KASPER
ANALYTICS

kasperanalytics.com

9123820085

MANUAL TESTING

hr@kasperanalytics.com

kasper-analytics

What is software?

- A Software is a collection of computer programs that helps us to perform a task.
- Types of Software:
- System software

Ex: Device drivers, OS, Servers, Utilities, etc.

- Programming software

Ex: compilers, debuggers, interpreters, etc.

- Application software

Ex: industrial automation, business software, games, telecoms, etc.

Product Vs Project

- If software application is developed for specific customer requirement then it is called Project.
- If software application is developed for multiple customers requirement then it called Product.

What is Software Testing?

- Software Testing is a part of software development process.
- Software Testing is an activity to detect and identify the defects in the software.
- The objective of testing is to release quality product to the client.

Why do we need testing?

- Ensure that software is bug free.
- Ensure that system meets customer requirements and software specifications.
- Ensure that system meets end user expectations.
- Fixing the bugs identified after release is expensive.

Software Quality

- Quality: Quality is defined as justification of all the requirements of a customer in a product.
    - Note: Quality is not defined in the product. It is defined in the customer`s mind.
- Quality software is reasonably
    - Bug-free.
    - Delivered on time.
    - Within budget.
    - Meets requirements and/or expectations.
    - Maintainable.

Error, bug & failure

- Error: Any incorrect human action that produces a problem in the system is called an error.
- Defect/Bug: Deviation from the expected behavior to the actual behavior of the system is called defect.
- Failure: The deviation identified by end-user while using the system is called a failure.

Why there are bugs in software?

- Miscommunication or no communication
- Software complexity
- Programming errors
- Changing requirements
- Lack of skilled testers

Etc..

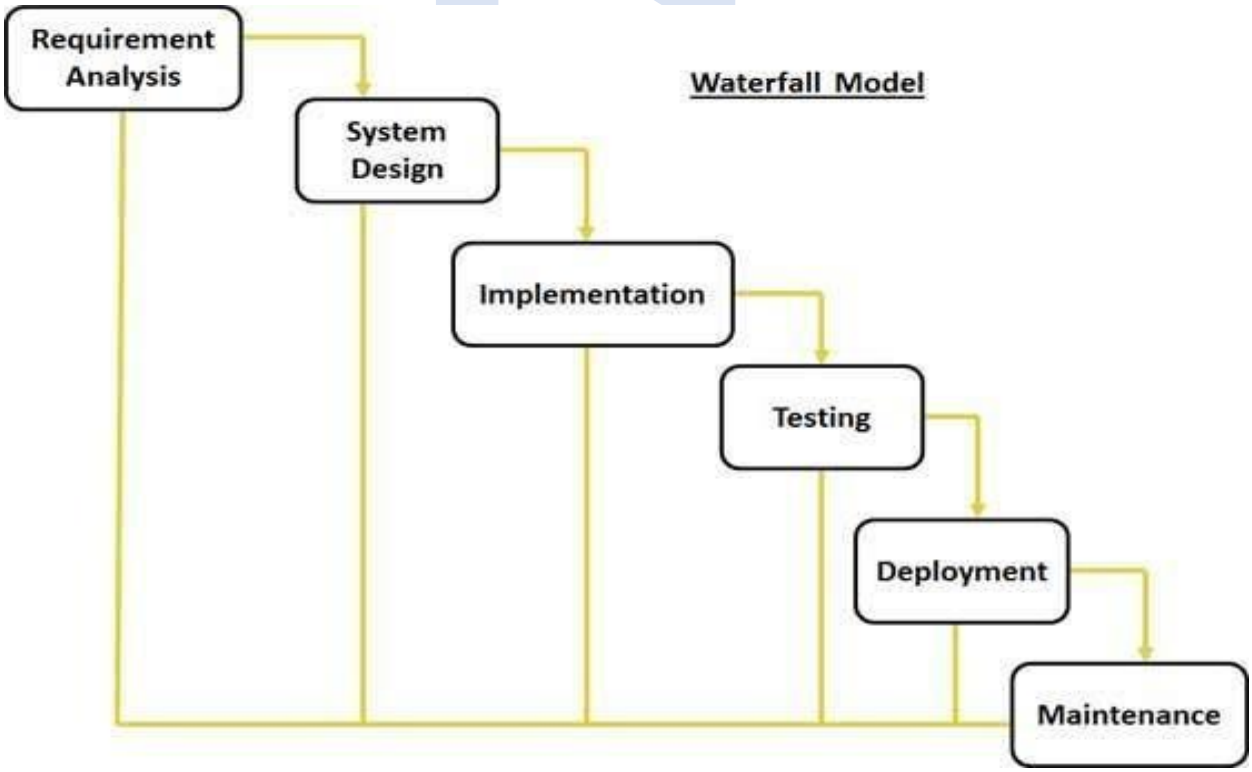
Software Development Life Cycle (SDLC)

- SDLC, Software Development Life Cycle is a process used by software industry to design, develop and test high quality software's.
- The SDLC aims to produce a high quality software that meets customer expectations.

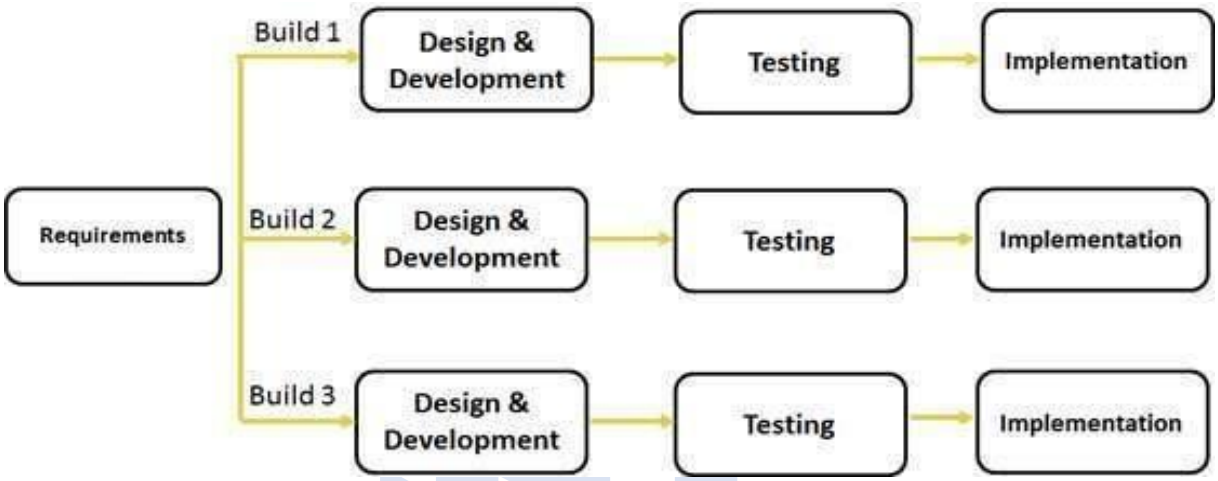
SDLC Models

- Waterfall Model
- Incremental Model
- Spiral Model
- V- Model
- Agile Model

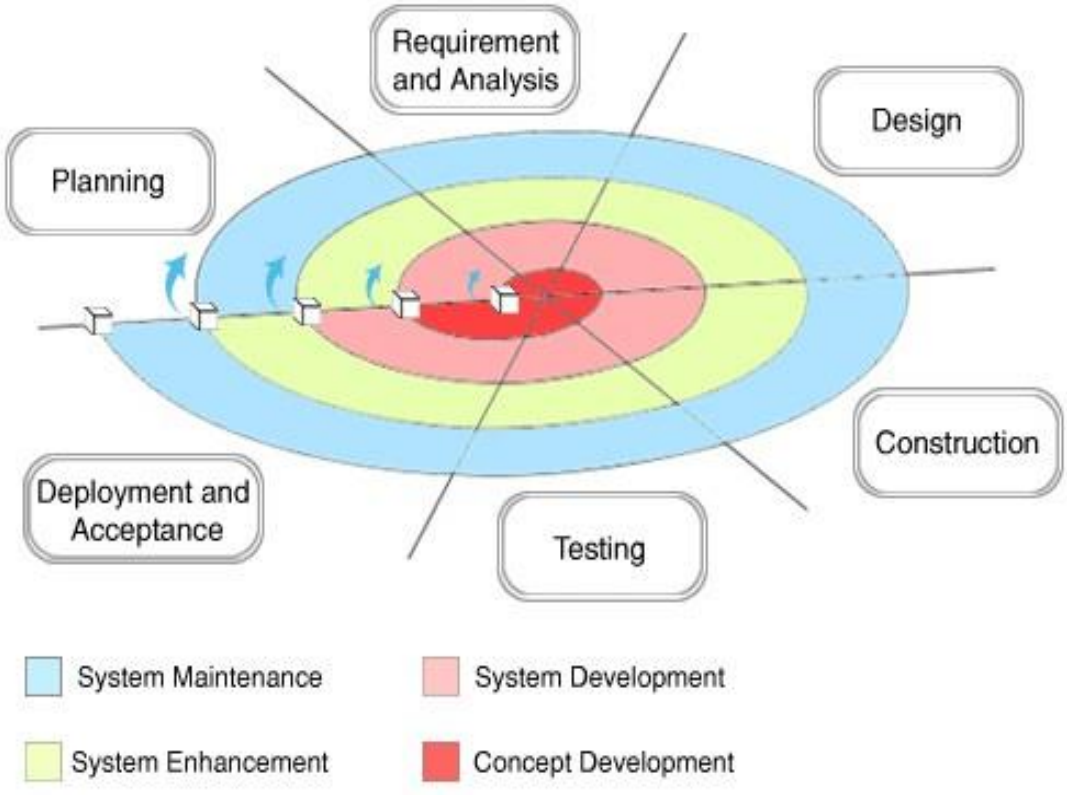
Waterfall Model



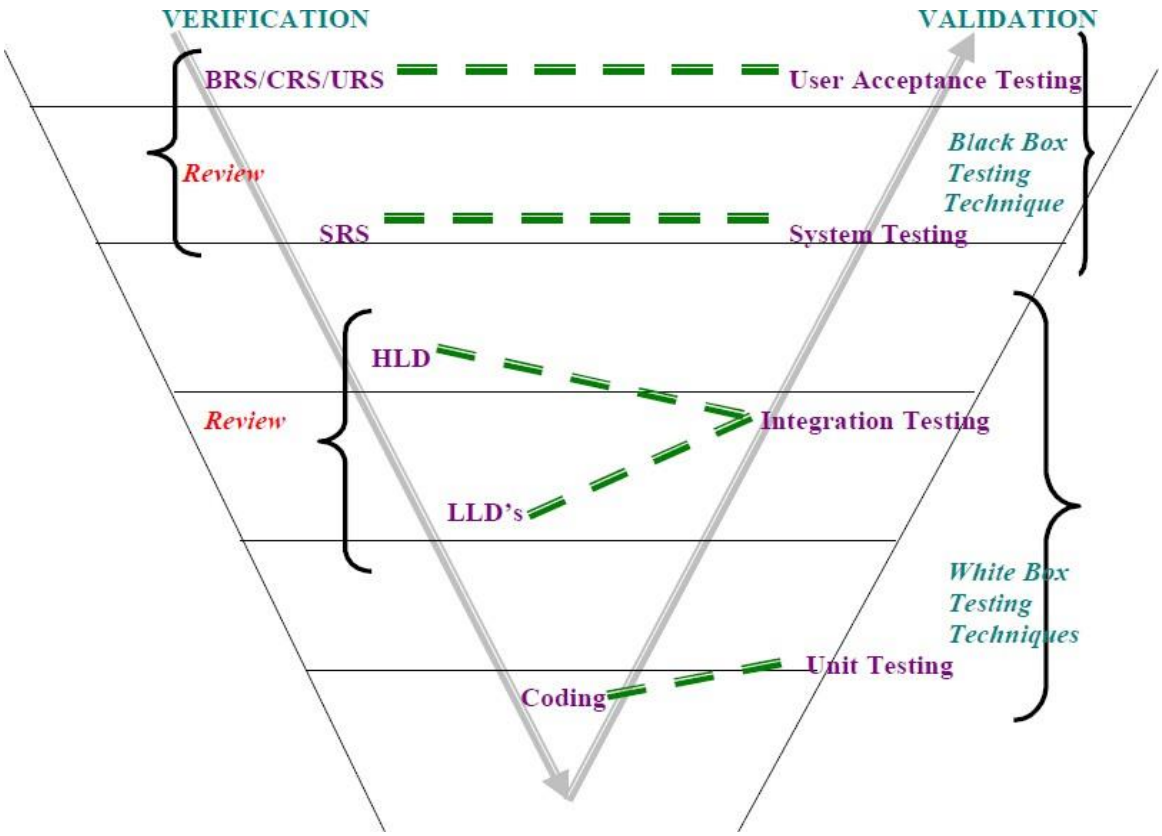
Incremental/Iterative Model



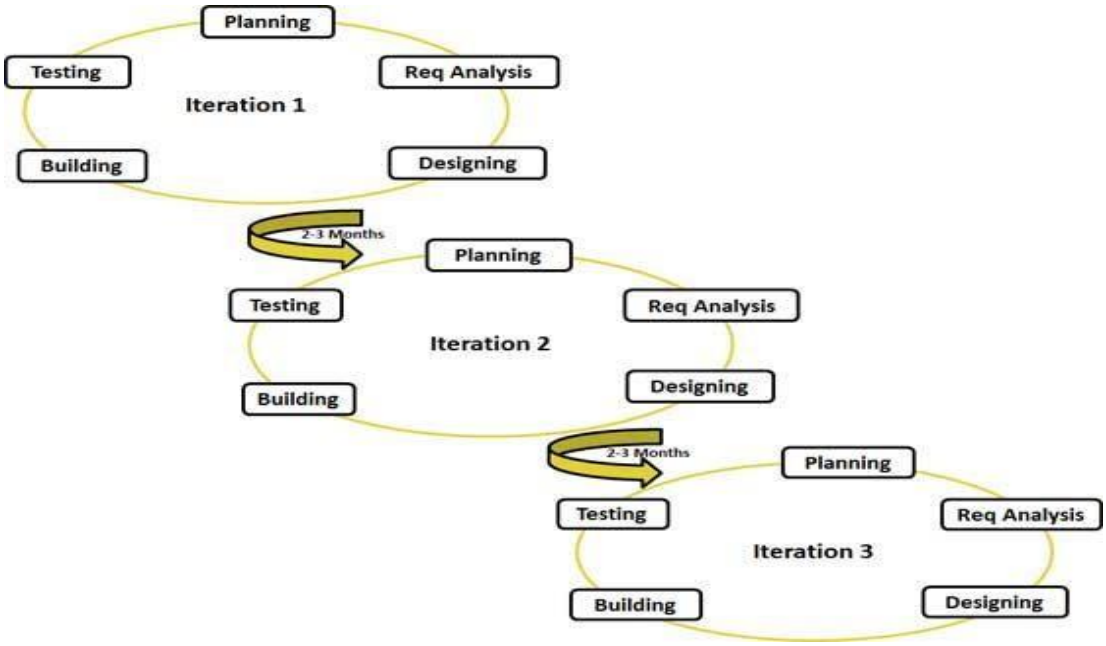
Spiral Model



V-Model



Agile Model



QA Vs QC/QE

- QA is Process related.
- QC is the actual testing of the software.
- QA focuses on building in quality.
- QC focuses on testing for quality.
- QA is preventing defects.
- QC is detecting defects.
- QA is process oriented.
- QC is Product oriented.
- QA for entire life cycle.

Verification V/S Validation

- Verification checks whether we are building the right system.
- Verification typically involves.
    - Reviews
    - Walkthroughs
    - Inspections
- Validation checks whether we are building the system right.
    - Takes place after verifications are completed.
- Validation typically involves actual testing.
    - System Testing

Static V/S Dynamic Testing

- Static testing is an approach to test project documents in the form of Reviews, Walkthroughs and Inspections.
- Dynamic testing is an approach to test the actual software by giving inputs and observing results.

Review, Walkthrough & Inspection

- Reviews:
  - Conducts on documents to ensure correctness and completeness.
  - Example:
    - Requirement Reviews
    - Design Reviews
    - Code Reviews
    - Test plan reviews
    - Test cases reviews etc.

Walkthroughs:

- It is a formal review and we can discuss/raise the issues at peer level.
- Also walkthrough does not have minutes of the meet. It can happen at any time and conclude just like that no schedule as such.

Inspections:

- Its a formal approach to the requirements schedule.
- At least 3- 8 people will sit in the meeting 1- reader 2-writer 3- moderator plus concerned.
- Inspection will have a proper schedule which will be intimated via email to the concerned developer/tester.

Levels of Software Testing

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing(UAT)

Unit Testing

- A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output.
- Unit testing conducts on a single program or single module.
- Unit Testing is white box testing technique.
- Unit testing is conducted by the developers.
- Unit testing techniques:
    - Basis path testing
    - Control structure testing
        - Conditional coverage
        - Loops Coverage
    - Mutation Testing

Integration Testing

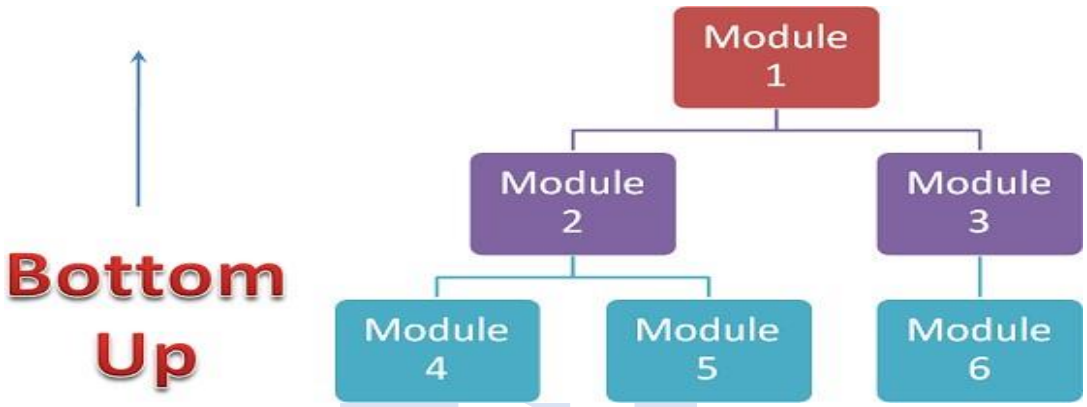
- In Integration Testing, individual software modules are integrated logically and tested as a group.
- Integration testing focuses on checking data communication amongst these modules.
- Integrated Testing is white box testing technique.
- Integrated testing is conducted by the developers.
- Approaches:
    - Top Down Approach
    - Bottom Up Approach
    - Sandwich Approach(Hybrid)

- **Stub**: Is called by the Module under Test.

- **Driver**: Calls the Module to be tested.

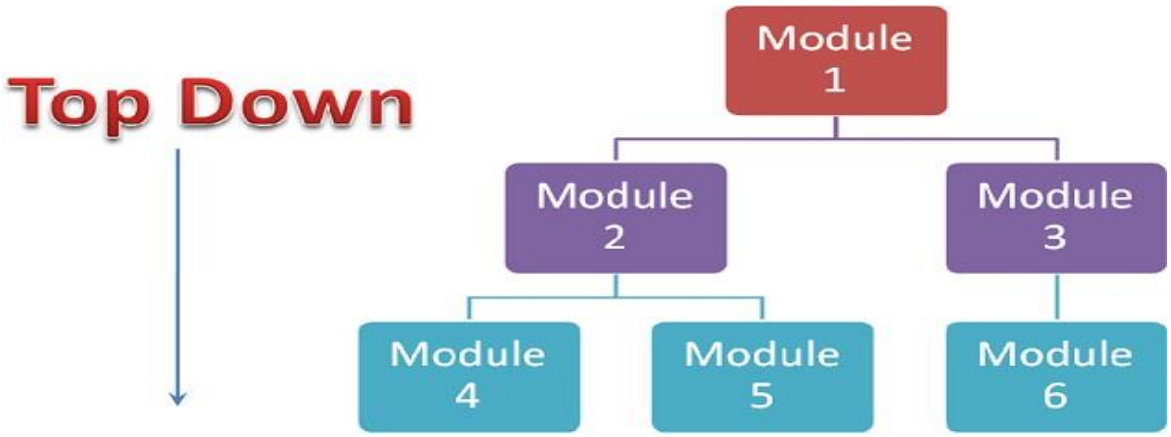
Bottom-Up Integration

- In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing.



Top down Integration

- In Top to down approach, testing takes place from top to down following the control flow of the software system.
- Takes help of stubs for testing.



System Testing

- Testing over all functionality of the application with respective client requirements.
- It is a black box testing technique.
- This testing is conducted by testing team.
- Before conducting system testing we should know the requirements.
- System Testing focusses on below aspects.
    - User Interface Testing (GUI)
    - Functional Testing
    - Non-Functional Testing
    - Usability Testing

User Acceptance Testing

- After completion of system testing UAT team conducts acceptance testing in two levels.
    - Alpha testing
    - Beta testing

Testing Methodologies

- White box Testing
- Black box Testing
- Grey box Testing

White Box Testing

- White Box Testing conducts on internal logic of the programs.
- Programming Skills are required.
- Ex: Unit Testing & Integration Testing

Block Box Testing

- Testing conducts on functionality of the application whether it is working according to customer requirements or not.
- Ex: System Testing & UAT Testing

Grey Box Testing

- Both combination of white box and black box testing.
- Ex: Database Testing

Black Box Testing Types

- GUI Testing
- Usability Testing
- Functional Testing
- Non-Functional Testing

What is GUI?

- There are two types of interfaces in a computer application.
    - Command Line Interface is where you type text and computer responds to that command.
    - GUI stands for Graphical User Interface where you interact with the computer using images rather than text.

GUI Testing

- Graphical User Interface (GUI) testing is the process of testing the system's GUI.
- GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.
- During GUI Testing Test Engineers validates user interface of the application as following aspects:
    - Look & Feel
    - Easy to use
    - Navigations & Shortcut keys
- GUI Objects:
    - Window, Dialog Box, Push Button, Radio Button, Radio Group, Tool bar, Edit Box, Text Box, Check Box, List Box, Drop down Box, Combo Box, Tab, Tree view, progress bar, Table, Scroll bar Etc.

Check list for GUI Testing

- It checks if all the basic elements are available in the page or not.
- It checks the spelling of the objects.
- It checks alignments of the objects.
- It checks content display in web pages.
- It checks if the mandatory fields are highlights or not.
- It checks consistency in background color and color font type and fond size etc.

Usability Testing

- During this testing validates application provided context sensitive help or not to the user.
- Checks how easily the end users are able to understand and operate the application is called usability testing.

Functional Testing

- Object Properties Coverage
- Input Domain Coverage (BVA, ECP)
- Database Testing/Backend Coverage
- Error Handling Coverage
- Calculations/Manipulations Coverage
- Links Existence & Links Execution
- Cookies & Sessions

Object Properties Testing

- Every object has certain properties.
    - Ex: Enable, Disable, Focus, Text etc..
- During Functional testing Test Engineers validate properties of objects in run time.

Input Domain Testing

- During Input domain testing Test Engineers validate data provided to the application w.r.t value and length.
- There are two techniques in Input domain Techniques.
    - Equalance Class Partition (ECP)
    - Boundary Value Analysis(BVA)

ECP & BVA

- Requirement:-
- User name field allows only lower case with min 6 max 8 letters.

ECP for User Name

Valid	Invalid
a....z	A....Z 0...9 Special characters (@ , #, $, & etc..)

BVA for User Name

Parameters	Value	Result
Min	6	Valid
Min+1	7	Valid
Min-1	5	Invalid
Max	8	Valid
Max+1	9	Invalid
Max-1	7	Valid

Database Testing

- During Database testing Test Engineers validate the data w.r.t database.
- Validates DML operations(Insert , Update, Delete & Select)
- SQL Language: DDL, DML, DCL etc..
- DDL - Data Definition Langue - Create , alter, drop
- DML - Data Manipulation language - Insert, update, select, delete
- DCL - Commit, roll back etc.

Error Handling Testing

- Validate error messages thrown by the application when we provide invalid data.
- The error messages should be clear and easy to understand to the user.

Calculations/Manipulations Testing

- Validate mathematical calculations.

Links Coverage

- Links existence - Links placed in the appropriate location or not.
- Links execution - link is navigating to appropriate page or not.
- Types of links:-
    - Internal links
    - External links
    - Broken links

Cookies & Sessions

- Cookie- Temporary internet files which are created at client side when we open the web sites. These files contains User data.

- Session- Sessions are time slots which are allocated to the user at the serve side.

Non-Functional Testing

- Performance Testing
    - Load Testing
    - Stress Testing
    - Volume Testing
- Security Testing
- Recovery Testing
- Compatibility Testing
- Configuration Testing
- Installation Testing

Performance Testing

- Load: Testing speed of the system while increasing the load gradually till the customer expected number.
- Stress: Testing speed of the system while increasing/reducing the load on the system to check any where its breaking.
- Volume: Check how much volumes of data is able to handle by the system.

Security Testing

- Testing security provided by the system.
- Types:
    - Authentication
    - Access Control/Authorization
    - Encryption/Decryption

Recovery Testing

- Testing Recovery provided by the system. Whether the system recovering abnormal to normal condition or not.

Compatibility Testing

- Testing Compatibility of the system w.r.t OS, H/W & Browsers.
- Operating System Compatibility
- Hardware Compatibility (Configuration Testing)
- Browser Compatibility
- Forward & Backward Compatibility

Installation Testing

- Testing Installation pf the application on Customer expected platforms and check installation steps, navigation, how much space is occupied in memory.
- Check Un-Installation.

Sanitation/Garbage Testing

- Check whether application is providing any extra/additional features beyond the customer requirements.

Testing Terminology

- **Adhoc Testing:**
  - Software testing performed without proper planning and documentation.
  - Testing is carried out with the knowledge of the tester about the application and the tester tests randomly without following the specifications/requirements.
- **Monkey Testing:**
  - Test the functionality randomly without knowledge of application and test cases is called Monkey Testing.

Testing Terminology cont...

- **Re- Testing:**
  - Testing functionality repetitively is called re-testing.
  - Re-testing gets introduced in the following two scenarios.
  - Testing is functionality with multiple inputs to confirm the business validation are implemented (or) not.
  - Testing functionality in the modified build is to confirm the bug fixers are made correctly (or) not.
- **Regression Testing:**
  - It is a process of identifying various features in the modified build where there is a chance of getting side-effects and retesting these features.
  - The new functionalities added to the existing system (or) modifications made to the existing system .It must be noted that a bug fixer might introduce side-effects and a regression testing is helpful to identify these side effects.

Testing Terminology cont...

- **Sanity Testing:**
  - This is a basic functional testing conducted by test engineer whenever receive build from development team.

- **Smoke Testing:**
  - This is also basic functional testing conducted by developer or tester before releasing the build to the next cycle.

Testing Terminology cont...

- **End to End Testing:**
  - Testing the overall functionalities of the system including the data integration among all the modules is called end-to-end testing.

- **Exploratory Testing:**
  - Exploring the application and understanding the functionalities adding or modifying the existing test cases for better testing is called exploratory testing.

Testing Terminology cont...

- **Globalization Testing (or) Internationalization Testing(I18N):**
  - Checks if the application has a provision of setting and changing languages date and time format and currency etc. If it is designed for global users, it is called globalization testing.

- **Localization Testing:**
  - Checks default languages currency date and time format etc. If it is designed for a particular locality of users is called Localization testing.

Testing Terminology cont...

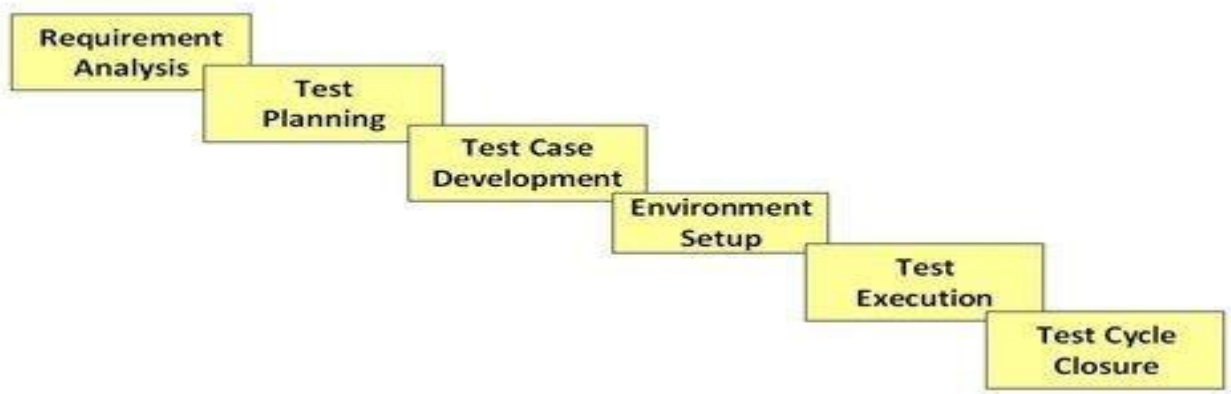
- Positive Testing (+ve):
  - Testing conducted on the application in a positive approach to determine what system is supposed to do is called a positive testing.
  - Positive testing helps in checking if the customer requirements are justifying the application or not.

- Negative Testing (-ve):
  - Testing a software application with a negative perception to check what system is not supposed to do is called negative testing.

Software Testing Life Cycle (STLC)

- Requirement Analysis
- Test Planning
- Test Designing
- Test Execution
- Test Closure

STLC (Software Testing Life Cycle)



STLC

Microsoft Excel Worksheet

Test Plan Definition

- A document describing the scope, approach, resources and schedule of testing activities.
- It identifies test items, the feature to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.
- A detail of how the test will proceed, who will do the testing, what will be tested, in how much time the test will take place, and to what quality level the test will be performed.

Test Plan Contents

- Objective
- Features to be tested
- Features not to be tested
- Test Approach
- Entry Criteria
- Exit Criteria
- Suspension & Resumption criteria
- Test Environment
- Test Deliverables & Milestones
- Schedules
- Risks & Mitigations

Use case, Test Scenario & Test Case

- Use Case:
  - Use case describes the requirement.
  - Use case contains THREE Items.
    - Actor
    - Action/Flow
    - Outcome
- Test Scenario:
  - Possible area to be tested (What to test)
- Test Case:
  - How to test
  - Describes test steps, expected result & actual result

Use Case V/s Test Case

- Use Case – Describes functional requirement, prepared by Business Analyst(BA).

- Test Case – Describes Test Steps/ Procedure, prepared by Test Engineer.

Test Case V/s Test Scenario

- Test case consist of set of input values, execution precondition, excepted Results and executed post condition, developed to cover certain test Condition. While Test scenario is nothing but test procedure.

- Test cases are derived (or written) from test scenario. The scenarios are derived from use cases.

- In short, Test Scenario is 'What to be tested' and Test Case is 'How to be tested'.

- Example:-
- Test Scenario: Checking the functionality of Login button
    - TC1: Click the button without entering user name and password.
    - TC2: Click the button only entering User name.
    - TC3: Click the button while entering wrong user name and wrong password.

Positive V/s Negative Test Cases

- Requirement:
    - For Example if a text box is listed as a feature and in SRS it is mentioned as Text box accepts 6 - 20 characters and only alphabets.
- Positive Test Cases:
    - Textbox accepts 6 characters.
    - Textbox accepts upto 20 chars length.
    - Textbox accepts any value in between 6-20 chars length.
    - Textbox accepts all alphabets.
- Negative Test Cases:
    - Textbox should not accept less than 6 chars.
    - Textbox should not accept chars more than 20 chars.
    - Textbox should not accept special characters.
    - Textbox should not accept numerical.

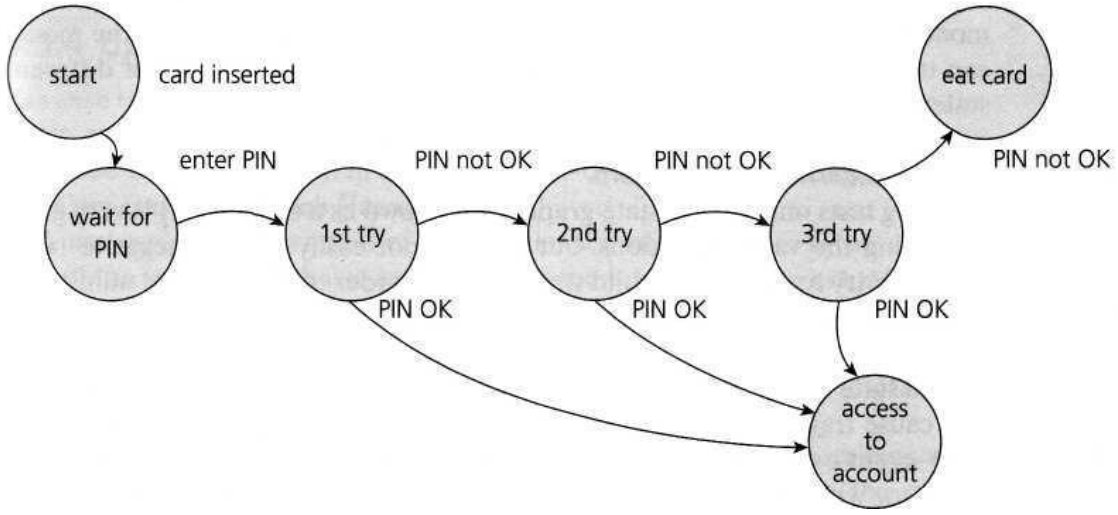
Test case Design Techniques

- Boundary Value Analysis (BVA)
- Equivalence Class Partitioning (ECP)
- Decision Table Testing
- State Transition Diagram
- Use Case Testing

Decision Table

CONDITIONS	EMAIL	V	V	I	I	B	V	I	B	B
	PASSWORD	V	I	V	I	V	B	B	I	B
ACTIONS	EXPECTED RESULT	PASSED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED

State Transition Diagram

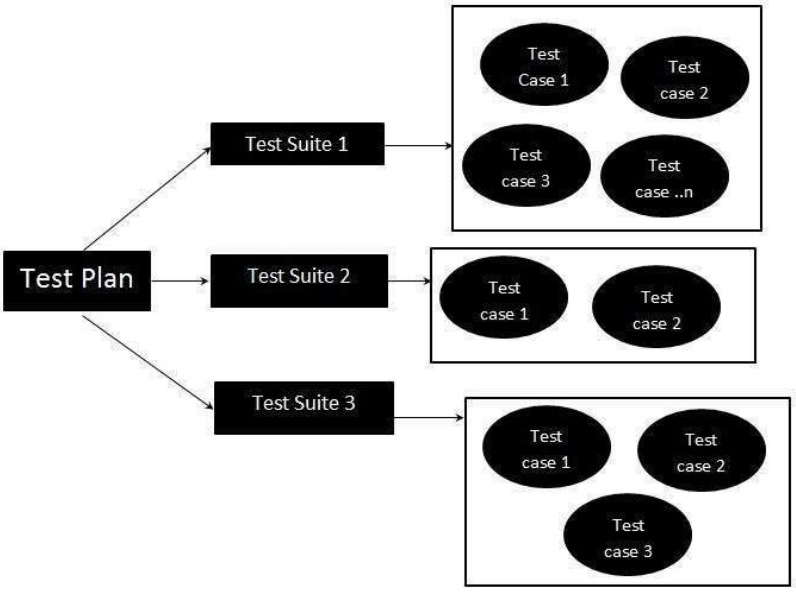


Use Case

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit

Test Suite

- Test Suite is group of test cases which belongs to same category.



Test Case Contents

- Test Case ID
- Test Case Title
- Description
- Pre-condition
- Priority (P0, P1,P2,P3) – order
- Requirement ID
- Steps/Actions
- Expected Result
- Actual Result
- Test data

Requirement Traceability Matrix(RTM)

- RTM – Requirement Traceability Matrix
- Used for mapping of Requirements w.r.t Test cases

TEST CASE ID	REQUIREMENT ID
1	1.0
2	1.0
3	1.1
4	1.2
5	1.2
6	2.0

Characteristics of good Test Case

- A good test case has certain characteristics which are:
  - Should be accurate and tests what it is intended to test.
  - No unnecessary steps should be included in it.
  - It should be reusable.
  - It should be traceable to requirements.
  - It should be compliant to regulations.
  - It should be independent i.e. You should be able to execute it in any order
  - without any dependency on other test cases.

- It should be simple and clear, any tester should be able to understand it by reading once.

Test Data

- Test Data is required for Test cases.
- Test Data will be provided as input to the cases.
- Prepared by Test Engineers.
- Maintains in Excel sheets.

Test Environment Setup (Test Bed)

- Test environment decides the software and hardware conditions under which a work product is tested.
- 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 test team 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.

Defect Reporting

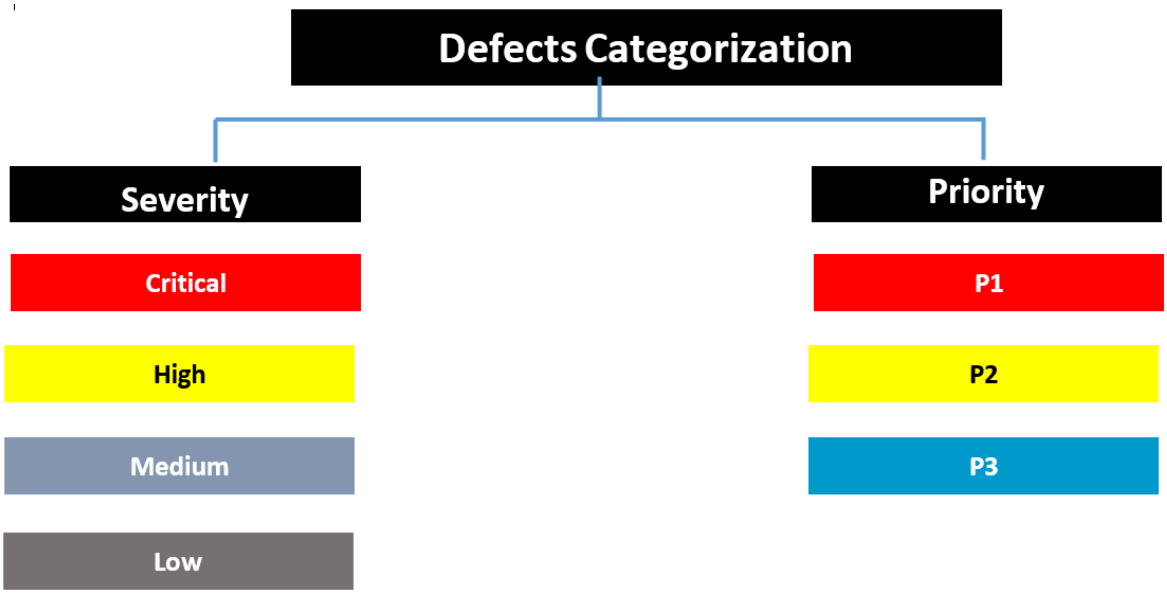
- Any mismatched functionality found in a application is called as Defect/Bug/Issue.
- During Test Execution Test engineers are reporting mismatches as defects to developers through templates or using tools.
- Defect Reporting Tools:
  - Clear Quest
  - DevTrack
  - Jira
  - Quality Center
  - Bug Jilla etc.

Defect Report Contents

- Defect_ID - Unique identification number for the defect.
- Defect Description - Detailed description of the defect including information about the module in which defect was found.
- Version - Version of the application in which defect was found.
- Steps - Detailed steps along with screenshots with which the developer can reproduce the defects.
- Date Raised - Date when the defect is raised
- Reference- where in you Provide reference to the documents like . requirements, design, architecture or may be even screenshots of the errorto help understand the defect
- Detected By - Name/ID of the tester who raised the defect
- Status - Status of the defect , more on this later
- Fixed by - Name/ID of the developer who fixed it
- Date Closed - Date when the defect is closed
- Severity which describes the impact of the defect on the application

Defect Management Process

Defect Classification



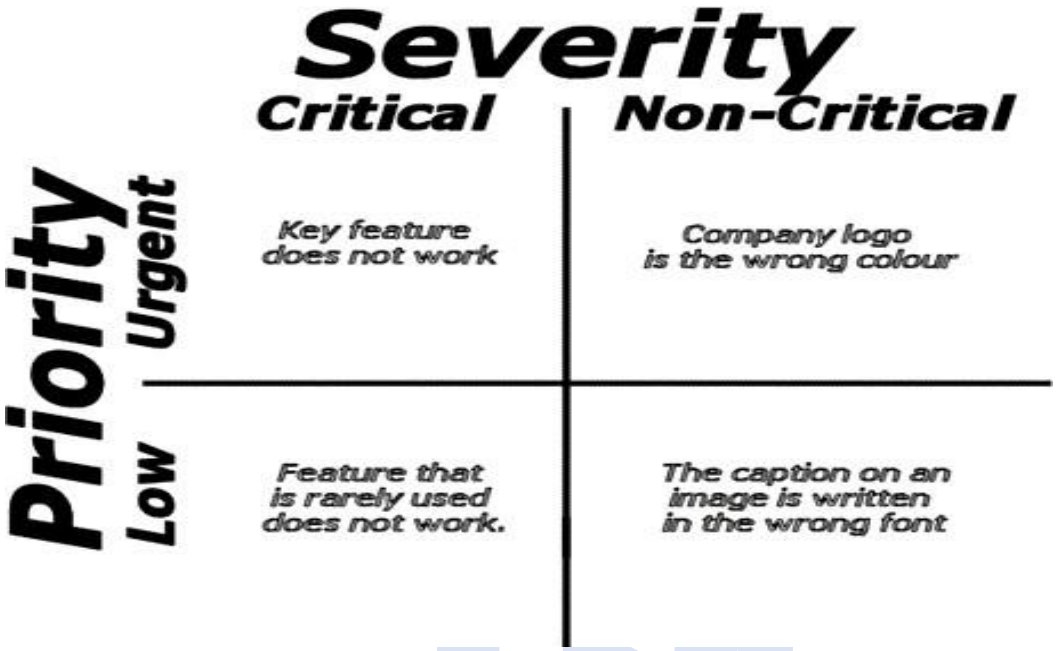
Defect Severity

- Severity describes the seriousness of defect.
- In software testing, defect severity can be defined as the degree of impact a defect has on the development or operation of a component application being tested.
- Defect severity can be categorized into four class
    - Critical: This defect indicates complete shut-down of the process, nothing can proceed further
    - High: It is a highly severe defect and collapse the system. However, certain parts of the system remain functional
    - Medium: It cause some undesirable behavior, but the system is still functional
    - Low: It won't cause any major break-down of the system

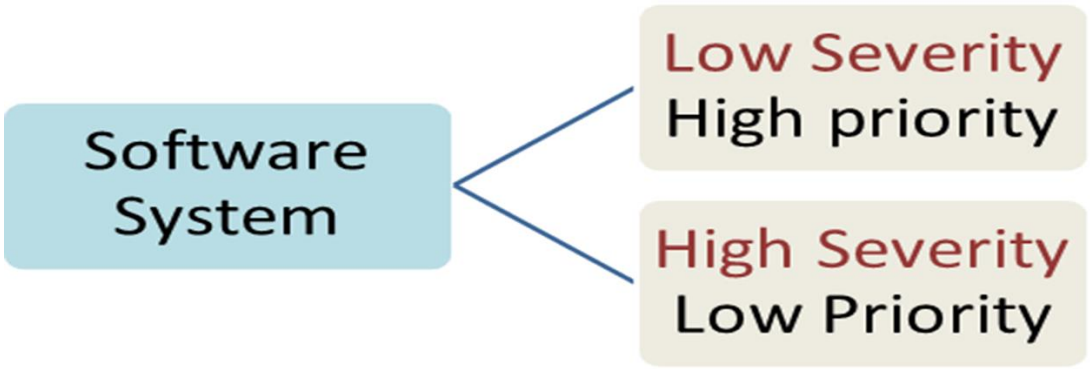
Defect Priority

- Priority describes the importance of defect.
- Defect Priority states the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.
- Defect priority can be categorized into three class
    - P1 (High) : The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed
    - P2 (Medium): During the normal course of the development activities defect should be resolved. It can wait until a new version is created
    - P3 (Low): The defect is an irritant but repair can be done once the more serious defect have been fixed

Tips for determining Severity of a Defect



A software system can have a



- A very low severity with a high priority:
    - A logo error for any shipment website, can be of low severity as it not going to affect the functionality of the website but can be of high priority as you don't want any further shipment to proceed with wrong logo.

- A very high severity with a low priority:
    - For flight operating website, defect in reservation functionality may be of high severity but can be a low priority as it can be scheduled to release in a next cycle.

Some more examples...

- Example for High Priority & High Severity defect:
    - If 'Login' is required for an Application and the users are unable to login to the application with valid user credentials. Such defects need to be fixed with high importance. Since it is stopping the customer to progress further.

- Example for Low Priority and High Severity defect:
    - If an application crashes after multiple use of any functionality i.e. if 'Save' Button (functionality) is used for 200 times and then the application crashes, such defects have High Severity because application gets crashed, but Low Priority because no need to debug right now you can debug it after some days.

- Example for High Priority & Low Severity defect:
    - If in a web application, company name is miss spelled or Text "User Nam:" is displayed instead of "User Name:" on the application login page. In this case, Defect Severity is low as it is a spell mistake but Priority is high because of its high visibility.

- Low priority-Low severity - A spelling mistake in a page not frequently navigated by users.
- Low priority-High severity - Application crashing in some very corner case.
- High priority-Low severity - Slight change in logo color or spelling mistake in company name.
- High priority-High severity - Issue with login functionality.

Defect Resolution

- After receiving the defect report from the testing team, development team conduct a review meeting to fix defects. Then they send a Resolution Type to the testing team for further communication.
- Resolution Types:-
  - Accept
  - Reject
  - Duplicate
  - Enhancement
  - Need more information
  - Not Reproducible
  - Fixed
  - As Designed

Defect Triage

- Defect triage is a process that tries to do the re-balancing of the process where test team faces the problem of limited availability of resources.
- When there are large number of the defects and limited testers to verify them, defect triage helps trying to get as many defects resolved based on defect parameters like severity and priority.

- Defect Triage Process:

The triage process includes following steps

- Reviewing all the defects including rejected defects by the team
- Initial assessment of the defects is based on its content and respective priority and severity settings
- Prioritizing the defect based on the inputs
- Assign the defect to correct release by product manager
- Re-directs the defect to the correct owner/team for further action

Guidelines that every tester should consider before selecting severity

- Understand the concept of priority and severity well
- Always assign the severity level based on the issue type as this will affect its priority
- Understand how a particular scenario or test case would affect the end- user
- Need to consider how much time it would take to fix the defect based on its complexity and time to verify defect

Exercise

- Assign the Severity for the following issues.

1	The website performance is too slow	
2	The login function of the website does not work properly	
3	The GUI of the website does not display correctly on mobile devices	
4	The website could not remember the user login session	
5	Some links doesn't work	

Here are the recommended answers

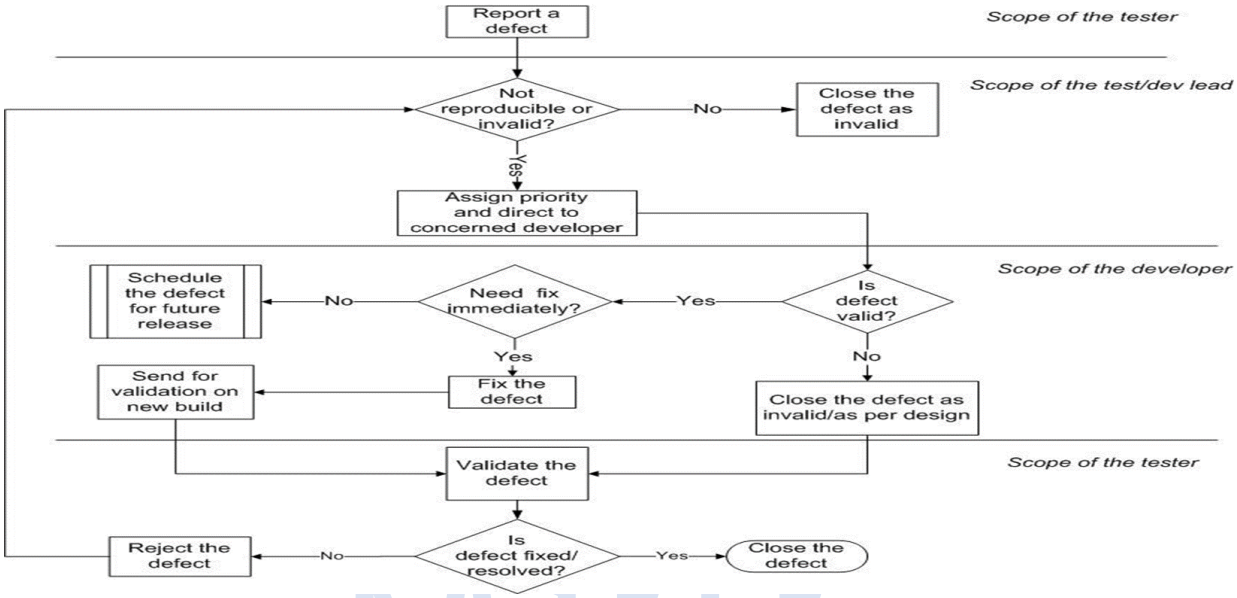
No.	Description	Severity	Explanation
1	The website performance is too slow	High	The performance bug can cause huge inconvenience to user.
2	The login function of the website does not work properly	Critical	Login is one of the main function of the banking website if this feature does not work, it is serious bugs
3	The GUI of the website does not display correctly on mobile devices	Medium	The defect affects the user who use Smartphone to view the website.
4	The website could not remember the user login session	High	This is a serious issue since the user will be able to login but not be able to perform any further transactions
5	Some links doesn't work	Low	This is an easy fix for development guys and the user can still access the site without these links

Tips for good Bug report

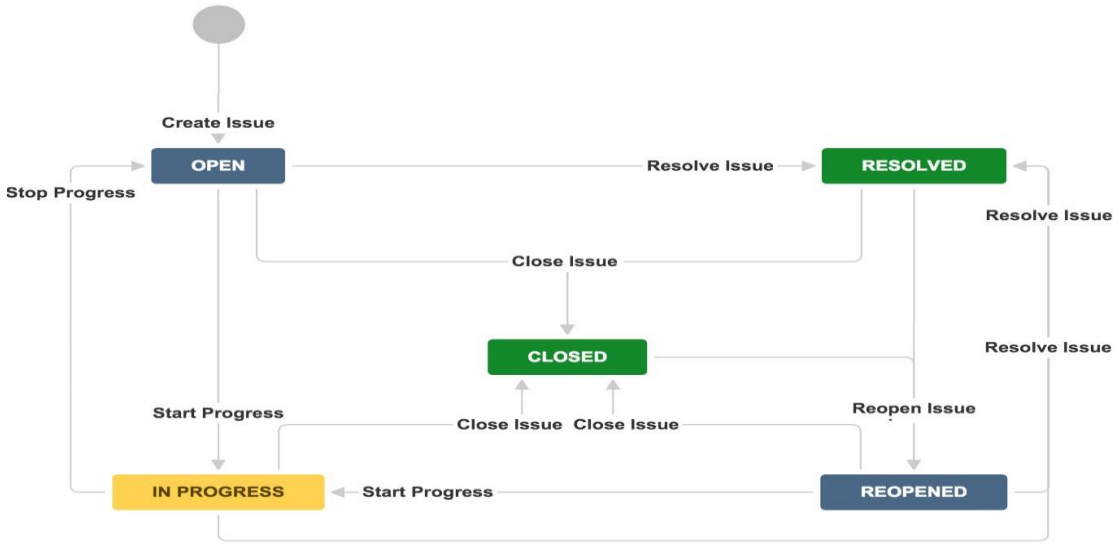
- Structure: test carefully (Use deliberate, careful approach to testing)
- Reproduce: test it again (rule of thumb 3 times)
- Isolate: test it differently (Change variables that may alter symptom)
- Generalize: test it elsewhere Does the same failure occur in other modules or locations?
- Compare: review results of similar tests (Same test run against earlier versions)
- Summarize: relate test to customers (Put a short "tag line" on each report)

- Condense: trim unnecessary information (Eliminate extraneous words or steps)
- Disambiguate: use clear words (Goal: Clear, indisputable statements of fact)
- Neutralize: express problem impartially (Deliver bad news gently)

Bug Life Cycle



Jira Defect Workflow



Sates of Defects

- New: A bug is reported and is yet to be assigned to developer
- Open: The test lead approves the bug
- Assigned: Assigned to a developer and a fix is in progress
- Need more info: When the developer needs information to reproduce the bug or to fix the bug
- Fixed: Bug is fixed and waiting for validation
- Closed: The bug is fixed, validated and closed
- Rejected: Bug is not genuine
- Deferred: Fix will be placed in future builds
- Duplicate: Two bugs mention the same concept
- Invalid: Not a valid bug
- Reopened: Bug still exists even after the bug is fixed by the developer

Test Cycle Closure

- **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

QA/Testing Activities

- Understanding the requirements and functional specifications of the application.
- Identifying required Test Scenario's.
- Designing Test Cases to validate application.
- Execute Test Cases to valid application
- Log Test results (How many test cases pass/fail).
- Defect reporting and tracking.
- Retest fixed defects of previous build
- Perform various type of testing assigned by Test Lead (Functionality, Usability, User Interface
- and compatibility) etc.,
- Reports to Test Lead about the status of assigned tasks
- Participated in regular team meetings.
- Creating automation scripts for Regression Testing.

Test Metrics

SNO	Required Data
1	No. Of Requirements
2	Avg. No. of Test Cases written Per Requirement
3	Total No.Of Test Cases written for all Requirement
4	Total No. Of test cases Executed
5	No.of Test Cases Passed
6	No.of Test Cases Failed
7	No.of Test cases Blocked
8	No. Of Test Cases Un Executed
9	Total No. Of Defects Identified
10	Critical Defects Count
11	Higher Defects Count
12	Medium Defects Count
13	Low Defects Count
14	Customer Defects
15	No.of defects found in UAT

- % of Test cases Executed:
  - No.of Test cases executed / Total No. of Test cases written) * 100
- % of test cases NOT executed:
  - (No.of Test cases NOT executed/Total No. of Test cases written) * 100
- % Test cases passed
  - (No.of Test cases Passed /Total Test cases executed) * 100
- % Test cases failed
  - (No.of Test cases failed / Total Test cases executed) * 100
- %Test cases blocked
  - (No.of test cases blocked / Total Test cases executed) * 100
- Defect Density: Number of defects identified per requirement/s
  - No.of defects found / Size(No. of requirements)
- Defect Removal Efficiency (DRE):
  - (A / A+B) * 100
  - (Fixed Defects / (Fixed Defects + Missed defects)) * 100
    - A- Defects identified during testing/ Fixed Defects
    - B- Defects identified by the customer/Missed defects
- Defect Leakage:
  - (No.of defects found in UAT / No. of defects found in Testing) * 100
- Defect Rejection Ratio:
  - (No. of defect rejected /Total No. of defects raised) * 100
- Defect Age: Fixed date-Reported date

Test Efficiency Vs Test Effectiveness

- Effectiveness - How well the user achieves the goals they set out to achieve using the system (process).
- Efficiency - The resources consumed in order to achieve their goals.

What is Agile?

- AGILE is a methodology that promotes continuous iteration of development and testing throughout the software development life cycle of the project.

- Both development and testing activities are concurrent unlike the Waterfall model.

- The agile software development emphasizes on four core values.
    - Individual and team interactions over processes and tools
    - Working software over comprehensive documentation
    - Customer collaboration over contract negotiation
    - Responding to change over following a plan