Kirirom Institute of Technology

Unit 5

Software Testing

What is testing?

Is a method :

- to check whether the actual SW product matches requirement
- check if SW is defect free

- Involves manual/automated tools
- Identify errors, gaps and missing requirements in contrast to actual requirements
- Can be done manually or with some automated tools

Why testing is important?

- Today's technology is controlled by some sort of Software

Why testing is important?

- The success of SW app has impact on Business & growth

Why testing is important?

- Early identification of bugs before delivering SW product saves time, money and effort

- Also ensures reliability, security and High performance

Why testing is important?

- Customer satisfaction

Why testing is important?

- SW bugs could be expensive even dangerous

- Bugs in SW caused many effects like plane crash to rocket failures even caused death to people

Who does Testing?

- Big companies recruit testers

- SW Testers, Developers, Project lead, Team manager and end users are also part of testing

- Testing can be applied to different phases of SDLC

Benefits of SW Testing:

- Save money
- Ensuring security
- Product quality
- Customer satisfaction

Principles of Testing:

“Testing is an imaginative and intellectual task”

There 7 principles:

1. Testing shows the presence of defects
2. Exhaustive Testing is not possible
3. Early Testing
4. Defect Clustering
5. Pesticide Paradox
6. Testing is context-dependent
7. Absence of errors fallacy

1. Testing shows the presence of defects

- Testing can minimize/reduce the number of bugs
- Testing can't prove a SW is error free
- Few bugs are identified after deployment

2. Exhaustive Testing is not possible

- It is not always possible to test the system with all possible combinations of inputs
- Exhaustive testing is unnecessary (Waste of hard work)
- Exhaustive testing may cause issue with the product timeline

3. Early Testing

- Testing must start from the earlier stages
- Requirement analysis, High Level Design, Low level design stages itself
- Identifying and fixing bugs in the earlier stages will save cost and effort
- Requirement Specification are the base for testing
- Proper requirement = Easy fixing

4. Defect clustering:

- Most of the bugs in SW are from few modules
- 80 - 20 rule (80% complications are from 20% of modules)
- Checking those 20% module with the same test cases will not identify bugs

5. Pesticide Paradox

Kirirom Institute of Technology

- Think about mutation of the virus
- Pests(Insects) are attaining evolution when the same medicine is used
- Same is the case of test cases and bugs
- Using same test cases for a longer period will not identify bugs
- To avoid : Test cases must be updated

6. Testing is context-dependent

- Testing differs based on the context of the application
- Testing a web app differs from testing a mobile app

7. Absence of errors fallacy

- Error free application is not possible
- Many possible ways of bugs still unidentified
- An app tested and didn’t have error doesn’t mean the app is error free

Most popular model:

- V-Model
- In other words verification and validation model

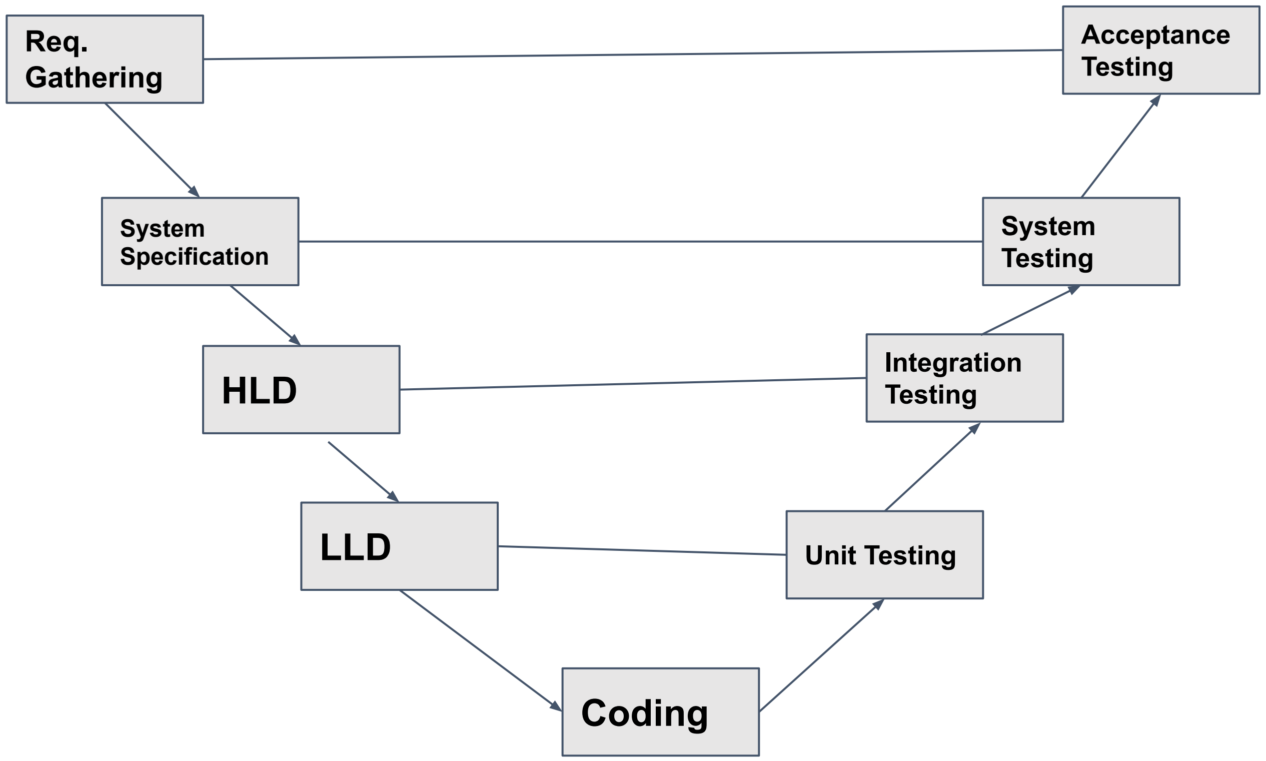
- **Verification:**

  Verification - Static analysis (We don’t test the code eg. Inspections, reviews etc)

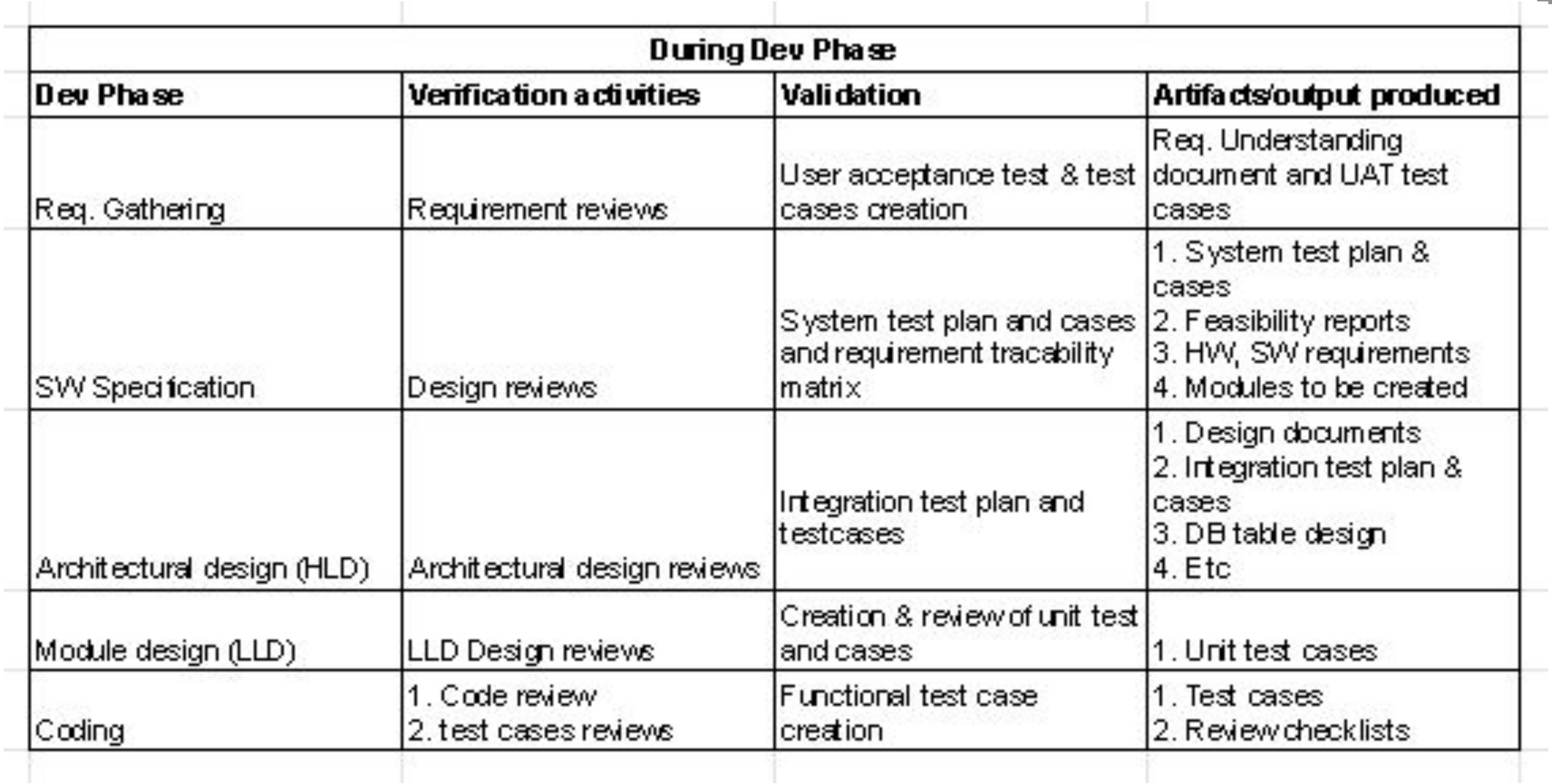
  Validation - Dynamic analysis (Testing with executing the code - Black box, Whitebox and Greybox)

  Both verification and validation are complementing each other

The V-Model:

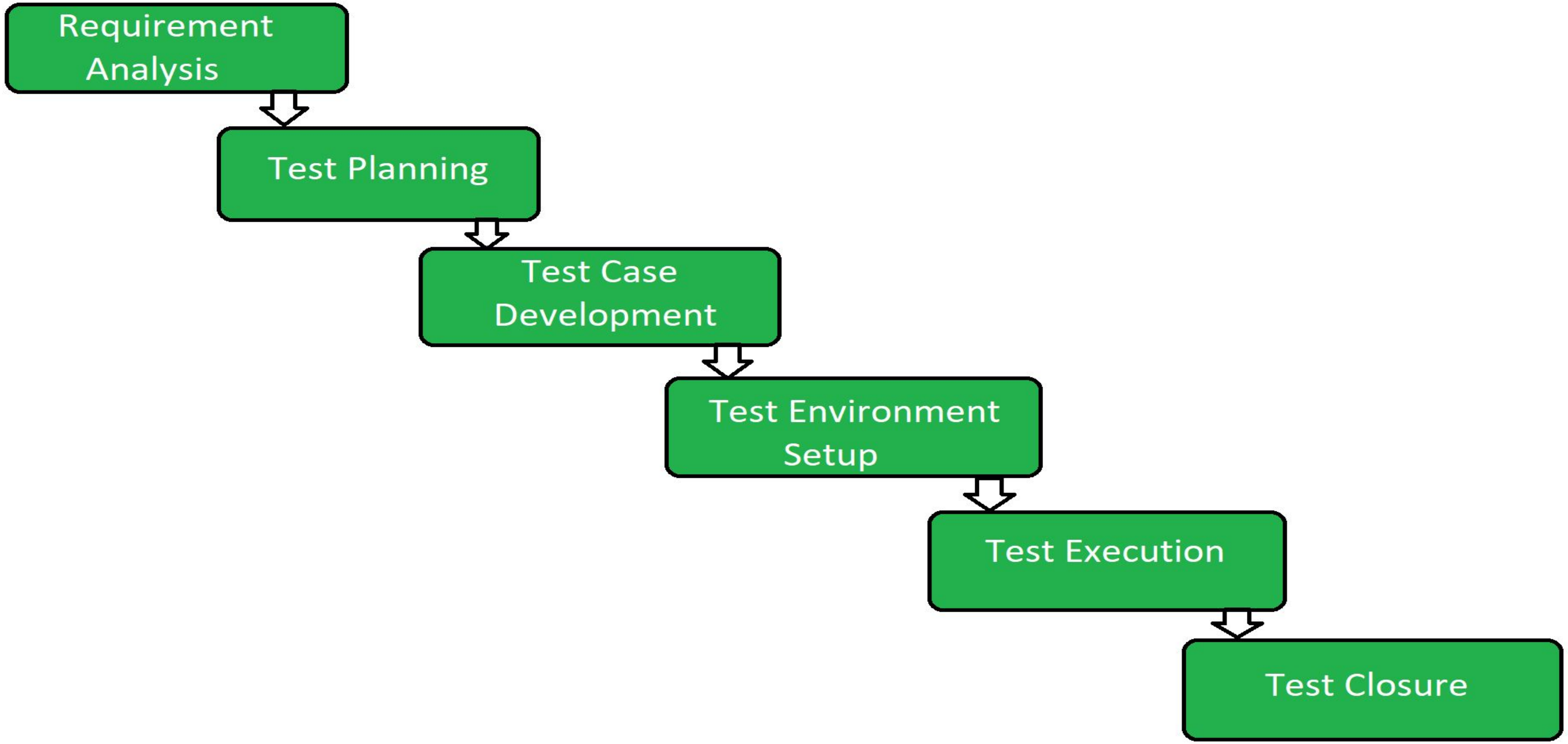

Req. Gathering
Acceptance Testing
System Specification
System Testing
HLD
Integration Testing
LLD
Unit Testing
Coding

V-Model:



During Dev Phase			
Dev Phase	**Verification activities**	**Validation**	**Artifacts/output produced**
Req. Gathering	Requirement reviews	User acceptance test & test cases creation	Req. Understanding document and UAT test cases
SW Specification	Design reviews	System test plan and cases and requirement tracability matrix	1. System test plan & cases 2. Feasibility reports 3. HW, SW requirements 4. Modules to be created
Architectural design (HLD)	Architectural design reviews	Integration test plan and testcases	1. Design documents 2. Integration test plan & cases 3. DB table design 4. Etc
Module design (LLD)	LLD Design reviews	Creation & review of unit test and cases	1. Unit test cases
Coding	1. Code review 2. test cases reviews	Functional test case creation	1. Test cases 2. Review checklists

STLC:


Requirement Analysis
Test Planning
Test Case Development
Test Environment Setup
Test Execution
Test Closure

Types of Testing:

Kirirom Institute of Technology

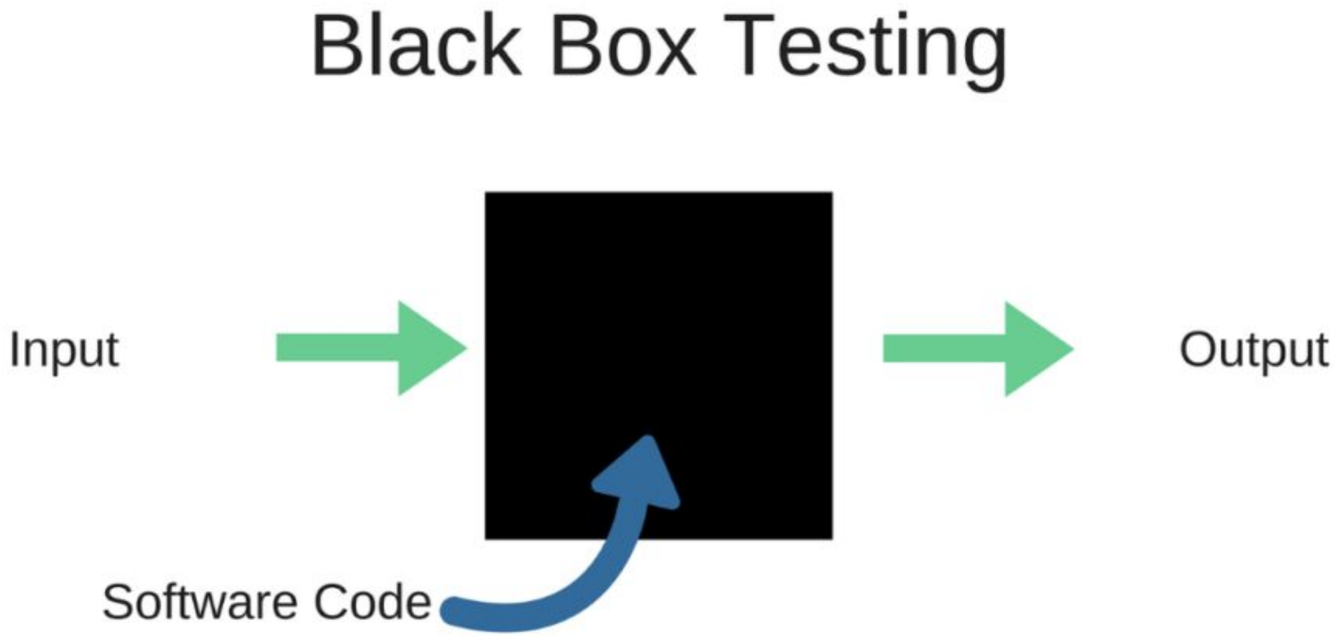
1. Automated Testing
   - Automation process of a manual process
   - Tester writes scripts and tools execute testing
   - Efficient

2. Manual Testing
   - Apps are testing done manually by QA testers
   - The app must be tested in different environments and results recorded
   - Delay in work, Boring, fatigue, prone to error etc

SW testing methods:

- Black box testing
- White box testing
- Gray box testing

Blackbox testing:



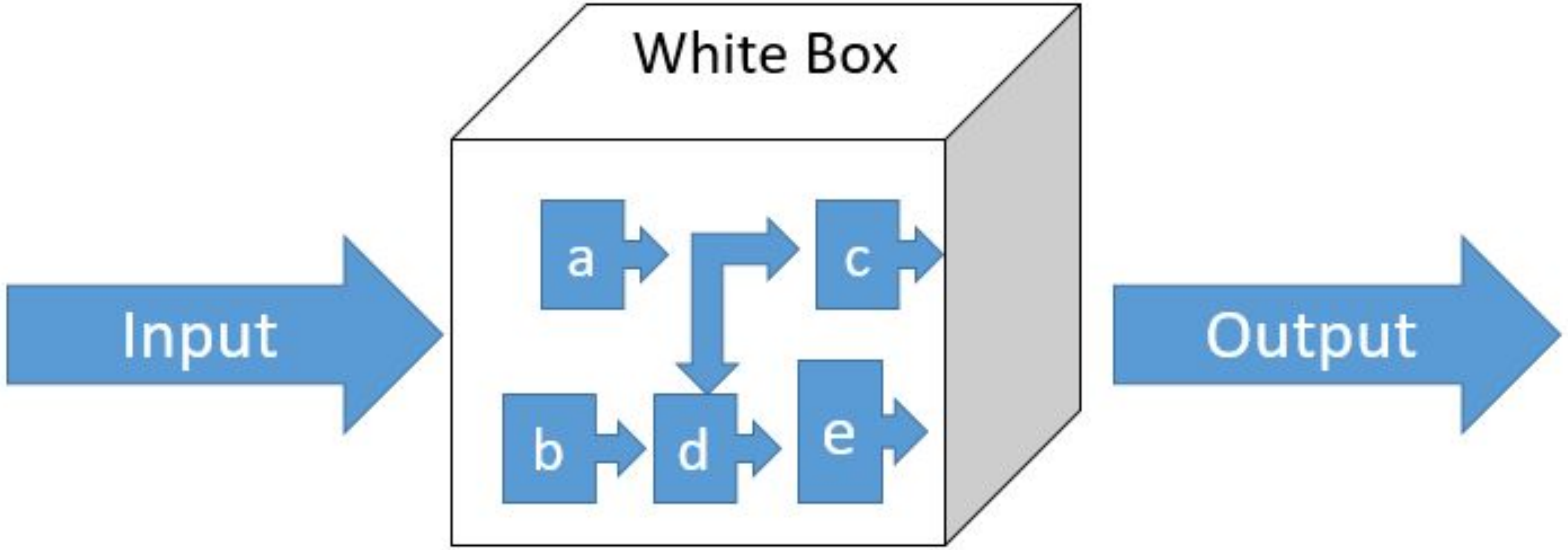
Blackbox testing:

- Behavioral testing
- No internal logics/implementations are tested
- Testing with some sample inputs, design Etc.,

What are tested?

- Accuracy of the system
- Speed
- Usability
- Performance
- Etc.,

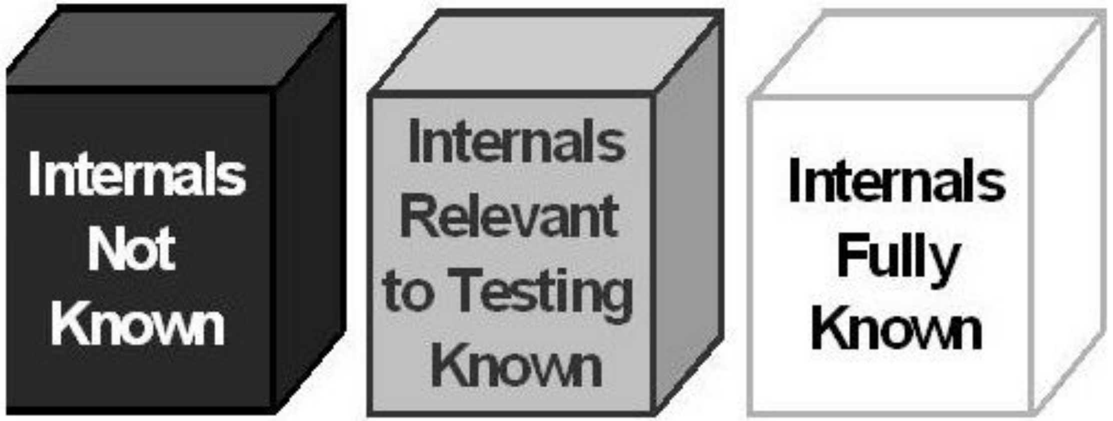
White box testing:



White box testing:

- Structure of the app is tested
- AKA structural / glassbox / clearbox testing etc
- Internal code and infrastructure of the SW
- Inner working of the SW is tested
- Done by developers and leave to testers to do blackbox testing
- Line by line code check, loop condition check etc are few examples
- **Unused libraries, data handling of routines, efficiency of control structures, Memory leaks, Security holes, variable usage, memory size and etc are tested**

Grey box Testing:



Grey box Testing:

- AKA translucent testing
- Tester with partial code knowledge
- Partial knowledge of internal structure
- Combination of Blackbox and whitebox
- Bridges the gap between Developers and testers
- User perspective

Functional Testing:

- Performed before Non functional testing
- Based on customer requirements
- Describes what the product does
- UT, UAT, Smoke, regression, integration testing

Functional Testing - Examples

Unit testing - Checking Source code

User acceptance testing - Done before production (Requirement vs product)

Smoke - Are major components of the system works properly (Done after new build)

Regression testing - To check if there is a bug after the addition of new features

Integration testing - To check if all the individual working components works well together

Non Functional Testing:

- Performed after functional
- Based on customer expectations
- How the product works
- Eg. Performance testing

Performance testing examples:

Load testing - How system performs when when peak load (CPU, mem usage etc) [Peak load is a highest load to the system in a day/week/month and a new applications it is 120% - 150% of the average load]

Stress testing - AKA torture testing, SW tested in extreme conditions

Documentation Artifacts:

Following documentations are maintained during test life cycle:

1. Test Plan
2. Test scenario
3. Test cases
4. Traceability Matrix

Test Plan:

- Answers the What questions
- High level testing objectives
- Scope of test
- What are not in scope of the test
- Risks
- Defining test closure

Test Scenario:

Kirirom Institute of Technology

- Identifying any functionality of the SW that can be tested
- Ensures complete test coverage
- Defines what are we going to test
- Example:

Test Scenario 1: Testing registration module of Shuttle bus booking system

Test Scenario 2: Testing the customer bus booking module

Test Cases:

- Low level
- Step by step of the test

Example (Based on our scenario):

- Register as a customer
- Register as a driver
- One time Password generation for registering

Click here to view a sample test case document

RTM - Requirement Traceability Matrix:

- Maps user requirements with test cases
- Captures all requirements in a single document
- Delivered at the conclusion of the SDLC
- Makes sure no functionality is untested

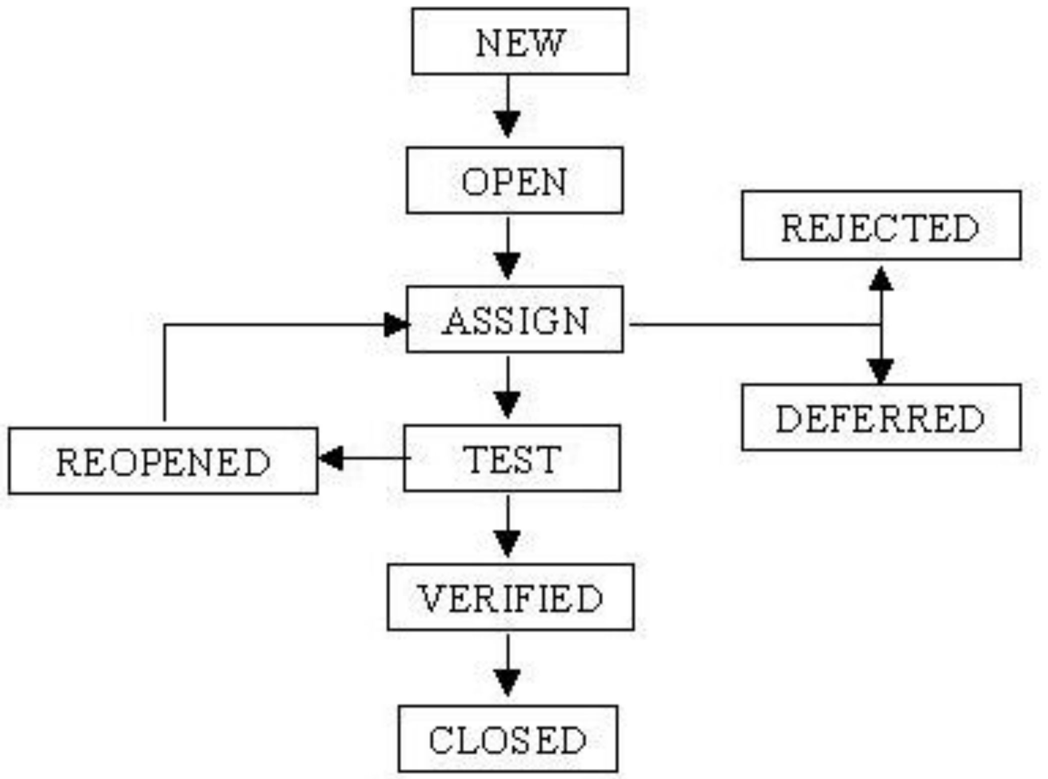
Defect Management:

- Process of identifying bugs and fixing them

  Steps:

1. Defect detection
2. Preparing Bug report
3. Bug fix
4. Bug list creation

Bug Life cycle:


NEW
OPEN
ASSIGN
REJECTED
DEFERRED
REOPENED
TEST
VERIFIED
CLOSED