# Software Testing

## Fundamentals of software testing

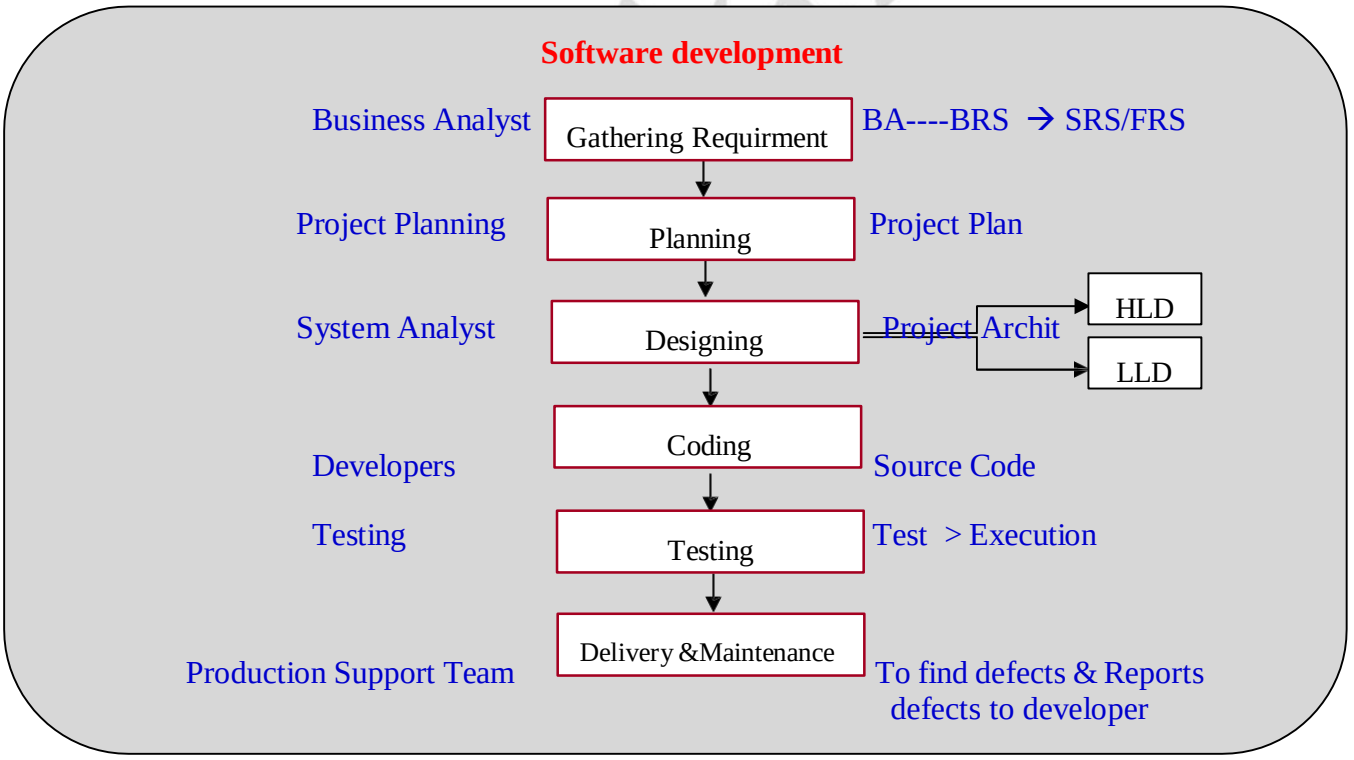
### Introduction:

This software testing eBook is helpful resource to understand software testing and quality assurance concepts. We tried to make this eBook simple to understand with practical information wherever possible.

This eBook covers almost all testing terminologies which are useful for preparing for software testing job interview.

Besides this we have hundreds of articles covering everything from software testing industry. You can read all these articles on: **http://www.softwaretestinghelp.com/**

Please feel free to share this eBook with your friends and help them understand the software testing concepts.




**Errors**: Any incorrect human action that produces a problem in the system is caller error.

**Defect**: Deviation between expected behavior to actual behavior of the system is called defect.

**Failure**: The deviation identified by end-user while using a system is called a failure.

***Note :**

***Presence of errors results in defects and presence of defects results in failure of the product.***

** When defects will arise while developing software?

**Request 1** **Request 2** **Request 3** **Request 4**

| | | | |
|---|---|---|---|
| Correct Requirement | Correct Requirement | Correct Requirement | Incorrect Requirement |
| Designed as per requirement | Designed as per requirement | Mistakes made in design | Designed as per requirement |
| Developed as per design | Mistakes in development | Developed as per design | developed as per design |
| Correct Product | Product has coding defects | Product has design defects | Wrong product. |

* The above diagram explains the importance of early testing.

**Early Testing:** Conducting testing as soon as possible in development life cycle to find defects at early stages is called early testing. Early testing is helpful to reduce the cost of fixing defects.

* Why does software have defects?

* Incorrect requirements

* Wrong design

* Poor coding

* Complex business logic

* Complex technology

* Work pressure

* Frequently changing requirements.

**Testing**: it is a process of verifying are we developing the right product or not and also validating does the developed product is right or not.

**Software testing = Verification + Validation.**

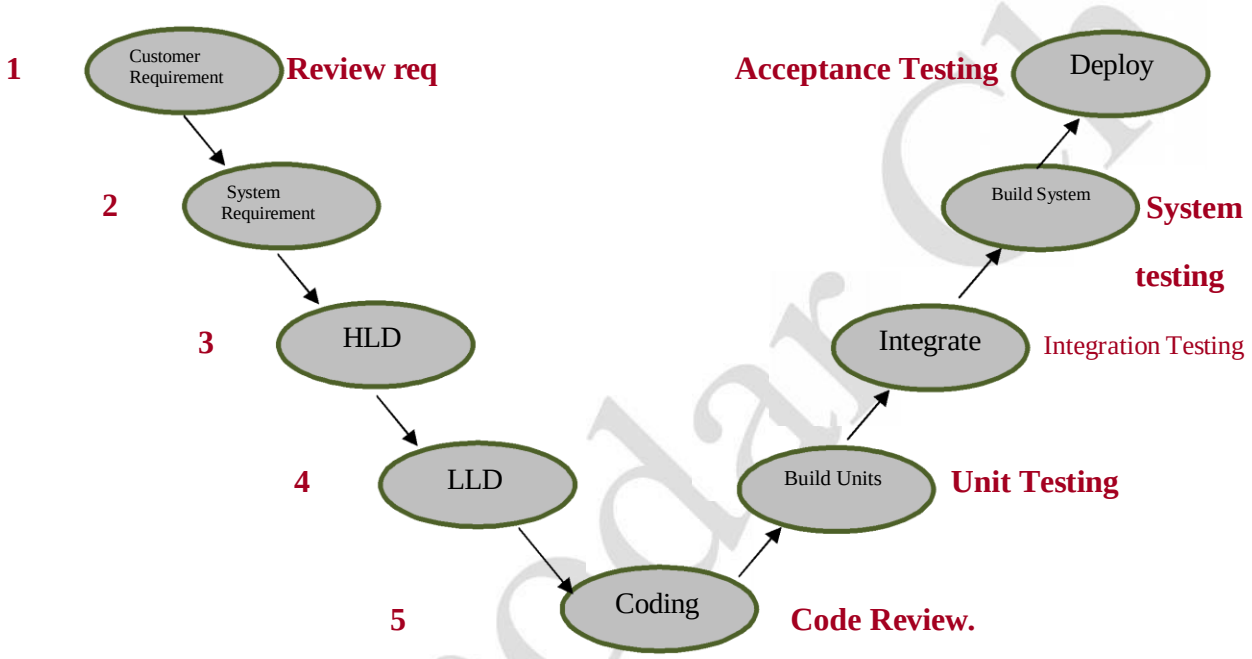
* What is Verification?

It is a process of verifying: Are we developing the right product or not. Known as static testing.

* What is Validation?

It is a process of validating: Does the developed product is right or not. Also called as dynamic testing.

## Verification Vs Validation




*******  1,2,3,4,5, are verification  and other side is validation.**

**Software Testing Techniques:**

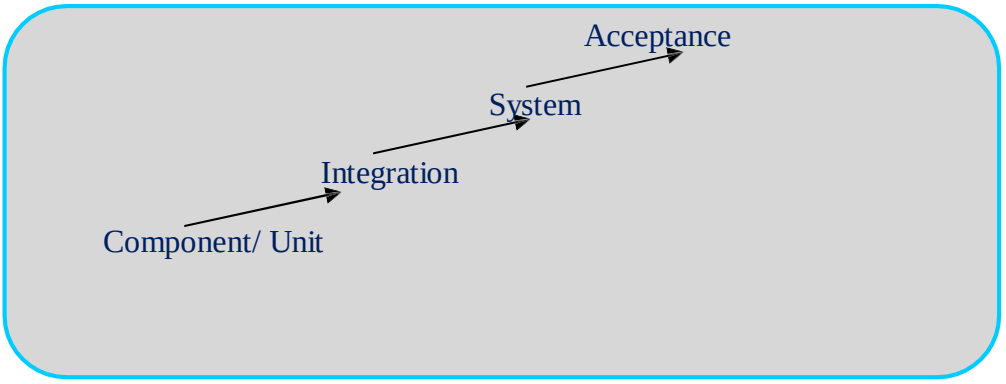
Static Testing

White Box Testing

Black Box Testing

Grey Box Testing.

**Levels of Dynamic Testing:**



# Validation

**Testing Approach**:

**Traditional testing approach**: Positive Testing (+ve).

- Show that the system:
  - Does what it should
  - Goal: show working.
  - Success: system works.
  - Easy to write system cases
  - Result: faults left in.

**Better Testing Approach / -ve Testing**:

- Show that the system:
  - Doesn't do what it shouldn't.
  - Does what is should

    **Goal**: find faults

    **Success**: system faults
  - Difficult to write test cases.
  - **Result**: fever faults left in.

**Most common defects**

- Incorrect functionality
- Incorrect data edits
- Poor performance
- Poor security
- Incompatibility
- Poor UI – User interface
- Poor usability

## Objectives of Testing :

**1)** To confirm whether the application is developed according to customer requirements or not

**2)** Finding defects

**3)** To make sure all problems are resolved and close.

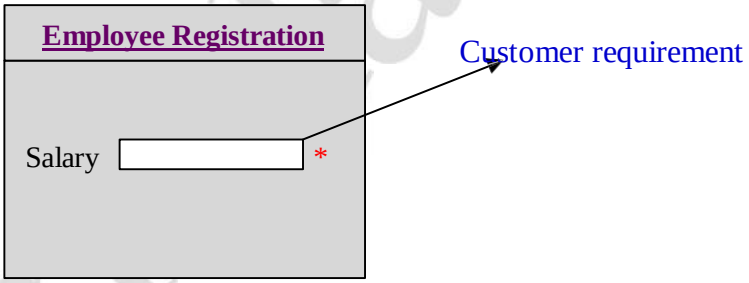
**4)** Finally testing is helpful to deliver a quality product and risk-free product to the customer.

## Software testing Principles:

**Principles of testing:** Exhaustive principle is impossible.

**Exhaustive Testing:** If you test functionality with all possible valid inputs and invalid inputs then it is called exhaustive testing.

*** Testing everything is called exhaustive testing.




It should accept a value between 5,000-50,000

- If you check the salary field 5000 , 5001,5002.....50000 and 4999, 4998 etc is called exhaustive testing.
- As exhaustive testing is impossible risk based testing is preferred (or) recommended..

**Risk Based Testing:** Identifying the operations which most likely to cause failures and then testing these functionalities on priority basis is called risk based testing.

**Defect Clustering:** The small number of modules or functionality may contain more number of defects. Concentrate more on testing these functionalities.

**Pesticide Paradox:** If prepared test cases are not finding defects, add/revise test cases to find more defects.

The prepared test cases are not helping to find defects then add or modify the test cases for better testing.

**Testing shows presence of defects:1** We have to test a application with an intension of showing defects. For this negative testing is the best approach.

**Early Testing:** Testing should start as early as possible in the SDLC.

**Testing is context dependent:** We have to select or opt appropriate testing approach based on the type of application we are testing.

**Absence of Errors is a fallocy:** Finding and fixing defects. 100% bug free app is impossible.

******* (Fallocy = False statement)**

**Static Testing:** Verifying if we developing the right system or not is called static testing. It is also called verification. This static testing covers reviews and walk through.

**Reviews :** Examine any project related work or project related work is called reviews.

**Types of Reviews:**

1) Management Reviews
2) Technical Reviews
3) Code Reviews
4) Test case Reviews (formal, Informal)

**Formal Reviews:** if any review is conducted with a prior plan and by following proper documentation and procedures are called formal reviews

Inspections and Audits are the best example of formal reviews.

**Informal Reviews:** if any review is conducted without following any procedures and documentation then reviews

**Walk-through:** knowledge transfer sessions in the form of peer review

**Objective of Reviews:** Reviews are helpful to determine

1) Defects in requirement.
2) Defects in design.
3) Deviations in coding standards.
4) To confirm if the prepared test cases are enough to validate a software
5) Reviews helpful improve the organization process.

**Dynamic Testing:** It is a process of checking if the source code and the application are working as expected. Also called dynamic testing or validation testing.

* **Levels of dynamic Testing**: dynamic testing will be carried out at 4 levels.

1) Unit Testing
2) Integration Testing
3) System Testing
4) User Acceptance Testing.

**Unit Testing:** A smallest separatable portion in the source code of the application is called unit. (functions, procedures, etc) Testing conducted on these units to check if the code behind the units is working as expected or not is called unit testing.

It is also called module testing or component testing

**Integration Testing:** Once all units are tested the programmers will integrate all units and check interactions among the units which is called integration testing.

**Note: Unit testing and integration testing is collectively called white box testing.**

**White box Testing:** Testing conducted on the source code by developers to check does the source code is working as expected or not is called white box testing.

* **What is the need of white box testing?**

As the source code is visible, finding and rectifying the problems is easy for developers.

The defects that are identified in white box testing are very economical to

resolve. To reduce the defects as early as possible white box testing is helpful.

To ensure 100% code coverage.

***** **White box testing is also called as glass box, structural, clear box testing.**

**Black box Testing:** Testing is conducted on the application by test engineers or by domain experts to check whether the application is working according to customer requirements.

**What is the need of Black Box Testing?**

1) White box testing conducted by developer in a technical perception where as black box testing is conducted by test engineers with end-user perception.

2) Programmers will conduct white box testing in a positive perception whereas tester will conduct black box testing with a negative perception where there is a more chance of finding more defects

   The more defects you identify results in a quality system.

3) White box testing will not cover non functional areas. As functional requirements are also very important for production system those are covered in black box testing.

4) The objective of white box testing is 100% coverage whereas the objective of black box testing is 100% customer business requirement coverage.

5) Black Box Testing = System Testing + User Accepting Testing which is called as requirement Based Testing (or) Specification Based Testing.

**System Testing:** Validating the functional and non functional requirements of the system is called system testing.

System testing broadly classified into two types i.e.

1) Functional system testing.

2) Non-functional system testing.

Functional system testing will be conducted both in a positive perception and also in a negative perception.

**Positive Testing (+ve ):** Testing conducted on the application in a positive approach to determine what system suppose to do is called positive testing.

******Note :***

***Positive testing is helpful to check whether the customer requirements are justifying by the application or not.***

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

******Note :***

***Negative testing is helpful to find defects from the software.***

**Functional System Testing Approach:** Smoke Testing:

It is a kind of quick test carried out on the application to determine whether the application is testable or not.

**Formal Testing:** If you tested software application by following all preplan procedures and proper documentation then it is called formal testing.

**Adhoc Testing:** If you test software without following any procedures and documentation then it is called adhoc-testing. It is also called informal testing.

**Risk Based Testing (or) Priority Based Testing:** Identifying the critical functionality in the system and testing it first

Or

Conducting testing in the same order of priority is called risk based testing or priority based testing.

**Re- Testing:** Testing functionality again or testing functionality repetitively is called re-testing.

Re-testing comes in the following 2 scenarios.

1) Testing a functionality with multiple inputs to confirm if the business validations are implemented or not

2) Testing a functionality on the modified build to confirm the bug fixers are made correctly or not.

**Regression Testing:** It is process of identifying various features in the modified build where there is a chance of getting affected and retesting these features.

1) The new functionalities added to the existing system or modifications made to the existing system or the bug fixes may introduce side-effects. Regression testing is helpful to identify these side effects.

**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 testing the functionalities

(or)

Understanding system, modifying existing test cases and executing those

**Monkey Testing:** Testing conducted on an application unevenly or zig zag way with an intension of finding tricky defects is called monkey testing.

**Non-Functional System Testing:** Validating various non functional aspects of the system such as user interfaces, user friendliness, security, compatibility, load, stress and performance etc is called non-functional system testing.

## Non Functional Testing

**UI / GUI Testing:** Validating if all user interfaces are professionally designed or not is called UI Testing.

**Check List for UI Testing:**

1) Check if all basic elements are available in the page or not.
2) Check spelling of the objects.
3) Check alignments of the objects.
4) Check content displayed in web pages.
5) Check if the mandatory fields are highlights or not.
6) Check consistency in background color, font type and fond size etc.

**Usability Testing:** Checking how easily the end user is able to understand and operate the application

**Security Testing:** Validating whether all security conditions are properly implemented in the software or not

**Check List for Security Testing:**

1) Check if the sensitive data such as password, credit card, CVV numbers are getting encrypted or not.
2) Check browser navigation after logout
3) Check direct URL access for the both secured and non secured pages.
4) Check for session expiry
5) Check view source code option for secured pages.
6) Check for Authorization
7) Check for Authentication
8) Check cookies

**Performance Testing:** It is a process of measuring various efficiency characteristics of a system such as response time, through put, load, stress transactions per minutes, transaction mix.

**Load Testing:** Analyzing functional and performance behavior of the application under various load conditions is called load testing.

**Stress Testing:** Checking the application behavior under stress conditions is called stress testing in other words reducing the system resources and keeping the load as constant checking how application is behaving is called stress testing.

**Recovery Testing:** Checking how system is able to handle some unexpected or unpredictable situations is called recovery testing.

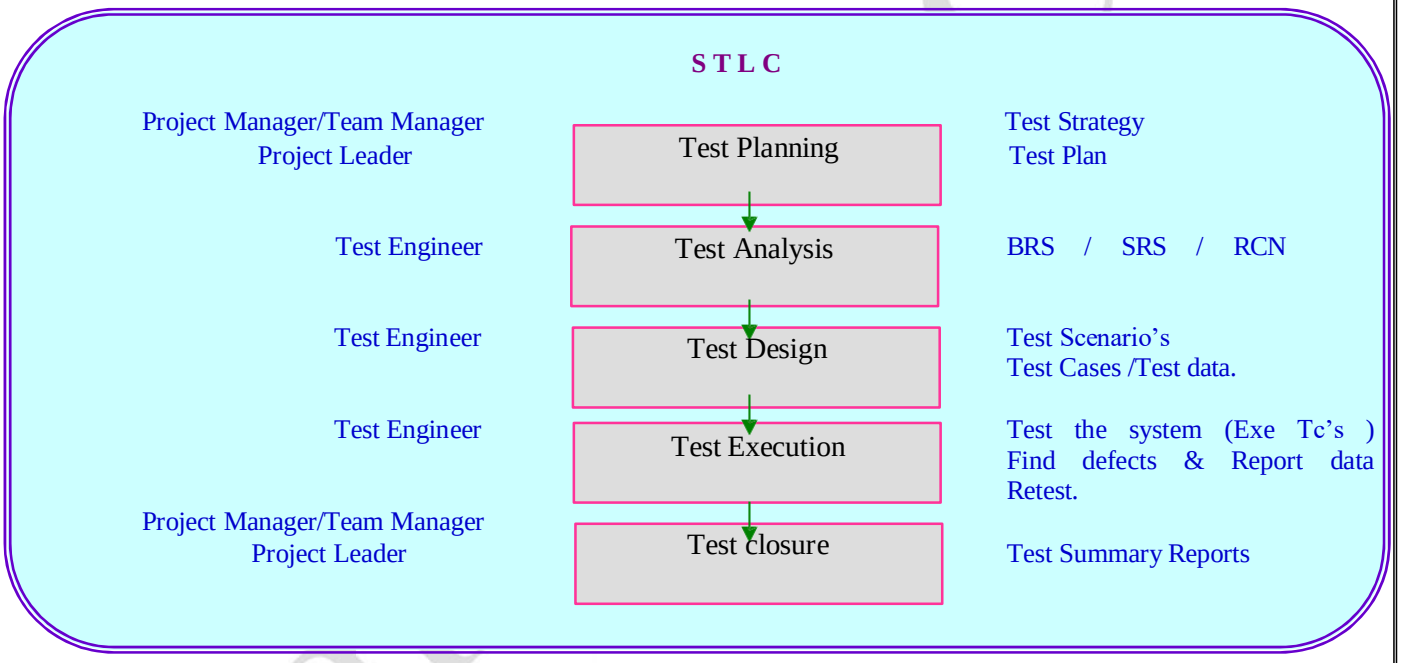
**Globalization Testing:** checking if t he application having a provision of setting and changing languages date and time format and currency etc. If it is designed for global users

**Localization Testing:** checking default languages currency date and time format etc. If it is designed for a particular locality of

**Installation Testing:** checking if we are able to install the software successfully or not as per the guidelines given in installation document

**Un-Installation Testing:** checking if we are able to uninstall the software from the system successfully or not

**Compatibility Testing:** checking if the application is compatible with the different software and hardware environment




**Test Planning:** It is the first phase of system testing where the management plan the high level and detail testing activities to be carried out.

Project manager or Test Manager will define test strategy and based on this test strategy the detail plan called test plan is prepared.

**Test Strategy:** It is high level management plan and approach that provides sufficient confidence on the project being tested.

**Test Plan:** It is detail day to day work plan that explains scope, approach resources and work schedules etc. Test plan can be prepared by test lead.

**Test Plan Template:**

1) Introduction
2) Scope
   2.1) In Scope
   2.2) Out of scope
3) Test Approach
4) Resources
5) Work Schedule
6) Entry Criteria & Exit Criteria
7) Test deliverables.

****Interview Question important****

**Objectives of a Test Plan:** a test plan document explains various procedures to be carried out while testing a software. The main object to test plan is to determine:

1) **Scope of Testing**: what are the features to be tested and what are the features not to be tested. Similarly what types of testing is applicable and what type of testing not applicable.
2) **Test Approach**: various guidelines to be followed while designing test cases, test data and defect codes etc.
3) **Resources:** Hardware, Software and human resources.
4) **Work schedules and Test delivery.**

**Test Analysis:** In this phase test engineer will study various project requirement documents such as BRS, SRS to understand the project requirement.

- While analyzing test requirements if there are any questions those will be recorded in requirement clarification note, once analysis is finished you can get the clarification from the domain experts.

**BRS Template:** A BRS document contains the following sections.

1) Client introduction
2) Project Introduction
3) Existing system
4) Drawbacks in the existing system
5) Proposed system

6) Project Architecture.

**SRS / FRS :** A typical SRS/FRS document will contain the following sections.

1) Overview
2) Prototype
3) Page elements
4) Use case diagram
5) Use case

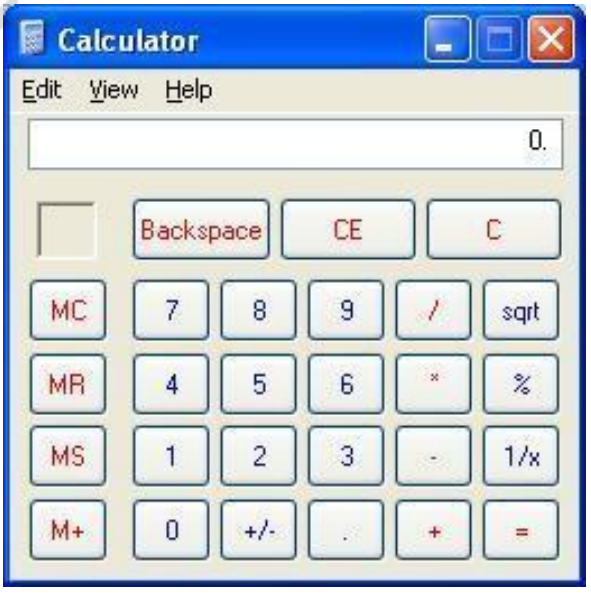
| | | | | |
|---|---|---|---|---|
| **Project Name : Primus Banking** | | | **Reviewed By:** | |
| **Module Name: Customer** | | | **Reviewed Date:** | |
| **Document: Reference Primus Banking-Admin-FRS-V1.0** | | | **Approved By:** | |
| **Author: Ch.Damodar** | | | **Approved Date:** | |
| **Created date: 24/07/2011** | | | | |
| **Requirement Specification Reference** | **Clarification Requirement** | **Clarification Provided** | **Clarification Provided By** | **Clarification Provided Date** |
| FRS-PgNo-15-Sec.No.3.2.1 | Services charges for Amount Below balance 1000 is not clear | Amount below Rs 1000/- is not transferable | **Ch.Damodar** | 24/07/2011 |
| | | | | |

**RCN Template:** (Requirement Clarification Note) :- is use to record your questions while analyzing the project requirement.

In a meeting you discuss about your questions with a subject matter experts and you get a clarification for the same.

**Test Design:** In this phase test engineer will prepare various documents such as test scenario's test cases etc. which helps in testing a software application effectively.

**Test Scenario:** An item or a feature or a functionality to be tested in the application under test is called test scenario.



## Test Scenario of a Calculator Application

**Scenario1:** Check Addition

**Scenario2:** Check Subtraction

**Scenario3:** Check Multiplication

**Scenario4:** Check Division

## Test Scenario of a Mobile Application

**Scenario1:** Check switch On /SwitchOff

**Scenario2:** Check Making a call

**Scenario3:** Check Receiving a call

**Scenario4:** Check sending sms

**Scenario5:** Check receiving sms

******Note :***

***Once the test scenarios are prepared, reviewed and approved based on these scenario's test cases will be prepared.***

| | | | |
|---|---|---|---|
| **Project Name : Primus Banking** | | | **Reviewed By:** |
| **Module Name: Admin** | | | **Reviewed Date:** |
| **Document: Reference Primus Banking-Admin-FRS-V1.0** | | | **Approved By:** |
| **Author: Ch.Damodar** | | | **Approved Date:** |
| **Created date: 25/07/2011** | | | |
| **Scenario ID** | **Module** | **Requirement** | **Test Scenario** |
| TS001 | Admin | R1.0 Primus Home Page | **Check Launching Application** |
| TS002 | Admin | R1.0 Primus Home Page | **check Navigation of Home Page** |
| TS003 | Admin | R1.0 Primus Home Page | **Check Admin Login** |
| TS004 | Admin | R1.0 Primus Home Page | **Check Bank Employee Login** |
| TS005 | Admin | R1.0 Primus Home Page | **Check Customer (individual, Corporate, NRI** |
| TS006 | Admin | R2.0 Admin Home Page | **Check Navigation of Admin Home Page** |
| TS007 | Admin | R2.0 Admin Home Page | Check Change Password |
| TS008 | Admin | R3.0 Branches | **Check New Password creation** |
| TS009 | Admin | R3.0 Branches | **Check Branch Modification** |
| TS010 | Admin | R3.0 Branches | **Check Branch search for different search criteria** |
| TS011 | Admin | R3.0 Branches | **Check for Branch deletion.** |

**Test Case:** A Test Case is set of preconditions steps to be followed with input data and expected behavior to validate functionality a system.

In simple words a test case is brief description of what to test and how to test.

******Note :***

***In Industry practically we prepare only functionality test cases.***

Three types of functional test cases:

1) Positive test cases 2) Negative test cases 3) Business validation test cases

**Positive Test Case:** A Test case prepared in a positive perception to check what a system suppose to do is called positive test case.

**Example**:-

Tc1: Check Login with valid inputs.

**Negative Test Case:** A Test case prepared to check what a system not suppose to do is called negative test case.

**Example**:-

Tc1: Check Login with invalid inputs.

**Business Validation Test case:** A test case prepared to check business conditions is called business validation test cases.

**Example**:-

Check fund transfer with 1000, 10000, 15000, etc is called business validation test case.

******Interview Question important******

**What is a good test case?**

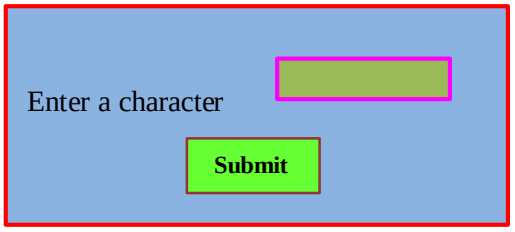
A test case that has high probability of catching defects is called a good test case.

**Test case design techniques:** Black box testing introduces the following techniques which help in preparing effective test cases to validate system functionality:

1) Equivalence class partitioning (ECP)
2) Boundary value analysis (BVA)
3) Decision table
4) State transition testing
5) Usecase testing.

**Equivalence class partitioning (ECP):** According to ECP at first identity all possible test cases to validate functionality in the system. Then segregate these test cases into groups. While making a group make sure that all test cases that belong to a group are producing the same out put then select one test case from each group, preferably middle one for testing.

Example:



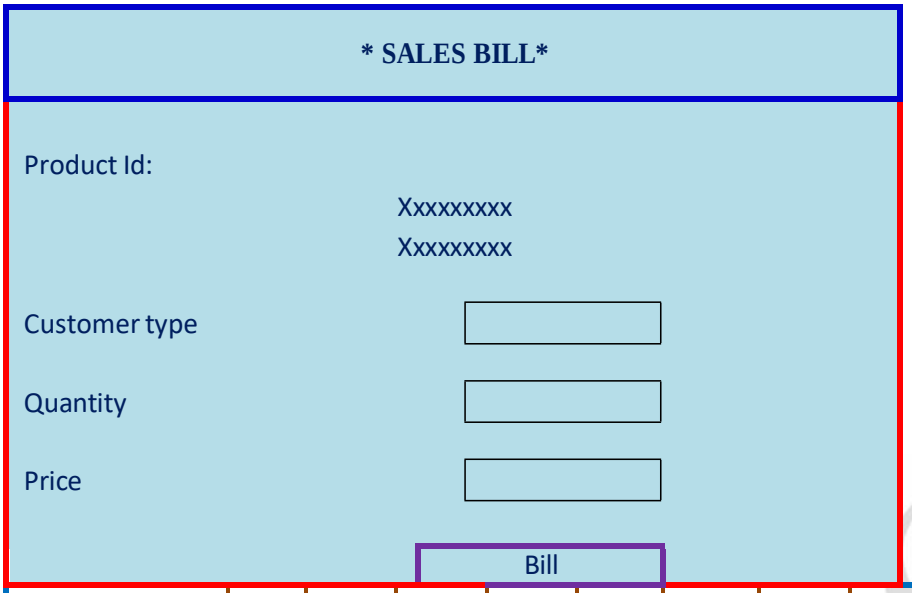

**Valid Partitions** **Invalid Partitions**

| Lowercase Alphabets | Uppercase Alphabets | Numeric | Spl chars | Null | >1 chars |
|---|---|---|---|---|---|
| TC1 a | TC27 A | TC53 0 | * | Blank | a |
| TC2 b | TC28 B | TC54 1 | - | | ab |
| TC3 c | TC29 C | TC55 2 | & | | abc |
| TC4 d | TC30 D | TC56 3 | % | | abcd |
| . | . | . | @ | | . |
| . | . | . | ^ | | . |
| . | . | . | ( | | . |
| . | . | . | ) | | abcdefg |
| . | . | . | + | | . |
| . | . | . | \| | | . |
| TC24 x | TC50 X | . | ! | | . |
| Tc25 y | TC51 Y | . | ~ | | . |
| TC 26 z | TC52 Z | TC62 9 | ` | | . |

**Boundary Value Analysis:** It has observed that most of the times programmers are committing mistakes while specifying the boundary conditions to determine these defects BVA is helpful.

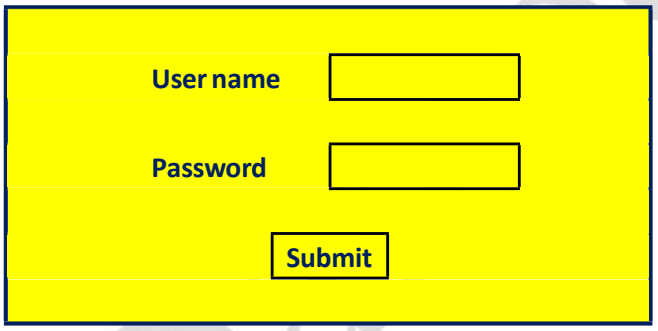
According to BVA once equivalence partitions are defined, identify the partition where there are inputs with range. Determine outer boundary an inner boundaries for this range. For positive negative testing consider lower boundary value minus one and upper boundary value plus one.

**Decision table testing:** Decision table testing is helpful to derive the test cases if functionality is depending on multiple inputs.




| Inputs | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 |
|---|---|---|---|---|---|---|---|---|
| Customer Type | yes | Yes | yes | yes | no | no | no | no |
| Quantity | yes | No | yes | no | yes | no | yes | no |
| Price | yes | Yes | no | no | yes | yes | no | no |
| Expected Result | | Error | error | error | error | error | error | error |

**For example**:




As login depending on two inputs i.e., uid,pwd to check login we can cover a following test cases

| Inputs | TC1 | TC2 | TC3 | TC4 |
|---|---|---|---|---|
| User Name | yes | Yes | No | no |
| Password | yes | No | yes | no |
| Submit | yes | Yes | No | no |
| | | Error | Error | Error |

If functionality is depending on more number of inputs we can reduce the test cases based on the application design

**State transition testing:**

*) Every software will have various possible states. The state of the application changes from one to another based on the user actions under input supplied.

*) State transition testing is helpful to derive and prepare test cases in order to cover all possible states of the application.

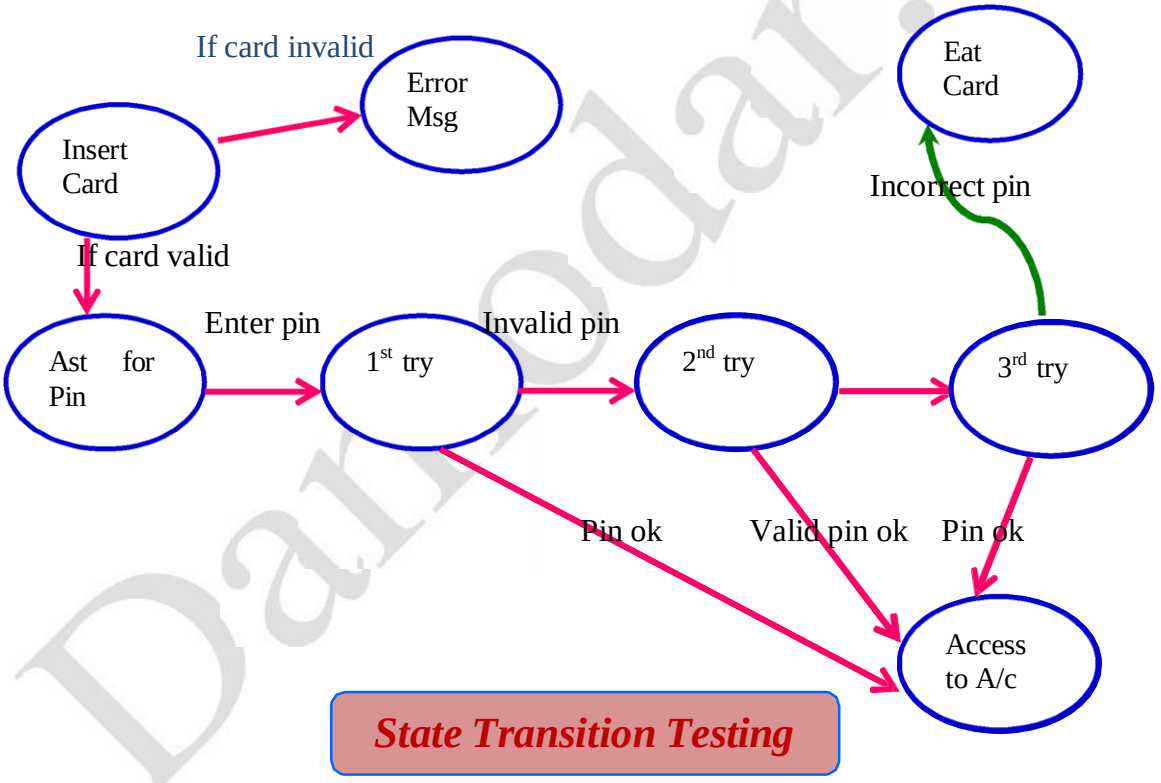
**Usecase testing:** Validating software to confirm whether it is develop as per the use cases or not is called user case testing.

**Requirement traceability Matrix (RTM):** Mapping between test cases and requirements is called traceability mapping.

**Advantages of traceability Matrix**:

*) To determine the percentage of test coverage.

*) To identify a group of test cases belongs to a requirement which helps in modify build testing and also implementing the change request easily




**Defect Reporting:** once a defect is identified we have to document the defect & report the same to developer

Severity - Stops the system execution.

Runtime error

Showstopper Severity - Stops the system execution.

Runtime error
Showstopper } High severity

Once function
It's not working } High severity

Cosmetic
Bugs } Low severity

Font, Colors
Test, etc. }

Defect Age: The time gap between date of detection & date of closure I known as the defect age

**Defect Severity:** How serious the problem in the system or the impact of the problem in the system is called defect severity.

Very high - fatal-S0

High - Major-S1

Medium - Minor-S2

Low – Low-S3

**Showstopper defect:** A defect which is not permitting to continue further test is called shostopper defect.

**Defect Priority:** The order in which the defect is to be resolved is called defect priority.

******Note :***

****) Defect severity may be specified by the tester & defect priority will be assigned by manager***

****) In general defect severity and priority's proposition to each other but in some situation that may vary.***

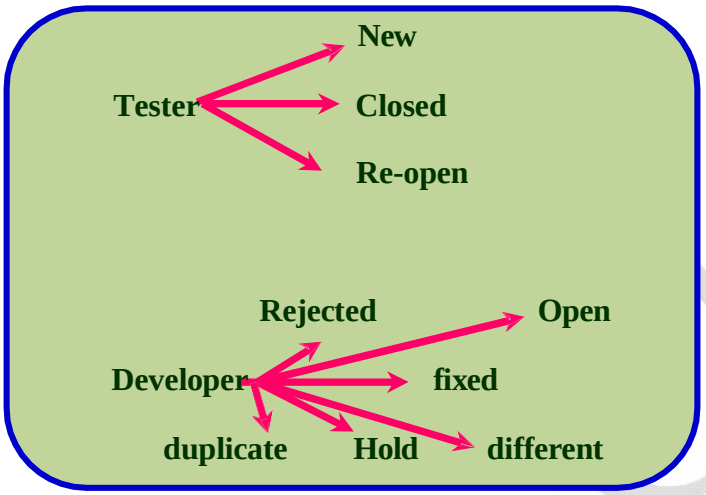
*) An example of a defect that takes low severity and high priority.

- Incorrect company logo and incorrect company title.

*) An example for a defect that takes high severity and low priority - A serious problem belongs to inter release module.

**Defect Age:** The time interval between date of detection and date of closure is called is defect age.

*) In other words how long a bug is present in the development life cycle is called defect age.




**Test Closure:** It is the last phase of SDLC where the management prepares various test summary reports that explains the complete statistics of the project.

**Common Repository:** Work Product:

A centralized computer system where you stored and mange all project resources is called common repository.

**Configuration Management:** Maintaining all project resources in a centralized system, maintaining appropriate version controlling based on the changes made to them is called configuration management.

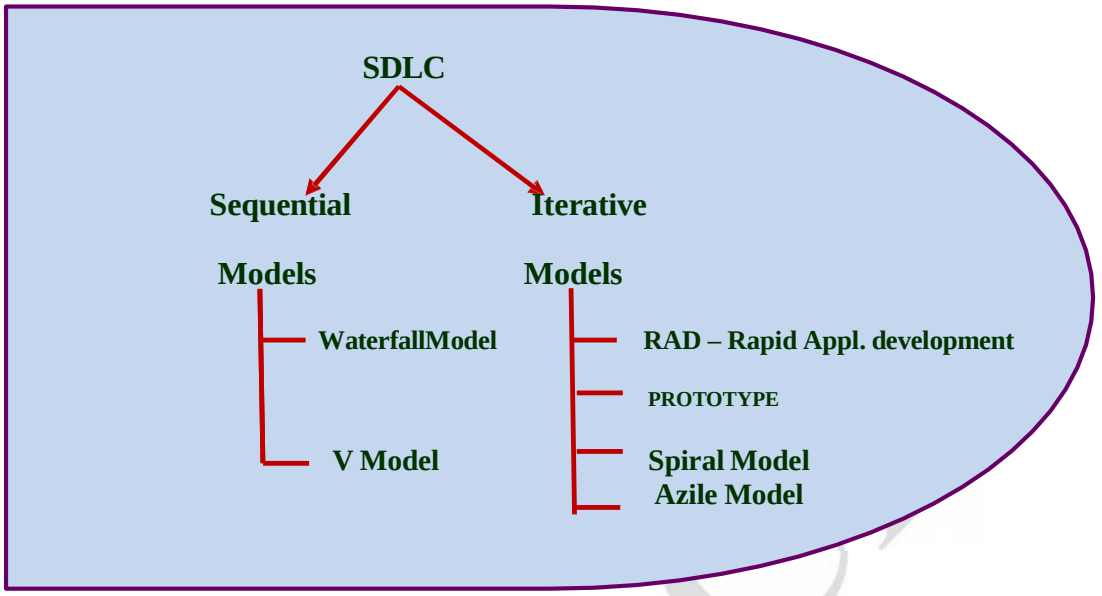
**Software Configuration Tools:**

*) VSS (Visual Source Safe)

*) CVS (Concurrent Version System (open source))

******Note:***

****) Quality center test management tools also provide configuration management services.***




**A SDLC model explains how development carried out in project implementation.**

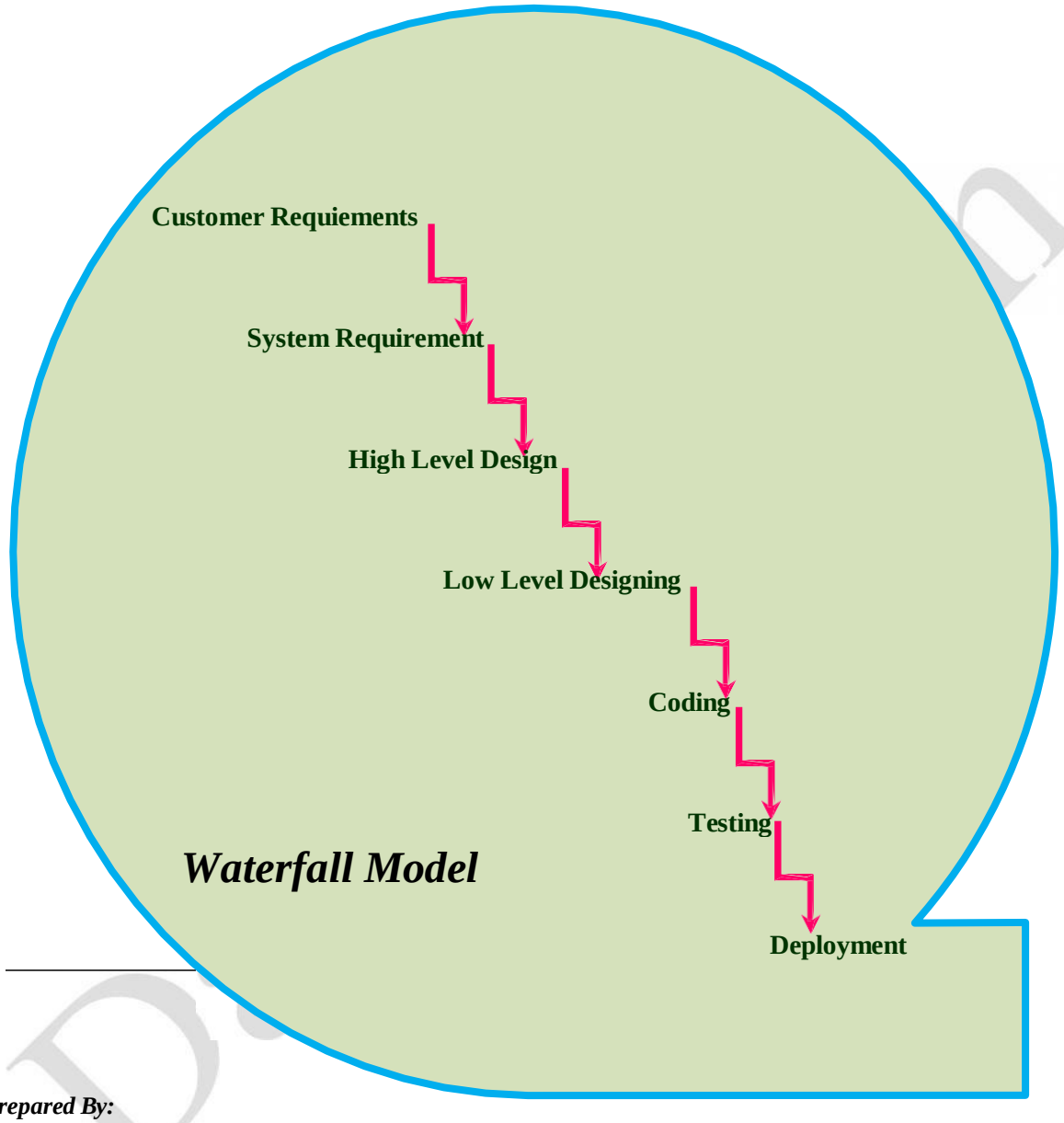
**Sequential Models:** These models are best suitable for small size of applications where all development activities are carried out in a sequential order for the entire project.

**Iterative Models:** These models are best suitable for big projects in which a big project will be divided into modules then the application will be implemented module by module.

**Waterfall Model:** It is a beginning approach of development model where all development activities are carried out one after another for the entire project.

***) In this approach tester conduct testing after coding i.e. only validation.**

***) As flow of all activities look like a waterfall is called waterfall model.**




***Prepared By:***
**Chindam Damodar**
chdamu2002@gmail.com

**Published by: http://SoftwareTestingHelp.com**

**For software testing career advice, jobs, automation testing guide, testing templates and latest news please visit: http://Softwaretestinghelp.com**

**Thanks,**
**Vijay**