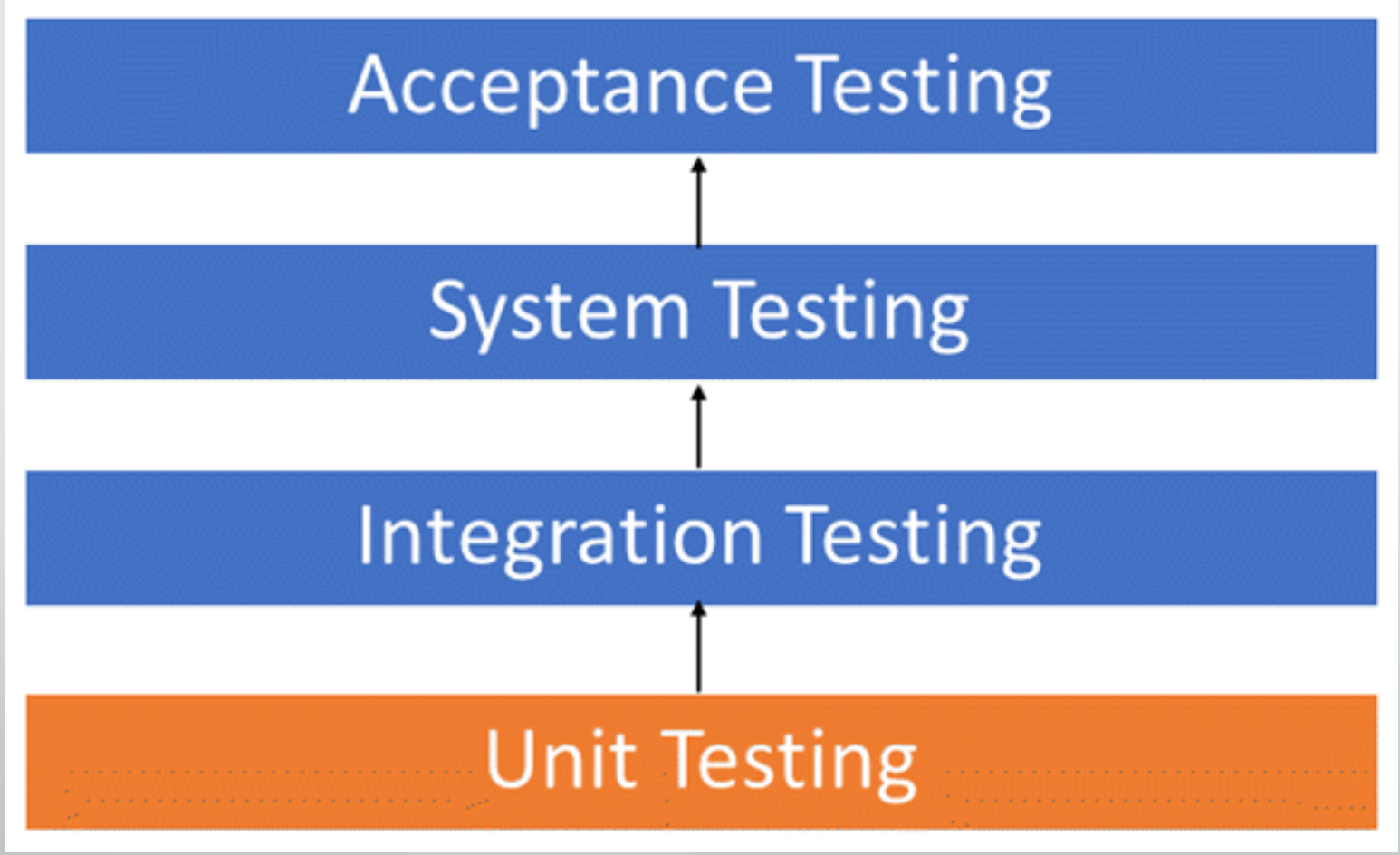
Types of Testing

- Manual Testing
  - Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug
- Automation Testing
  - Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process.
  - Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Different Levels of Testing



Unit Testing



What is Unit Testing?

- **UNIT TESTING** is a type of software testing where individual units or components of a software are tested.
- The purpose is to validate that each unit of the software code performs as expected.
- Unit Testing is done during the development (coding phase) of an application by the developers.
- Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

Why Unit Testing?

- Proper unit testing done during the development stage saves both time and money in the end. Here, are key reasons to perform unit testing.
  - Unit Tests fix bug early in development cycle and save costs.
  - It helps understand the developers the code base and enable them to make changes quickly
  - Good unit tests serve as project documentation
  - Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code till the tests run again.

How to do Unit Testing ?

- Unit Testing is of two types
  - Manual
  - Automated
- Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.

Unit testing Under the automated approach

- A developer **writes a section of code in the application just to test the function**. They would later comment out and finally remove the test code when the application is deployed.
- A developer could **also isolate the function** to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment.
- A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework,
- The workflow of Unit Testing is 1) Create Test Cases 2) Review/Rework 3) Baseline 4) Execute Test Cases.

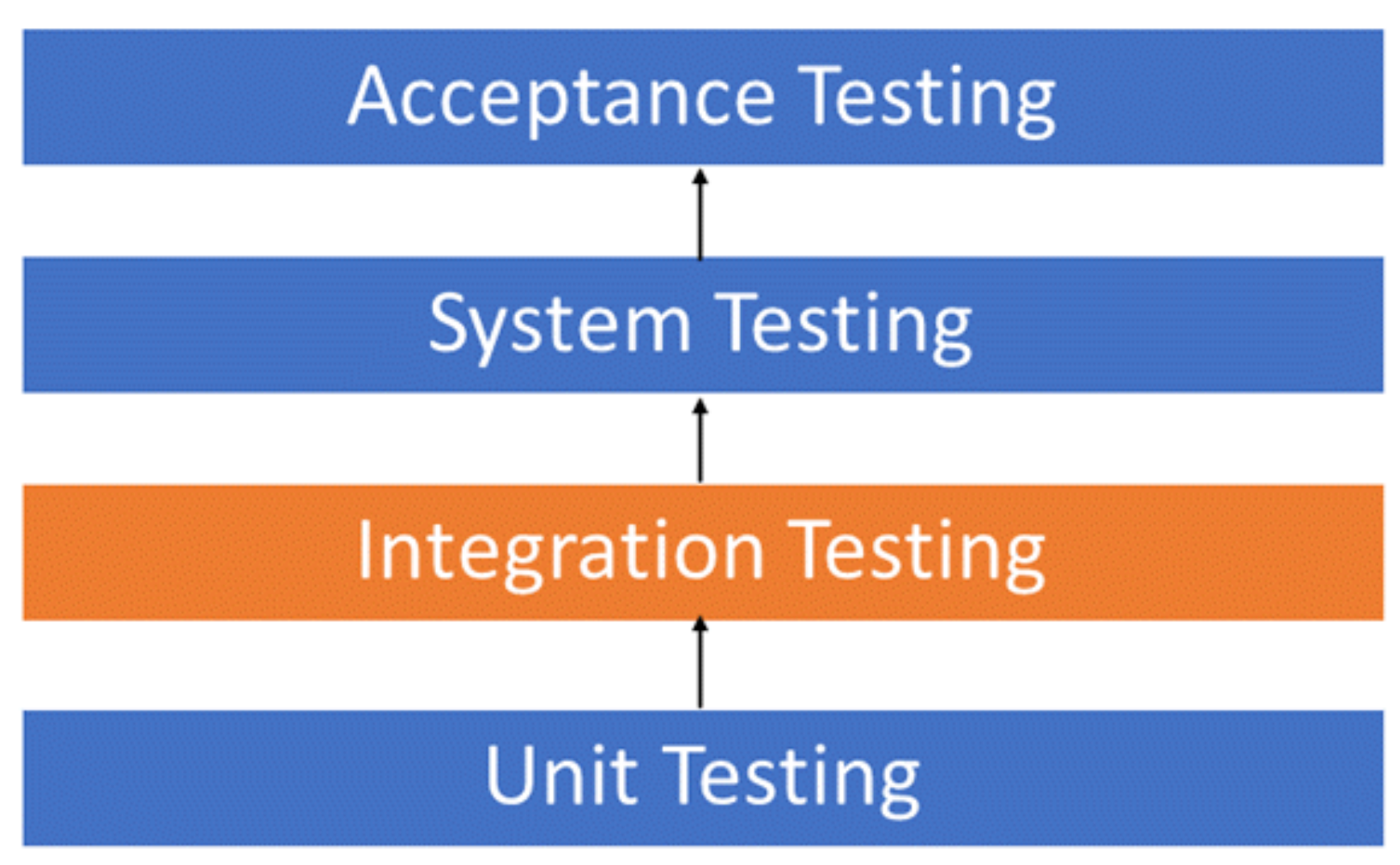
Unit Testing Tools

- Junit: Junit is a free to use testing tool used for Java programming language. It provides assertions to identify test method. This tool test data first and then inserted in the piece of code.
- NUnit: NUnit is widely used unit-testing framework use for all .net languages. It is an open source tool which allows writing scripts manually. It supports data-driven tests which can run in parallel.
- JMockit: JMockit is open source Unit testing tool. It is a code coverage tool with line and path metrics. It allows mocking API with recording and verification syntax. This tool offers Line coverage, Path Coverage, and Data Coverage.
- EMMA: EMMA is an open-source toolkit for analyzing and reporting code written in Java language. Emma support coverage types like method, line, basic block. It is Java-based so it is without external library dependencies and can access the source code.
- PHPUnit: PHPUnit is a unit testing tool for PHP programmer. It takes small portions of code which is called units and test each of them separately. The tool also allows developers to use pre-define assertion methods to assert that a system behave in a certain manner.

Java Script Unit testing tools

- MochaJS is the most popular testing framework that supports backend and frontend testing.
- Jasmine is a user-behavior mimicker that allows you to perform test cases similar to user behavior on your website.
- AVA is a minimalistic light-weight testing framework that leverages asynchronous nature of Javascript.
- JEST is one of the most popular frameworks that is maintained regularly by Facebook
- Karma is a productive testing environment that supports all the popular test description framework within itself
- Tape is pretty similar to AVA in its architecture. It does not support globals, and hence you need to include Tape in each test file

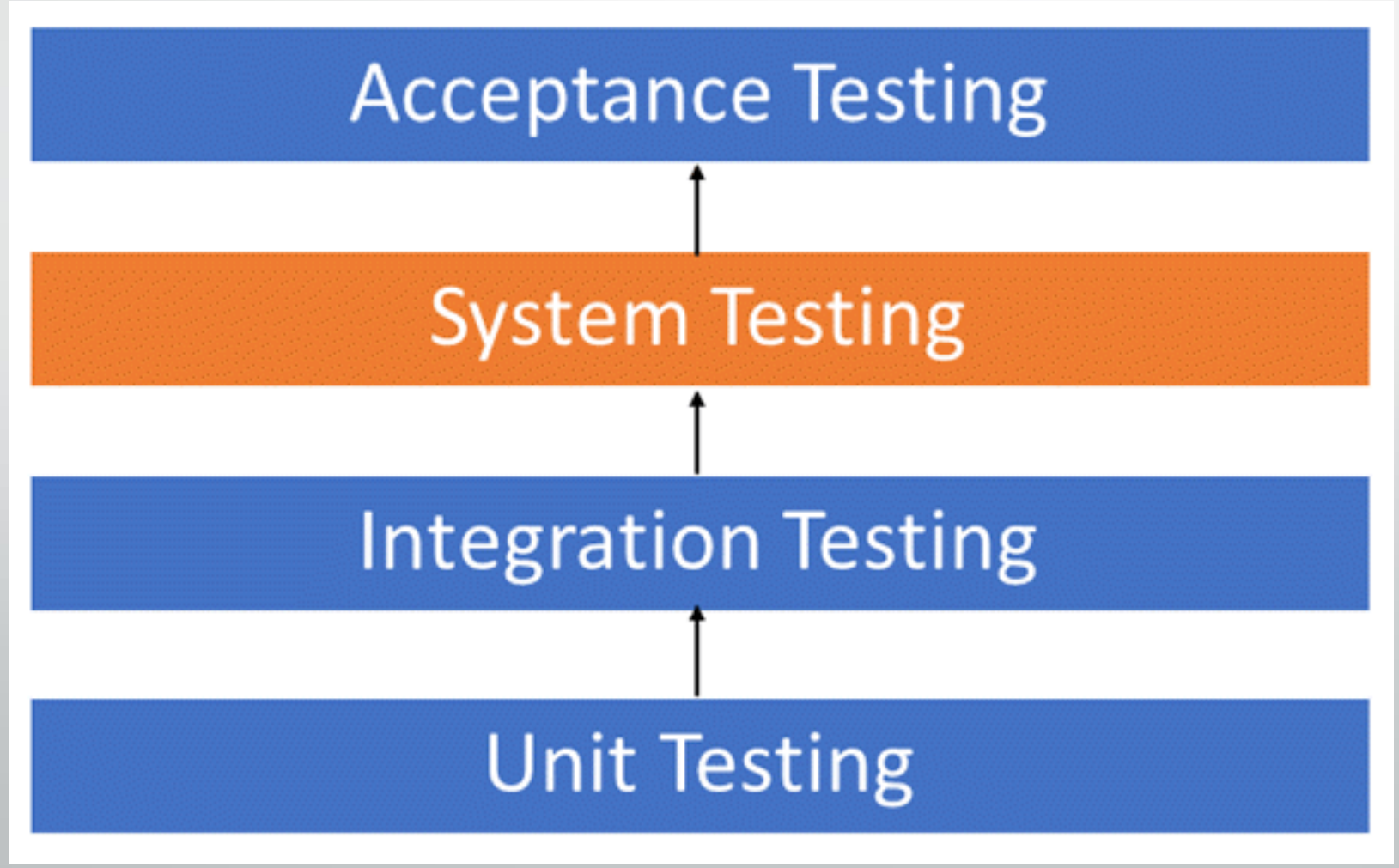
Integration Testing



What is Integration Testing?

- **INTEGRATION TESTING** is defined as a type of testing where software modules are integrated logically and tested as a group.
- A typical software project consists of multiple software modules, coded by different programmers.
- The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated
- Integration Testing focuses on checking data communication amongst these modules.

System Testing



What is System Testing?

- SYSTEM TESTING is a level of testing that validates the complete and fully integrated software product.
- The purpose of a system test is to evaluate the end-to-end system specifications.
- Usually, the software is only one element of a larger computer-based system.
- Ultimately, the software is interfaced with other software/hardware systems.
- System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

Acceptance Testing

- **User Acceptance Testing (UAT)** is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment.
- UAT is done in the final phase of testing after functional, integration and system testing is done.
- The main purpose of UAT is to validate the end to end business flow. It does NOT focus on Cosmetic errors, Spelling mistakes or System testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is a kind of black box testing where two or more end-users will be involved. The Full Form of UAT is User Acceptance Testing.

JavaScript unit testing with Mocha

- Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser.
- To install mocha run "npm install mocha chai --save-dev".
- Mocha is the library that allows us to run tests.
- Chai contains some helpful functions that we'll use to verify our test results.

Demo

In index.js file we create a function and export it to be seen in the test file

```
module.exports={
    addNumbers(x,y){
        return x+y
    }
}
```

Index.js

In test/index.js file import the chai assert library and the index.js file wiche contain the code we will test

```
const assert = require('chai').assert;
const index = require('../index.js');
```

Test/Index.js

```
describe('index.js', () => {
    it('adding numbers worke', ()=>{
        assert.equal(index.addNumbers(2,2),4)
    })
})
```

- ***describe*** is a function which holds the collection of tests. It takes two parameters, first one is the meaningful name to functionality under test and second one is the function which contains one or multiple tests. We can have nested *describe* as well.
- ***it*** is a function again which is actually a test itself and takes two parameters, first parameter is name to the test and second parameter is function which holds the body of the test.
- ***assert*** helps to determine the status of the test, it determines failure of the test.