Software Engineering Concepts

Lecture 16

Review of basic Testing Concepts

Chapter 3: Testing in Software Life Cycle

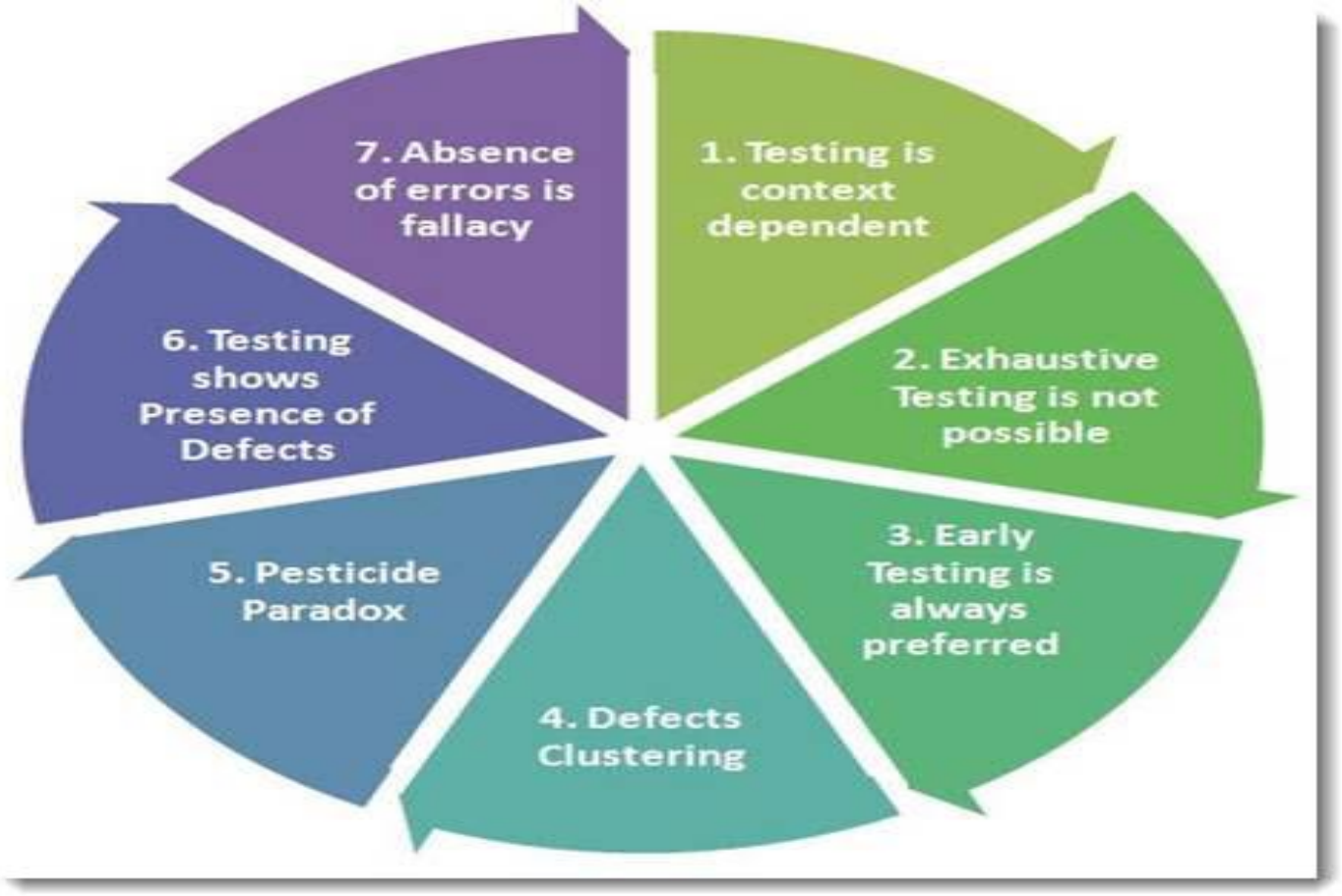
Agenda for today

- Remaining Levels of Testing
- System Testing
- Different Types of System Testing
- Acceptance Testing
- Acceptance Testing different Forms
- Generic Types of Testing
- Functional Testing
- Non-functional Testing
- Testing of Software Structure
- Testing related to changes and regression testing

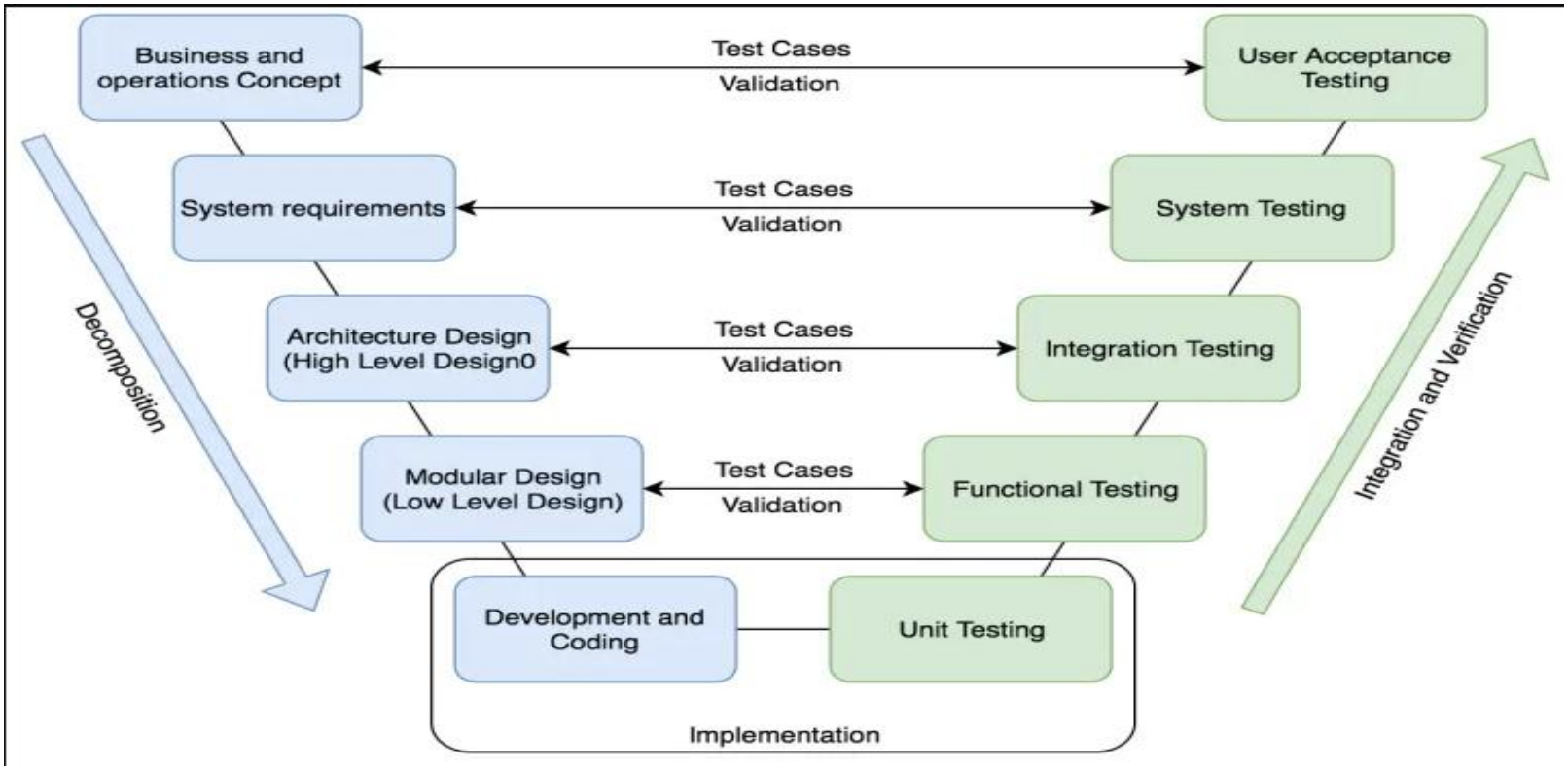
Software Testing Principles

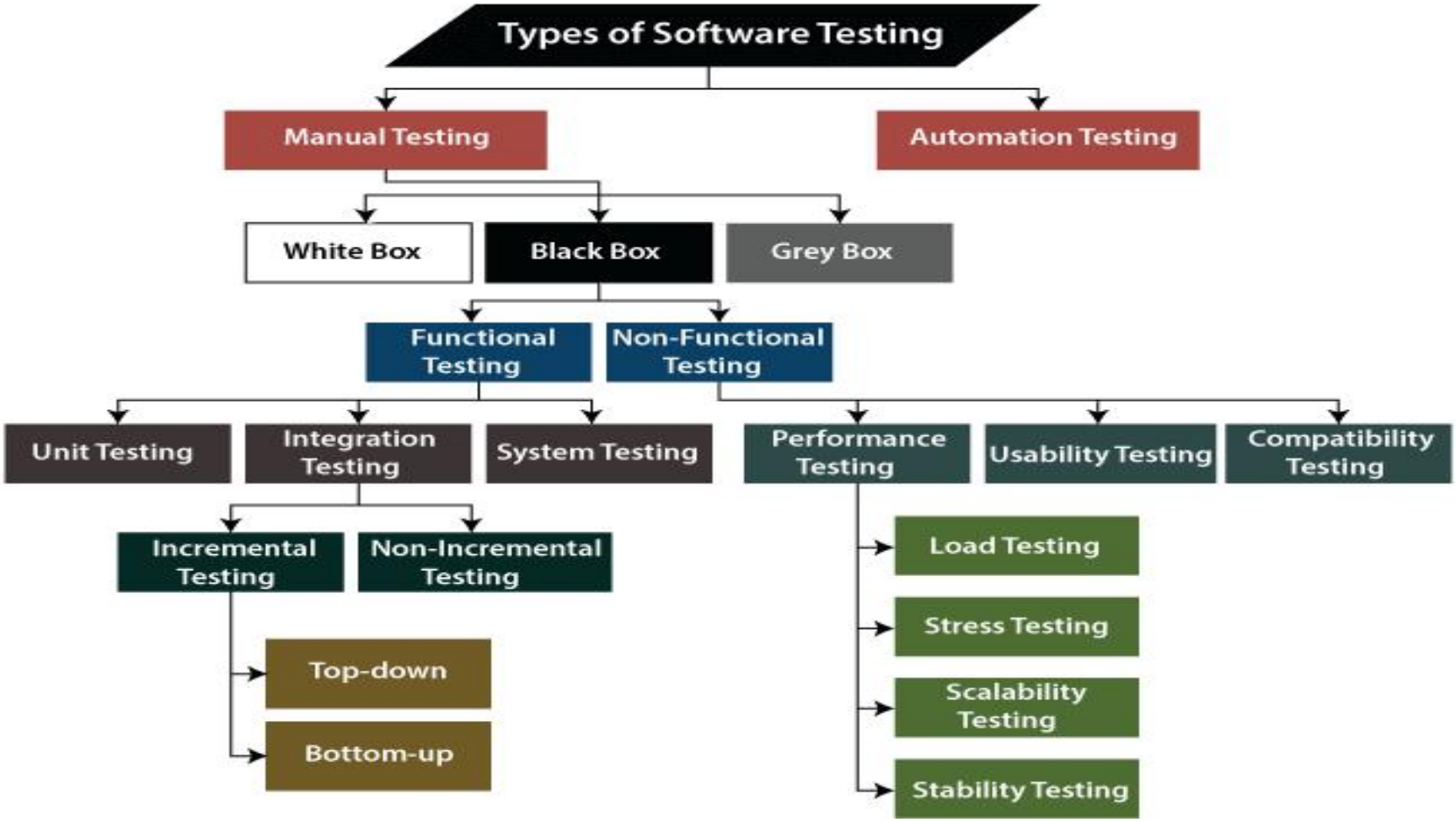
- Testing shows the presence of defects.
- Exhaustive testing is not possible.
- Early testing.
- Defect clustering.
- Pesticide paradox.
- Testing is context-dependent.
- Absence of errors fallacy.

General Principles of Testing

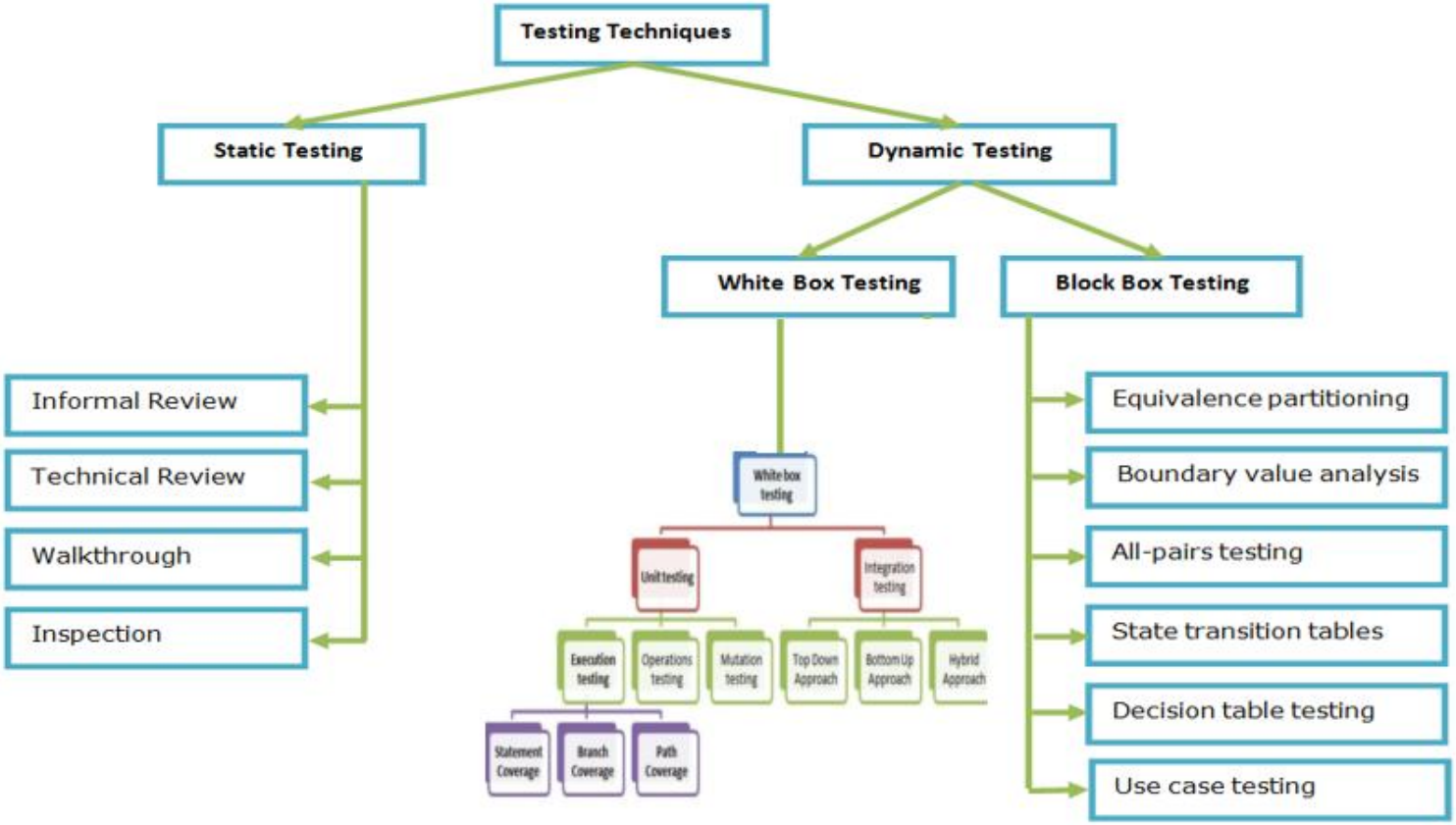


V-Model


Business and operations Concept
Test Cases
Validation
User Acceptance Testing
System requirements
Test Cases
Validation
System Testing
Architecture Design (High Level Design0
Test Cases
Validation
Integration Testing
Modular Design (Low Level Design)
Test Cases
Validation
Functional Testing
Development and Coding
Unit Testing
Implementation
Decomposition
Integration and Verification


Types of Software Testing
Manual Testing
Automation Testing
White Box
Black Box
Grey Box
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performance Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Top-down
Bottom-up
Load Testing
Stress Testing
Scalability Testing
Stability Testing

Testing Techniques - Types



Testing Documentation

- Test Plan/Strategy Document
- Test Policy Document
- Check Lists
- Quality Plan
  - A document or set of documents that describe the standards, quality practices, resources and processes pertinent to a specific product, service or project.
- Test Scenarios Document
  - Document containing the possible outcomes of the system under test
- Test Case Document
- Defect Report Document
- Testing Metrics Document
  - Requirement Traceability Matrix
  - Development Traceability Matrix
- Test Summary Report

Test Plan/Strategy Document

The test plan describes the testing process in terms of the features to be tested, pass/fail criteria and testing approach, resource requirements and schedules.

- Introduction
- Test items
- Features to be tested
- Testing approach
- Item pass/fail criteria
- Suspension and resumption
- Deliverables
- Tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Costs and schedule
- Risks and contingencies

Testing Life Cycle

Establish test objectives.

Design criteria (review criteria).

- Correct
- Feasible
- Coverage
- Demonstrate functionality

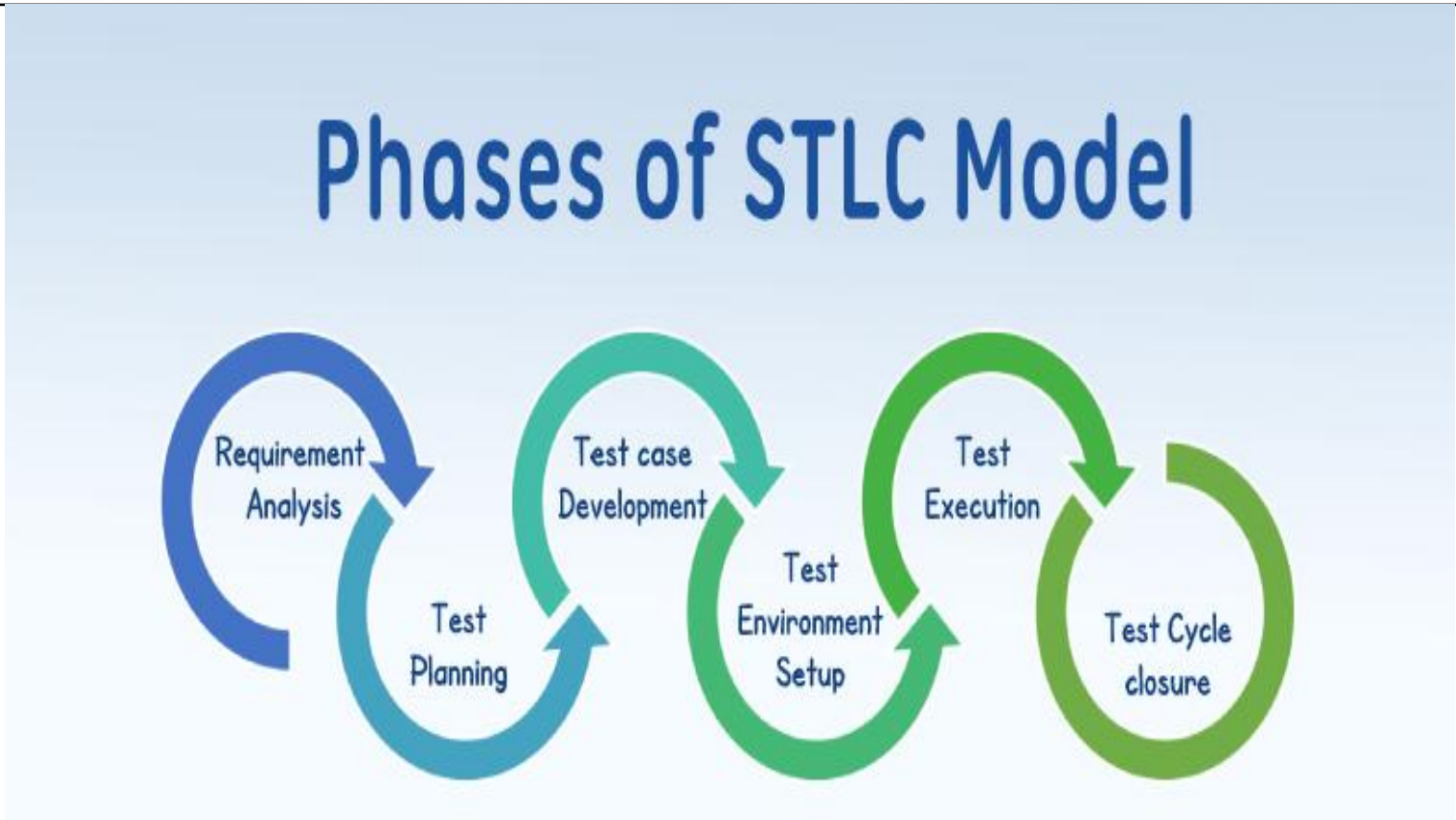
Writing test cases.

Testing/Checking test cases.

Execute test cases.

Evaluate test results.

Software Testing Life Cycle



Why we write test cases?

The basic objective of writing test cases is to validate the testing coverage of the application.

Test Coverage : Test coverage measures the amount of testing performed by a set of test. It will include gathering information about which parts of a program are actually executed.

Test coverage is a measure of testing and to have better test coverage it is required to write all the tested cases for identified scenario. Ideally the following process will be in action.

Customer will give requirement to developer.

Test engineer has to spend enough time to understand the requirement

Test engineer has to identify all possible scenarios

Test engineer has to write test cases for all identified scenarios.

Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects.

There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc. **Following are the attributes of a test case;**

- Test case ID.
- Product Module
- Product version
- Revision history
- Purpose
- Assumptions
- Pre-Conditions
- Steps.
- Expected Outcome
- Actual Outcome
- Post Conditions

Test Case Format

Test case ID:	The Unique_ID of the test case
Test case description:	The objective or summary of the test case
Preconditions:	Any preconditions which need to be executed before starting the test
Test steps:	Procedure to execute the test.
Test data:	Data required while executing the test
Expected Result:	The expected output of the test
Actual Result:	The actual output of the test
Test Case Status:	Pass, Fail, 'Not executed' when test case is not executed and 'Blocked' when high severity bug is found
Created By:	Name of the person who wrote the test case
Date of creation:	The date on which the test case was authored
Executed By:	Name of the person who ran or executed the test case
Date of execution:	The date on which the test case was executed

Test Case Example's

Test Case ID	TC001
Test Case Name	To Verify the "Login" of Gmail account
Preconditions	1.User is authorized. 2. Has an account in Gmail
Test Steps	1.Enter Valid username 2.Enter valid password 3.Click on "Login" button
Test Data	xyz123@gmail.com
Expected Result	1.User should be able to login his Gmail account with his valid credentials 2. "Invalid username or password" should get displayed if the username and password are not valid
Actual Result	1.If the valid credentials are entered then the user will be able to login his/her account 2.If the invalid credentials are entered then nothing happens (the expected message not displayed)
Status	Fail
Created By	John
Date of Creation	01/01/2018
Executed By	Jane
Date of Execution	02/10/2018
Test Environment	•OS: Windows Y •Browser: Chrome N

Test Case Example's

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TC01	Check Customer Login with valid Data	1.Go to site http://faculty.comsats.edu.pk 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = test123 Password = pass123	User should Login into application	As Expected	Pass
TC02	Check Customer Login with invalid Data	1.Go to site http://faculty.comsats.edu.pk 2.Enter UserId 3.Enter Password 4.Click Submit	Userid = test123 Password = pass123	User should not Login into application	As Expected	Pass

Testing Artifacts

Traceability Matrix

- A traceability matrix is a table that correlates requirements or design documents to test documents.

Test Case

- A test case normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result.

Test Script

- A test script is a procedure, or programing code that replicates user actions.

Test Suite

- The most common term for a collection of test cases is a test suite. The test suite often also contains more detailed instructions or goals for each collection of test cases.

Test Fixture

- or test data In most cases, multiple sets of values or data are used to test the same functionality of a particular feature.

Test Harness

- The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

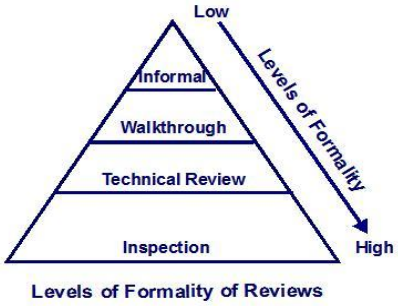
Testing Meetings & Discussions

Reviews

Walkthrough

Technical Review

Inspection



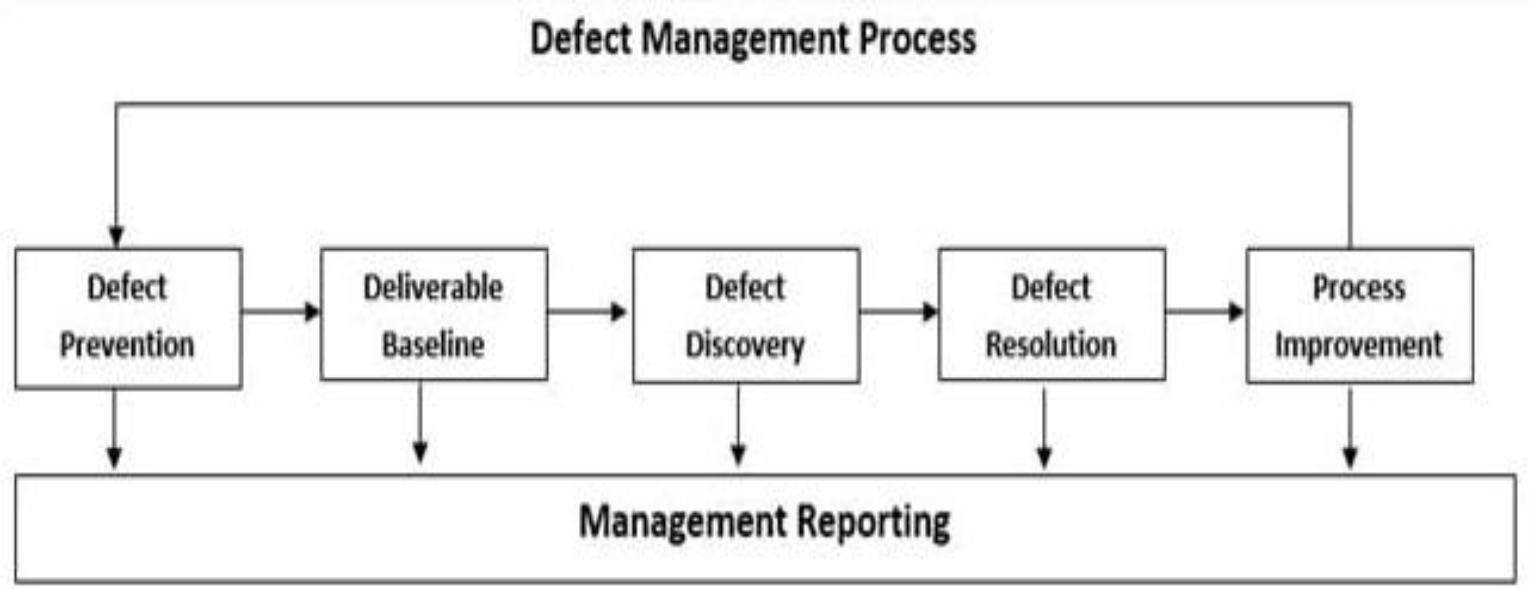
Retrospect Meeting

- The retrospective meeting occurs on the last day of the sprint, after the sprint review meeting. In this meeting, your team inspects and explores how it has been working within the Scrum processes.

Kick Off Meeting

- a "**kick off meeting**" would be the first in a series of meetings or a meeting introducing an event that is going to take place over time

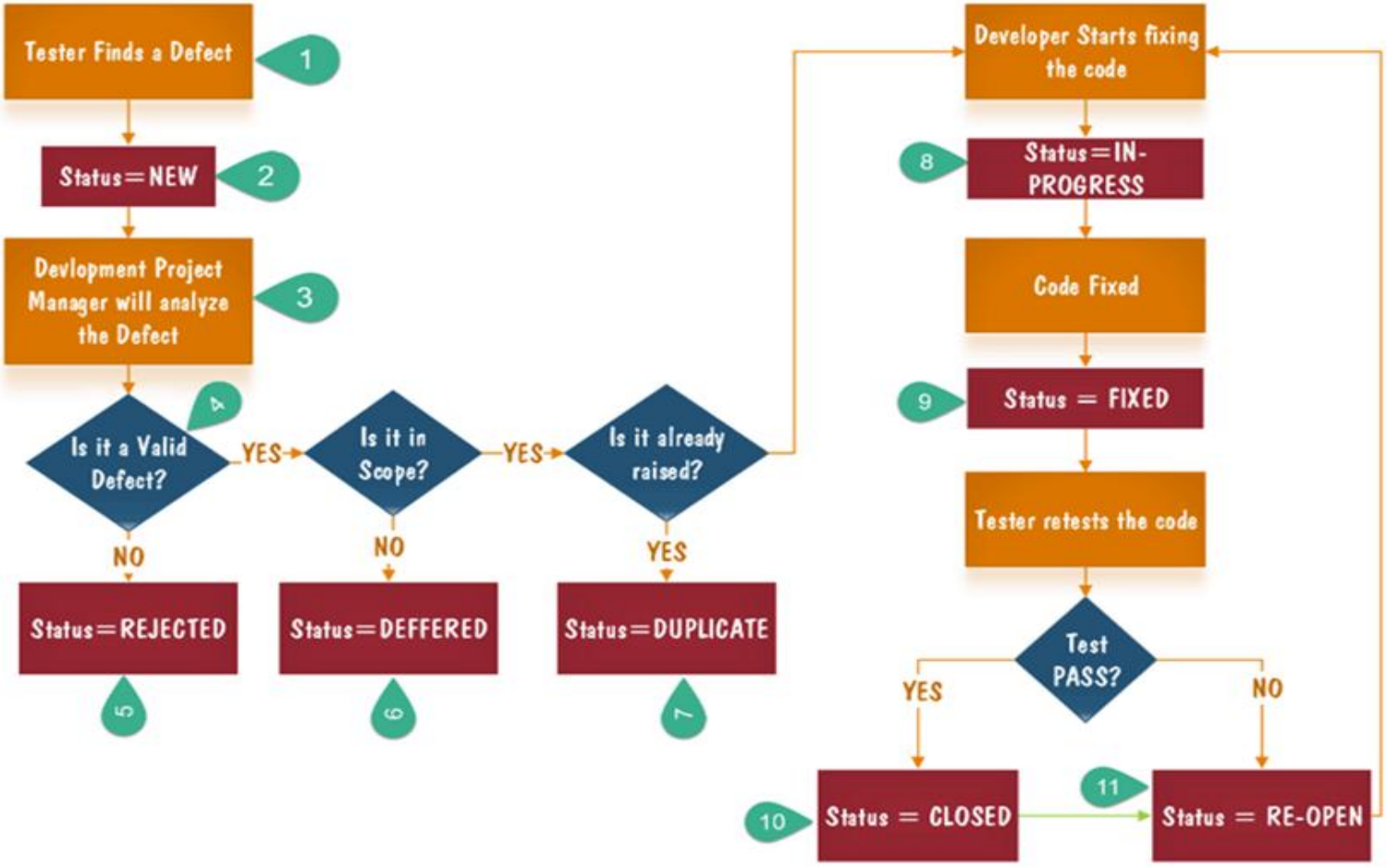
Defect Management Process in Software Testing



Attributes of a Defect (Bug)

1. Subject/Title
2. Description about Bug/Defect
3. Steps to reproduce
4. Type (Functionality, UI, etc)
5. **Priority (Low, Normal, High, Urgent, Immediate)**
6. **Severity (System Failure, Logical, Work round, Cosmetic, Suggestion)**
7. Product (Admin/User)
8. Component (User Management → Add User)
9. Version Occurred In
10. Version Fixed In
11. Reproducible (Always, sometimes, etc)
12. Operating System
13. Browser
14. Assignee
15. Attachment
16. Due Date

Defect/Bug Life Cycle in Testing



Manual Testing

- What is Manual Testing
- Process of Manual Testing.
- Benefits of Manual Testing.
- Drawbacks of Manual Testing.
- Tools for Manual Testing.

AUTOMATION TESTING

- What is Test Automation
- Process of Test Automation
- Benefits of Test Automation

Automated Vs. Manual Testing

ADVANTAGES	
Automated Testing	**Manual Testing**
• If you have to run a set of tests repeatedly automation is a huge gain	• If Test Cases have to be run a small number of times it's more likely to perform manual testing
• Helps performing "compatibility testing" - testing the software on different configurations	• It allows the tester to perform more ad-hoc (random testing)
• It gives you the ability to run automation scenariosto perform regressions in a shorter time	• Short term costs are reduced
• It gives you the ability to run regressions on a code that is continously changing	• The more time tester spends testing a module the grater the odds to find real user bugs
• Can be run simultaneously on different machines thus decreasing testing time	
• Long term costs are reduced	
DISADVANTAGES	
Automated Testing	**Manual Testing**
• It's more expensive to automate. Initial investments are bigger than manual testing	• Manual tests can be very time consuming
• You cannot automate everything, some tests still have to be done manually	• For every release you must rerun the same set of tests which can be tiresome
OTHER FACTORS	
• The performance of yout test tools	
• The knowledge level of your testing team	
• The continous growth of software to be tested	
• Number of necessary regressions	

Drawbacks of Test Automation.

Disadvantages of Automation Testing:

- Proficiency is required to write the automation test scripts.
- Debugging the test script is major issue. ...
- Test maintenance is costly in case of playback methods. ...
- Maintenance of test data files is difficult, if the test script tests more screens.

Tools for Test Automation.

Here's a list of the top automation testing tools :

- Selenium.
- Appium.
- Katalon Studio.
- Cucumber.
- HPE Unified Functional Testing (UFT)
- SoapUI.
- TestComplete.
- Worksoft.

Unit Testing

- Programming Languages for Unit Testing
- Types of Unit Testing

Tools for Unit Testing.

Here is the list of top Unit Testing Framework/Tools used to create accurate unit tests:

#1) NUnit
#2) JMockit
#3) Emma
#4) Quilt HTTP
#5) HtmlUnit
#6) Embunit
#7) SimpleTest
#8) ABAP Unit
#9) Typemock
#10) LDRA
#11) Microsoft unit testing Framework
#12) Unity Test Tools
#13) Cantata
#14) Karma
#15) Jasmine
#16) Mocha
#17) Parasoft
#18) JUnit
#19) TestNG
#20) JTest

What is Unit Testing.

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. The isolated part of the definition is important.

Integration Testing

- What Is Integration Testing.

Integration testing is performed using the black box method. This method implies that **a testing team interacts with an app and its units via the user interface – by clicking on buttons and links, scrolling, swiping, etc**. They don't need to know how code works or consider the backend part of the components.

Types of Integration Testing

- Big-Bang Integration Testing
- Bottom-Up Integration Testing
- Top-Down Integration Testing
- Mixed Integration Testing

Top Down & Bottom Up
Examples of Integration Testing

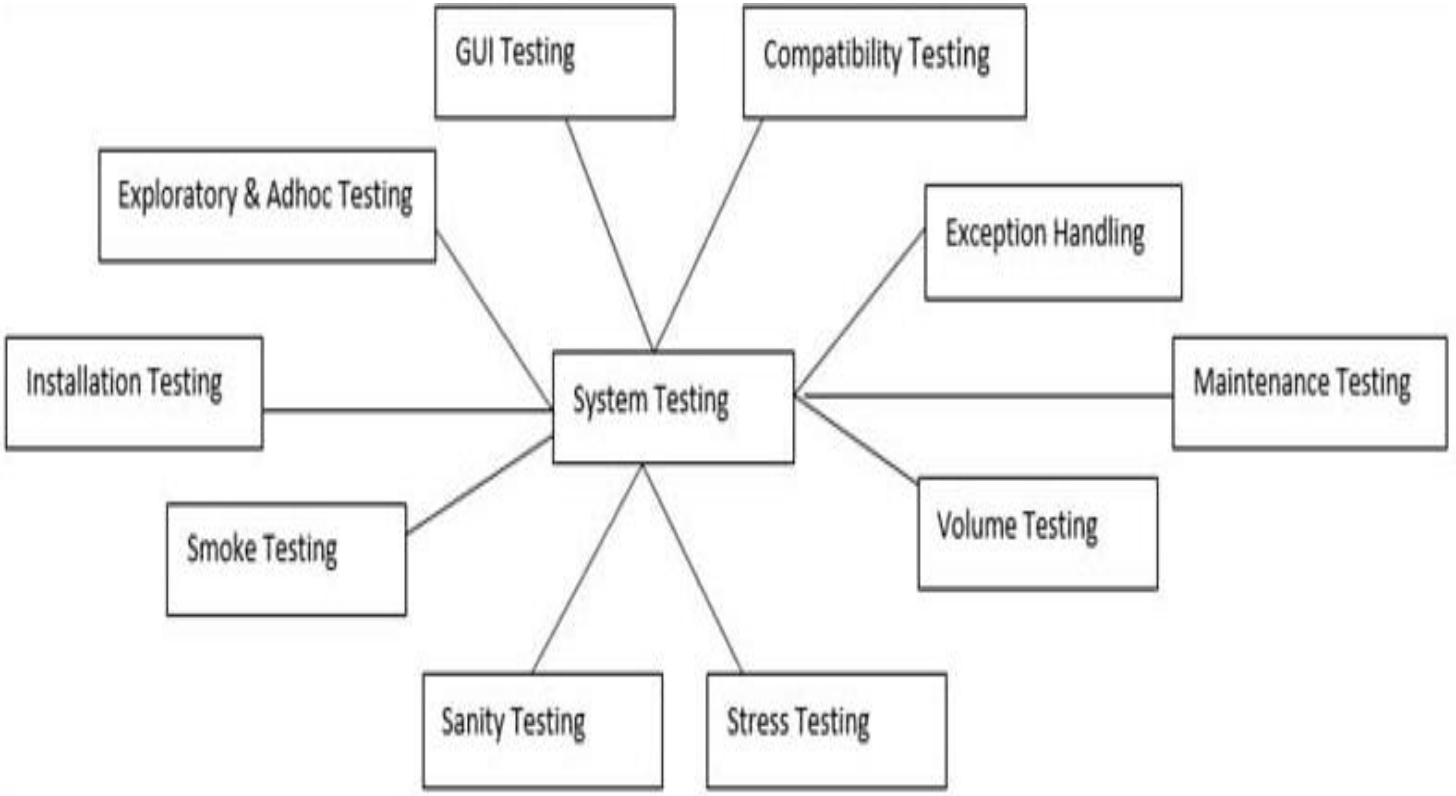
Top-down integration testing is **an integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated**.

Bottom Up

In Bottom Up Integration testing approach different modules are created first then these modules are integrated with the main function

System Testing

- **Types of System Testing**



Definition of System Testing

System testing is defined as **testing of a complete and fully integrated software product**. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team

for example, might check that every kind of user input produces the intended output across the application.

Examples of System Testing

System Testing Example is very simple one because the name system itself gives the answer that in System Testing we have to test the complete system

Sanity Testing Vs Smoke Testing

Smoke Testing	Sanity Testing
Whenever a new feature comes into the picture, we perform Smoke Testing.	Whenever Code changes or Bug fixes comes into the picture, we perform Sanity Testing.
It is a part of Acceptance testing	It is a part of Regression testing.
It is shallow and wide.	It is deep and narrow.
Smoke testing is documented.	Sanity testing is not documented.
Smoke Testing is performed mostly in unstable builds.	Sanity Testing is performed in stable builds only.
Smoke Testing can be performed either manually or by using Automation tools.	Sanity Testing is commonly executed manually, not by using Automation approach.

Regression Testing Vs Retesting Testing

Regression Testing	Re-Testing
Regression testing is to ensure that changes have not affected unchanged part.	Retesting is done to make sure that the tests cases which failed in last execution are passed after the defects are fixed.
Regression testing is not carried out for specific defect fixes.	Retesting is carried out based on the defect fixes.
In Regression testing, the test cases which passed earlier can be included to check the functionality which was working earlier.	In Retesting, the cases which are failed earlier can be included to check if the functionality failure in an earlier build.
Regression test cases are derived from the functional specification, the user manuals, user tutorials, and defect reports in relation to corrected problems.	Test cases for Retesting cannot be prepared before start testing. In Retesting, test cases that are failed in the prior execution are only re-executed.
Automation is the key for regression testing. Manual regression testing tends to get more expensive with each new release. Automation always complements the regression test process.	Test cases for re-testing cannot be automated due to uncertainty
Defect verification doesn't fall under Regression testing.	Defect verification is coming under Retesting.
Based on the resource availability the Regression testing can be carried out in parallel with Retesting.	Priority of Retesting over Regression testing is higher, so it is carried out before regression testing.

Functional Testing

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Types of non Functional Testing.

- Unit Testing.
- Component Testing.
- Smoke Testing.
- Integration Testing.
- Regression Testing.
- Sanity Testing.
- System Testing.
- User Acceptance Testing.

Non Functional Testing

Non-functional testing is the testing of a software application or system for its non-functional requirements: the way a system operates, rather than specific behaviours of that system.

Types of non Functional Testing.

- Performance Tests.
- Load Tests.
- Stress Tests.
- Volume Tests.
- Security Tests.
- Upgrade & Installation Tests.
- Recovery Tests.

- **TYPES OF FUNCTIONAL TESTING ARE**
  - UNIT TESTING
  - SMOKE TESTING
  - SANITY TESTING
  - INTEGRATION TESTING
  - WHITE BOX TESTING
  - BLACK BOX TESTING
  - USER ACCEPTANCE TESTING
  - REGRESSION TESTING

  TYPES OF NON FUNCTIONAL TESTING ARE
  - PERFORMANCE TESTING
  - LOAD TESTING
  - VOLUME TESTING
  - STRESS TESTING
  - SECURITY TESTING
  - INSTALLATION TESTING
  - PENETRATION TESTING
  - COMPATIBILITY TESTING

TEST-TO-PASS AND TEST-TO-FAIL

Test-to-pass:

- assures that the software minimally works,
- does not push the capabilities of the software,
- applies simple and straightforward test cases,
- does not try to “break” the program.

Test-to-fail:

- designing and running test cases with the sole purpose of breaking the software. - try to “break” the program.
- strategically chosen test cases to probe for common weaknesses in the software.

Questions and Answers


Thank
you