MANUAL TESTING

INDEX: PAGE No:

MANUAL TESTING

What is MANUAL TESTING?

MANUAL TESTING is a process, in which all the phases of STLC (SOFTWARE TESTING LIFE CYCLE) like Test planning, Test development, Test execution, Result analysis, Bug tracking and Reporting are accomplished successfully and manually with Human efforts.

Why did U choose Testing?

- Scope of getting jobs is very very high.
- No need to depend upon any Technologies.
- Testing there for ever.
- One can be consistent throughout their life.

Who can do Testing?

Any graduate who is creative can do.

What exactly we want to get a job?

Stuff+communications+confidence+dynamism.

Why the Test engineers exclusively required in the software companies?

- One cannot perform two tasks efficiently at a time.
- Sentimental attachment.

Project: Project is something that is developed based on particular customer's requirements and for their usage only.

Product: Product is something that is developed based on the company specifications and used by multiple customers.

Note: The product based company will first have general survey in the market. Gather's clear requirements from different customers, and based on common requirements of so many customer`s. They will decide the specifications (Requirements).

Quality:

Classical Definition of 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.

Latest Definition of Quality:

Quality is defined as not only the justification of all the requirements but also the presence of the value (User friendliness).

Defect: Defect is defined as a deviation from the Requirements.

Testing: Testing is a process in which defects are identified, isolated, subjected for rectification and ensure that the product is defect free, in order to produce the quality product and hence the customer satisfaction. (or) Verification & Validation of software is called Testing.

Bidding: The project is defined as a request for proposal, estimation and signing off.

Kick off meeting: It is an initial meeting conducted in the software company, soon after the project is signed off, in order to discuss the overview of the project and also to select a project manager.

Usually Project managers, Technical managers, Quality managers, High level management, Test leads, Development leads and sometimes customer representatives will be involved in this meeting.

Note: Apart from this meeting any kind of startup meeting during the process can be considered as 'Kick off Meeting.

Project Initiation Note (PIN): It is a mail prepared by the project manager and sent to CEO of the software company as well as for all the core team members in order to intimate them, that they are about to start the actual project activities.

Software Quality:

Technical:

- Meeting Customer Requirements
- Meeting Customer Expectations (User friendly, Performance, Privacy)

Non-Technical:

- Cost of Product
- Time to Market

Software Quality Assurance: To monitor and measure the strength of development process, Organization follows SQA concepts.

Software Project: Software related problems solved by software engineers through a software engineering process.

Software Development Life Cycle (SDLC):

There are six phases in software development life cycle

1. **Initial (or) Requirements phase**
2. **Analysis phase**
3. **Design phase**
4. **Coding phase**
5. **Testing phase**
6. **Delivery and Maintenance phase**

I. Initial (or) Requirement phase:

Tasks: Interacting with customer and gathering the requirements.
Roles: Business analyst –BA, Engagement manager - EM
Process:

- First of all business analyst will take an appointment from the customer, collect the template from the company and meet the customer on appointed date, gather the requirements with the support of that template and comes back to the company with the requirement document.
- The engagement manager go through the requirements, if he find any extra requirements then he will deal with excess cost of the project.
- If at all he finds any confused requirements, then he will ask the concern team to build a prototype, demonstrate that prototype to the customer, gather's the clear requirements and finally hand over the required documents to the BA

Proof: The proof document of Initial phase is Requirement's Document (RD).

These documents are called different Names in different companies:

FRS: Functional Requirement Specifications
CRS: Customer (or) Client Requirement Specifications
URS: User Requirement Specifications
BRS: Business Requirement Specifications
BDD: Business Design Document
BD: Business Document

Some companies will maintain two documents. One is for overall business flow information and second is detailed functional information, but some companies will maintain both this information's in a single document.

Template: Template is pre-defined format, which is used for preparing a document very easily and perfectly.

Prototype: Prototype is a roughly and rapidly developed model which is used for demonstrating to the client in order to gather clear requirements and also to build the confidence of a customer.
Ex: Power Point slide show.

II. ANALYSIS PHASE:

Tasks:
- Feasibility study
- Tentative planning
- Technology selection and Environment confirmation
- Requirement analysis

Roles: System Analyst – SA, Project Manager – PM, Technical Manager - TM
Process:

a. **Feasibility study**:
It is detailed study conducted on the requirement documents, in order to confirm whether the given requirements are possible within the given budget, time and available resources or not.

b. **Tentative planning:**
In this section resource planning and time planning will be temporarily done.

c. **Technology selection & Environment confirmation:**
The list of all technologies required for accomplishing the project successfully will be analyzed, the environment suitable for that project will be selected and mentioned here in this section.

d. **Requirement analysis:**
The list of all the requirements that are required by the company to accomplish this project successfully will be analyzed and listed out clearly in this section.

Note: Requirements may be Human Resources, Software's, and Hardware's.
Proof: The proof document of the Analysis phase is System Requirements Specifications (SRS).

III. DESIGN PHASE:

Tasks:
- High level designing
- Low level designing

Roles:
- High level design is done by chief Architect(CA)
- Low level design is done by Technical Lead(TL)

Process:
- The chief architect will divide the whole project in to modules by drawing some diagrams using unified modeling language (UML).
- The Team lead will divide the modules into sub modules by drawing some diagrams using the same UML.
- In this phase they will also design GUI part of the application, as well as PSEUDO CODE also developed

Proof: The proof document of this phase is Technical Design Document (TDD).
LLD: Ex**:** DFD-Data Flow Diagram, E-R Diagram, Class Diagram, Object Diagram.

PSEUDO CODE: PSEUDO Code is a set of English instructions, which will make the developer's more comfortable, while developing the actual source code.

IV. CODING PHASE (WHITE BOX TESTING):

Tasks: Programming (or) Coding
Roles: Programmers (or) Developers
Process: The developers will develop the actual source code by following the coding standards and also with the support of Technical Design Document.

Example's for Coding standards:

- Proper Indentation (left margin)
- Color codling's
- Proper Commenting

Proof: Proof document of the Coding phase is Source Code Document (SCD).

V TESTING PHASE (BLACK BOX TESTING):

Tasks: Testing
Roles: Test Engineer's.
Process:

- The Testing department will receive the requirement document and the test engineers will start understanding the requirements.
- While understanding the requirements, if they get any doubts they will list out all the doubts in Requirement clarification Note and sent it to the author of the requirement document and wait for the clarification.
- Once the clarification is given and after understanding all the requirements clearly they will take the test case template and write the Test cases.
- Once the first build is released they will execute the test cases.
- If at all any defects are found they will list out all the defects in the defects profile and send it to the development department and will wait for the next build.
- Once the next build is released they will re execute the required test cases.
- If at all any more defects are found they will update the defect profile. Send it to development department and will wait for the next build.
- This Process continuous till the product is defect free.

Proof: The proof of Testing phase is Quality product.

BUILD: A finally intigrated all modules set .EXE form is called Build.

TEST CASES: Implementing the creative Ideas of the Test Engineer on the application for testing, with the help of requirement document is known as TEST CASES.

VI. DELIVERY AND MAINTENANCE PHASE:

Delivery:
Tasks: Hand over the Application to the client
Roles: Deployment engineers (or) Installation engineers.
Process: The Deployment engineers will go to the customers place, install the application in to the customers environment and handover the original software to the client.
Proof: The final official agreement made between the customer and company is proof document for Delivery.

Maintenance:

Once the application is delivered. The customer will start using it, while using if at all they face any problems then that particular problem will be created as tasks. Based on the tasks corresponding roles will be appointed. They will define the process and solves the problem .This process is known as Normal Maintenance. But some customers may request for continuous Maintenance, in that case a team of members will be continuously working in the client site in order to take care of their Software.

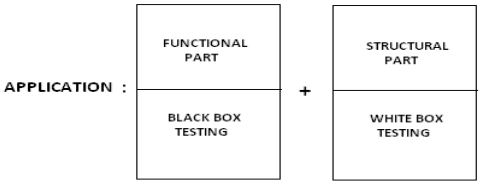
Where exactly testing comes in to the picture?
Which sort of testing we are expecting?
How many sort's of testing are there?

There are two sorts of Testing.

1. **Unconventional Testing**
2. **Conventional Testing**

Unconventional testing: It is sort of Testing in which the Quality Assurance people will check each and every out come document is according to the company standards or not right from the Initial phase to the end.

Conventional testing: It is sort of Testing in which one will check the developed applications or its related parts are working according to the exceptions or not, from the Coding phase to the end. Usually Quality Control people will do Conventional testing.



Testing methodology (Testing Techniques):

Basically there are 2 methods of Testing:

1. **Black Box Testing**
2. **White Box Testing**

Note: One more derived method is Grey Box Testing

BLACK BOX TESTING:

- It is method of testing in which one will perform testing only on the function part of the application without having the knowledge of structural part.
- Usually the Black Box Test engineers will perform.

WHITE BOX (or) GLASS BOX (or) CLEAR BOX TESTING:

- It is a method of testing in which one will perform testing on the structural part of the application.
- Usually the White Box Tester's or Developer's will perform.

GREY BOX TESTING:

It is method of testing in which one will perform testing on both the functional part as well as structural part of on application.

- Usually the Test engineer's who has the knowledge of structural part will perform.

LEVELS OF TESTING:

There are 5 levels of Testing:

1. **Unit level testing**
2. **Module level testing**
3. **Integration level testing**
4. **System level testing**
5. **User acceptance level testing**

1. UNIT LEVEL TESTING:

Unit: Unit is a smallest part of an application (Program).

In this stage the white box testers will test each and every program and combinations of programs in order to confirm whether they are working according to the expectations or not. They test the structural part of a module.

2. MODULE LEVEL TESTING:

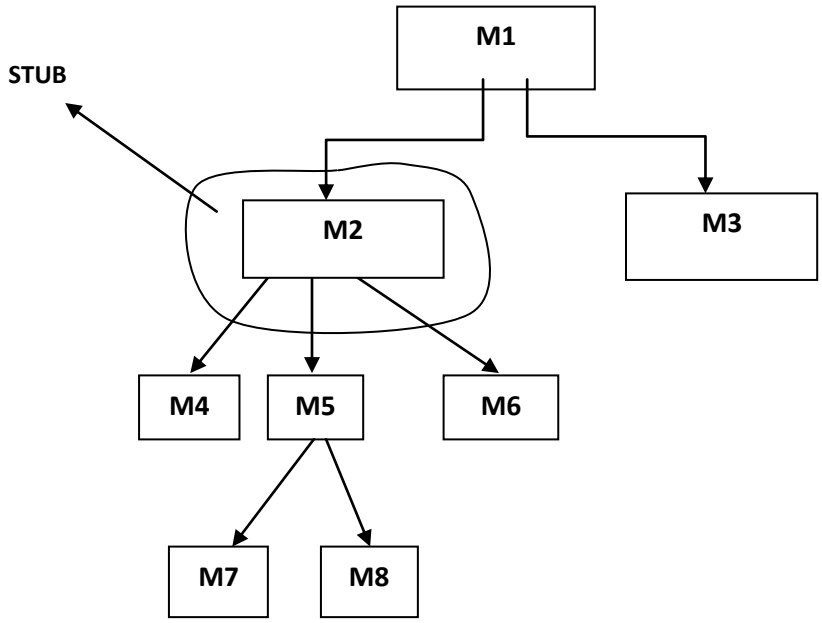
Module: Module is defined as a group of related features to perform a major task in an application.

In this stage the Black Box test engineer's will test the functional part of a module.

3. INTEGRATION LEVEL TESTING:

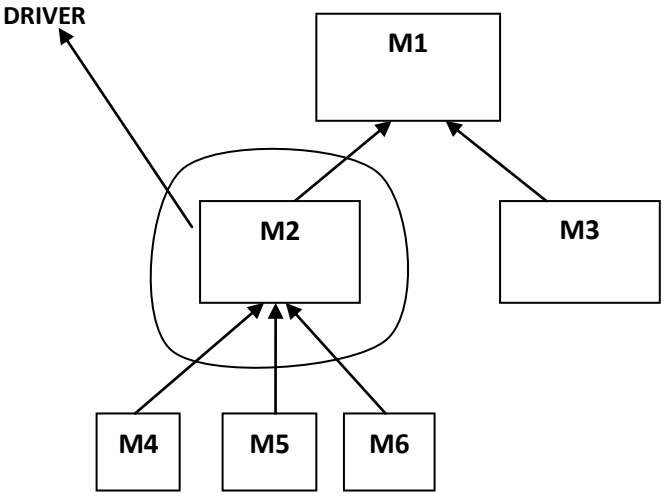
In this stage the developers will develop interfaces (Linking Prg's), in order to integrate the modules. The White Box testers will test whether the interfaces are working fine or not. Developers will integrate the modules by following any one of the following approach:

TOP-DOWN APPROACH: In this approach parent modules will be develop first and then related child modules will be integrated.



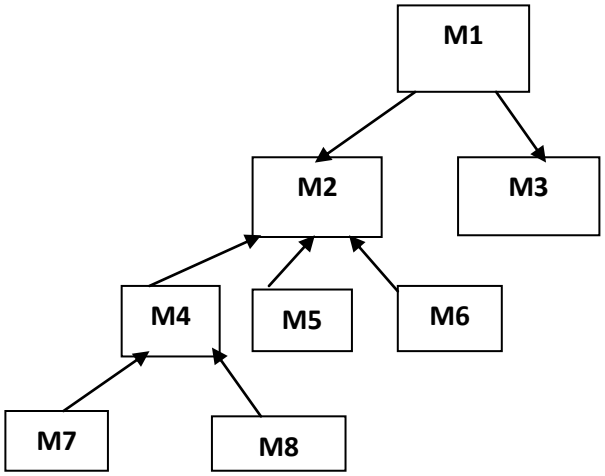
STUB: While integrating the modules in the Top-Down approach, if at all any mandatory module is missing then that module is replaced with a temporary program known as STUB.

BOTTOM-UP APPROACH: In this approach child modules will be developed first and then integrated back to the corresponding parent modules.

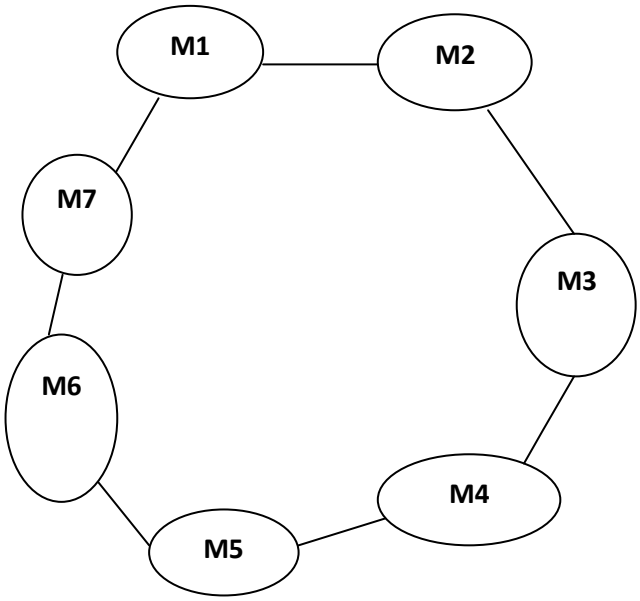


DRIVER: While integrating the modules in Bottom-Up approach, if at all any mandatory module is missing then that module is replaced with a temporary program known as DRIVER.

SANDWICH (OR) HYBRID APPROACH: This is a mixture of both Top-Down and Bottom-Up approach.



BIG BANG APPROACH: In this approach one will wait till the modules are developed and will integrate them at a time finally.



4. SYSTEM LEVEL TESTING:

In this level the Black Box test engineers will conduct so many types of testing like load testing, performance testing, stress testing, compatibility testing, system integration testing etc.

These type of Testings are also conducted:

1. Usability Testing
2. Functionality Testing

→ **CORE LEVEL**

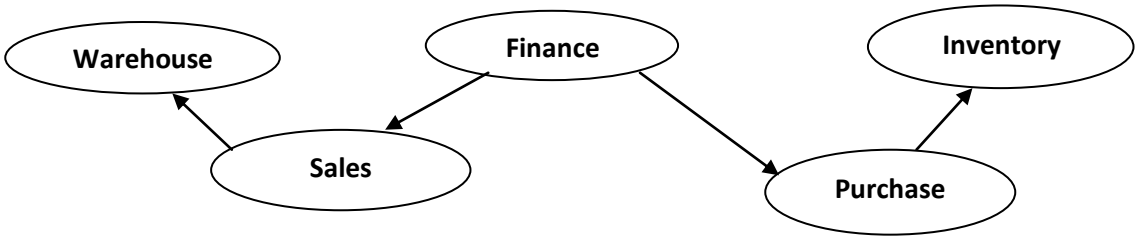
3. Performance Testing
4. Security Testing

→ **ADVANCE LEVEL**

During Usability Testing, testing team validates **User Friendliness** of screens.
During Functionality Testing, testing team validates **Correctness of Customer Requirements**.
During Performance Testing, testing team estimates **Speed of Processing**.
During Security Testing, testing team validates **Privacy to User Operations**.

SYSTEM INTEGRATION TESTING: It is a type of testing in which one will perform some actions at one module and check for the reflections in all the related areas.

Ex:



5. USER ACCEPTANCE TESTING:

In this stage the Black Box test engineers will test once again the user desired areas in the presence of the user in order to make him to accept the application.

ENVIRONMENT: Environment is defined as group of hardware components with some basic software's which can hold the business logic, presentation logic and database logic.

(Or)

Environment is a combination of Presentation layer, Business layer, and Database layer which can hold presentation logic, business logic and database logic.

TYPES OF ENVIRONMENTS:

There are 4 types of environments:

1. **STAND-ALONE ENVIRONMENT (OR) ONE-TIER ARCHITECTURE.**
2. **CLIENT-SERVER ENVIRONMENT (OR) TWO-TIER ARCHITECTURE.**
3. **WEB ENVIRONMENT (OR) THREE-TIER ARCHITECTURE.**
4. **DISTRIBUTED ENVIRONMENT (OR) N-TIER ARCHITECTURE.**

1. STAND-ALONE ARCHITECTURE:

In this environment all the three layers that is presentation layer, business layer, database layer will be available in the single tier. When the application needs to be used by a single user at a time then one can suggest this environment.

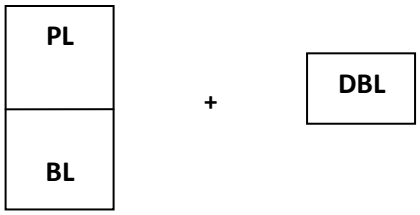
Ex: Personal Computer.

2. CLIENT-SERVER ENVIRONMENT:

In this environment two tiers will be there. One is for clients, other is for servers. Presentation layer and business layer will be available in each and every client; database layer will be available in the server.

Whenever the application need to be used by multiple users sharing the common data in a single premises and wants to access the application very fastly and there is no problem with security. Then one can suggest client-server environment.

Ex: LAN.



3. WEB ENVIRONMENT:

This environment contains three tiers. One is for clients, middle one is for application server and the last one is for database servers.

Presentation layer will be available in client, Business layer will be available in the application server and Database layer will be available in the database servers.

Whenever the application needs to be used all over the world by limited number of people then this environment can be suggested.

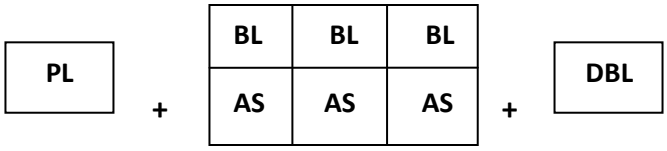
Ex: WAN.

4. DISTRIBUTED ENVIRONMENT:

This environment is similar to web environment but number of application servers are introduced in individual tiers. In order to distribute the business logic, so that load will be distributed and performance will be increased.

Whenever the application needs to be used all over the world by huge number of people then this environment can be suggested.

Ex: yahoo.co.in, yahoo.co.uk, yahoo.co.us….etc.



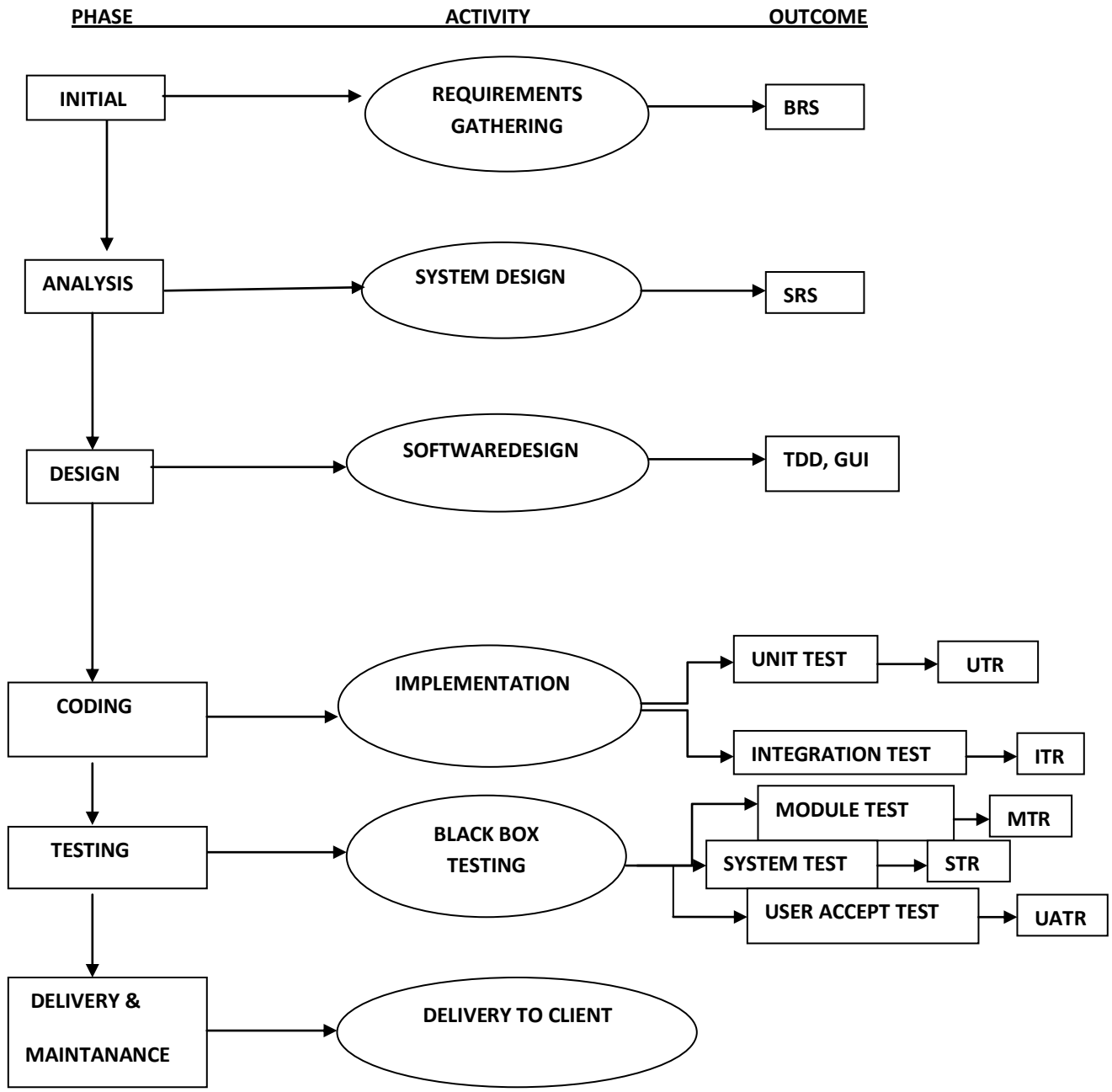
AS – APPLICATION SERVER
BL – BUSSINESS LOGIC

DATABASE: It is a base (or) a place where on can store and retrieve the data

SOFTWARE PROCESS DEVELOPMENT MODELS:

1. WATER FALL MODEL
2. PROTOTYPE MODEL
3. EVOLUTIONARY MODEL
4. SPIRAL MODEL
5. FISH MODEL
6. V-MODEL

1. WATERFALL MODEL:



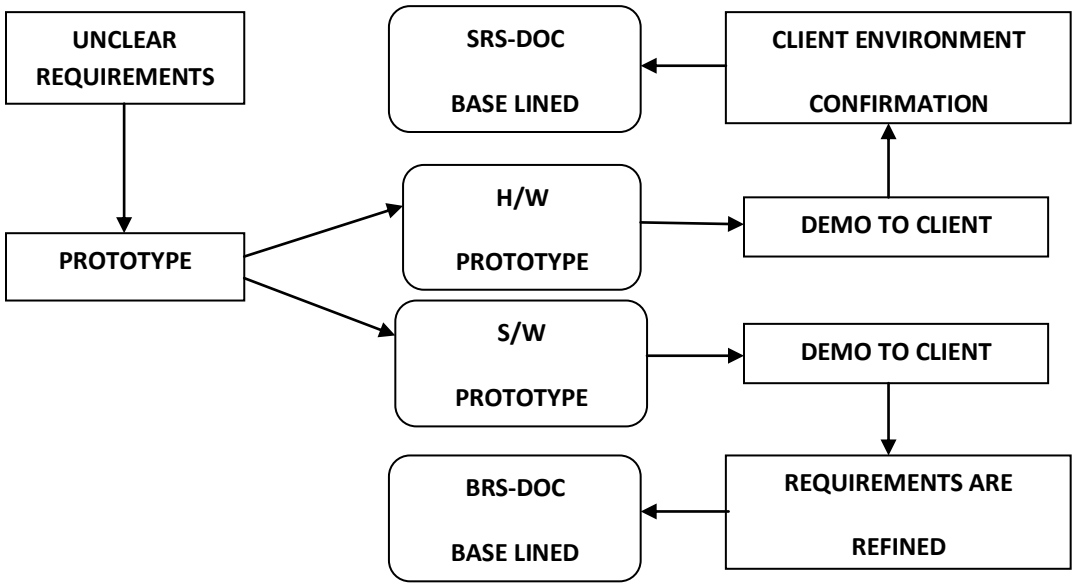
ADVANTAGES:

1. It is a simple model.
2. Project monitoring and maintenance is very easy.

DISADVANTAGES:

Can't accept the new requirements in the middle of the process.

2. PROTOTYPE MODEL:



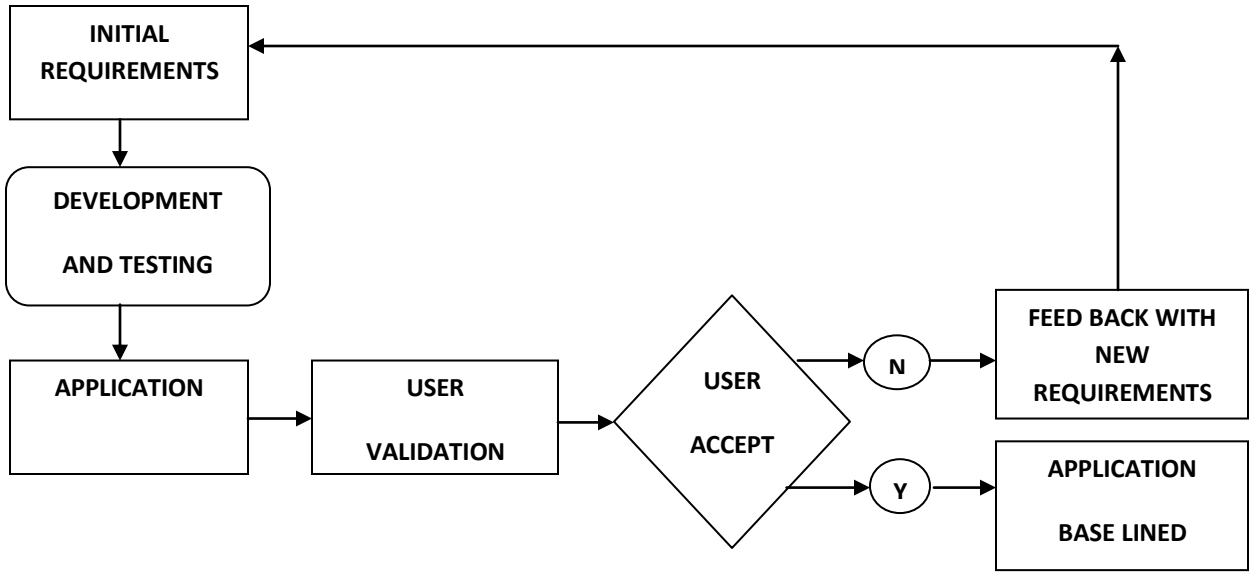
ADVANTAGES:

Whenever the customer's are not clear with their requirements then this is the best suitable model.

DISADVANTAGES:

a. It is not a full fledged process development model.
b. Prototype need to be build on companies cost.
c. Slightly time consuming model.
d. User may limit his requirements by sticking to the PROTOTYPE.

3. ENVIRONMENT MODEL:



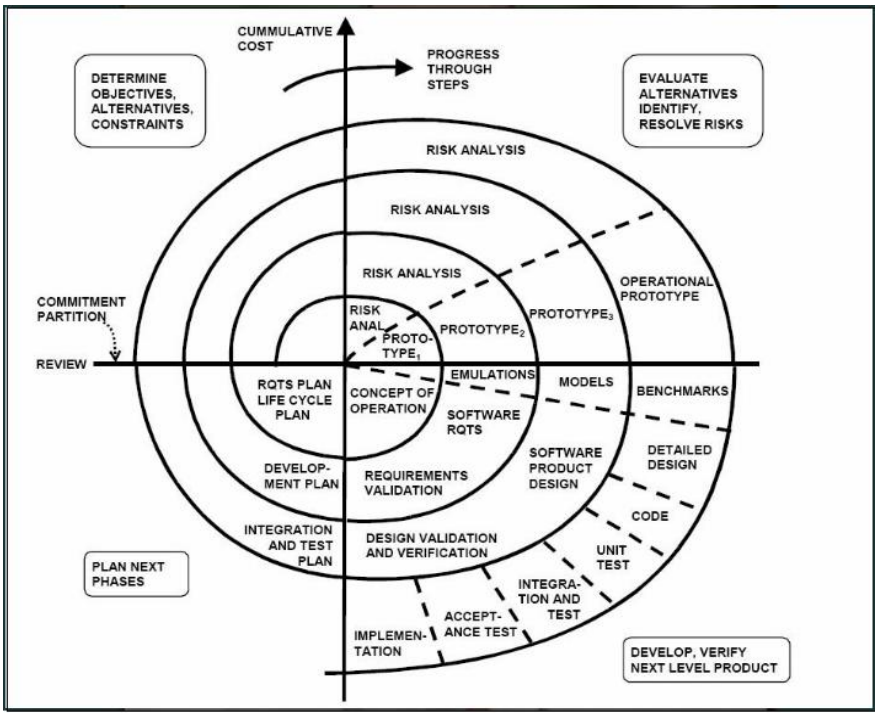
ADVANTAGES:

Whenever the customers are evolving the requirements then this is the best suitable model. Adding the new requirements after some period of time.

DISADVANTAGES:

1. Project monitoring and maintenance is difficult.
2. Can't define the deadlines properly.

4. SPIRAL MODEL:



Ex: Risk based scientific projects, Satellite projects.

ADVANTAGES:

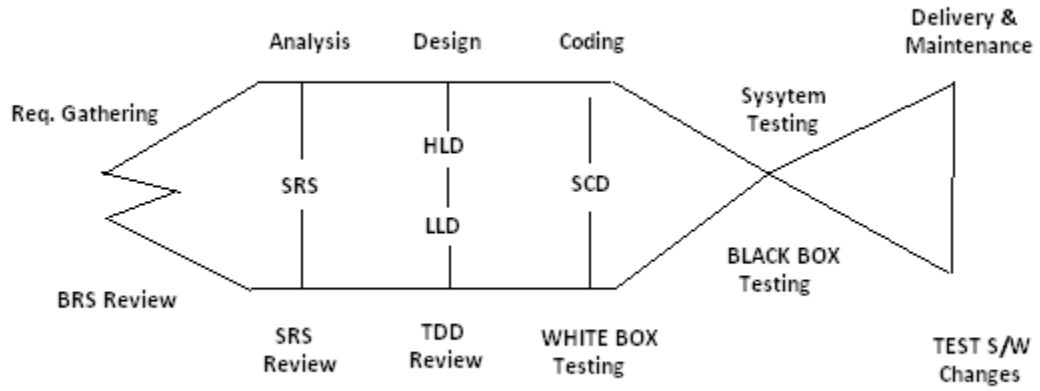
Whenever the project is highly risk based this is the best suitable model.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.
3. Risk route cause analysis is not an easy task.

NOTE: Cycles depend upon Risk involved in the project and size of the project, Every cycle has 4 phases, except the last phase.

5. FISH MODEL:



ADVANTAGES:

As both verification and validation are implemented the outcome will be a quality product.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.

VERIFICATION:

Verification is a process of checking each and every role in the organization in order to confirm weather they are working according to the company's process guidelines or not.

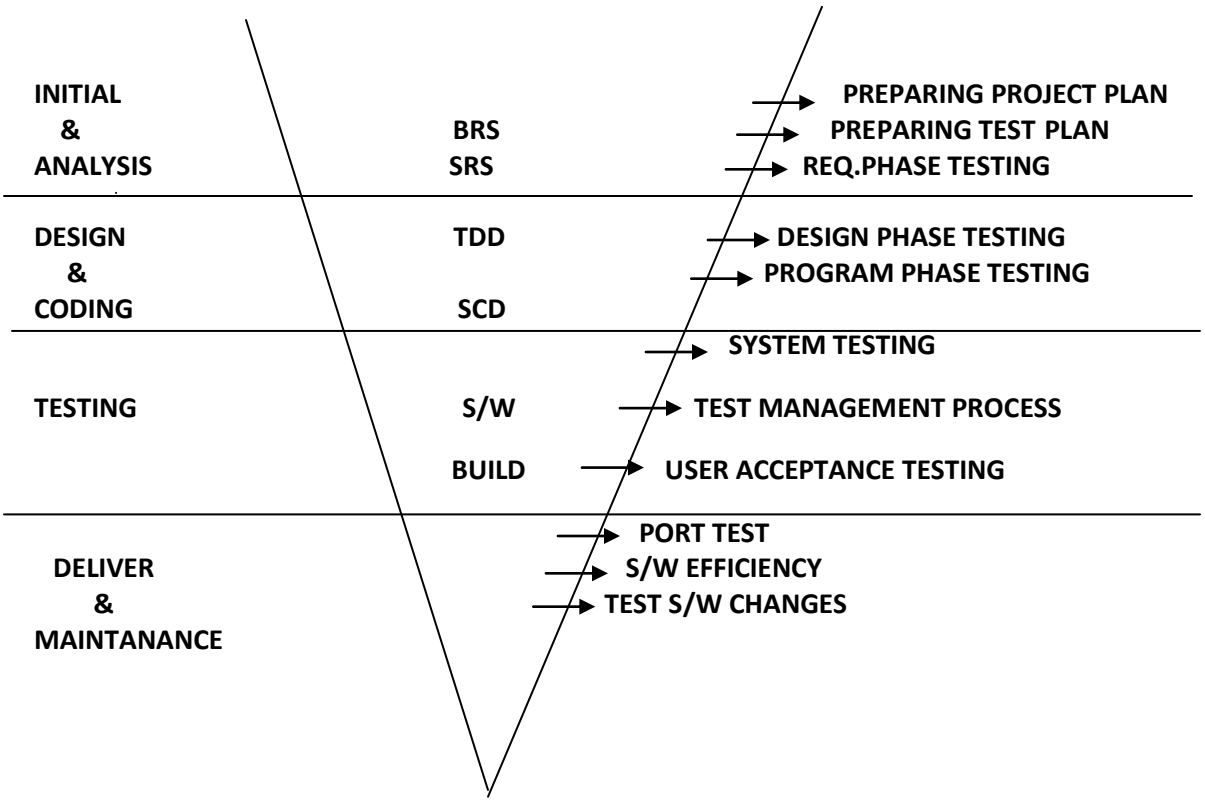
VALIDATION:

Validation is a process of checking, conducted on the developed product or its related parts in order to confirm weather they are working according to the expectations or not.

VERIFICATION: Quality Assurance people (Reviews, Inspections, Audits, Walk through).

VALIDATION: Quality Control people (Testing).

6. V-MODEL:



"V" Represents Validation and Verification.

DRE=0-1(Range).
DRE=A/A+B.
DRE = Defect Removal Efficiency.
A = Defects found by the Testing Team. .
B = Defects raised by the Customer.
DRE=80/80+20=80/100=4/5=0.8 (Good Software).

GOOD SOFTWARE :0.7-1
POOR SOFTWARE :Below 0.7

ADVANTAGES:

As verification, validation, test management process is maintained. The outcome will be a quality product.

DISADVANTAGES:

1. Time consuming model.
2. Costly model.

AGILE MODEL: Before development of the application, where testers write the test cases and gives to the development team, so that it can be easy for developers to develop defect free Programs.

TERMINOLOGY:

IF A DEVELOPER FINDS A MISTAKE IN CODE, WHILE DEVELOPING OF AN APPLICATION IT IS CALLED AN **ERROR**.
IF A TESTER FINDS A MISTAKE IN A BUILD, WHILE TESTING IT IS CALLED A **DEFECT** (or) **ISSUE.**
IF A DEFECT IS ACCEPTED BY DEVELOPER TO SOLVE IT IS CALLED A **BUG**.
IF A CUSTOMER FINDS ANY MISTAKES, WHILE USING THE APPLICATION IT IS CALLED A **MISTAKE** (or) **FAULT** (or) **FAILURE**.

A mistake in code is called **ERROR**. Due to errors in coding, test engineers are getting mismatches in application called **DEFECTS**. If defect accepted by development to solve called **BUG.**

TYPES OF TESTINGS

1. BUILD ACCEPTANCE TEST/BUILD VERIFICATION TEST/SANITY TESTING:

It is type of testing In which one will perform overall testing on the released build, in order to confirm whether it is proper for conducting detailed testing or not.

Usually during this type of testing they check the following:

- Whether the build is properly installed or not
- Whether one can navigate to all the pages of application or not
- Whether all the important functionality are available or not
- Whether all the required connections are properly established or not

Some companies even called this as SMOKE TESTING, but some companies will say that before releasing the build to the testing department, the developers will check whether the build is proper or not that is known as SMOKE TESTING, and once the build is released what ever the test engineers is checking is known as BAT or BVT or SAINITY TESTING (BAT: Build Acceptance Test, BVT: Build Verification Test).

2. REGRESSION TESTING:

It is type of testing in which one will perform testing on the already tested functionality again and again. Usually we do this in 2 scenarios:

- When ever the tester's identify the defects raise to the developers, next build is released then the test engineers will check defect functionality as well as the related functionality once again.
- When ever some new features are added, the next build is released to testing department team. Then the test engineers will check all the related features of those new features once again this is known as Regression Testing

Note: Testing new features for the first time is known as New testing it not the Regression testing.
Note: Regression testing starts from 2nd build and continuous up to last build.

3. RETESTING:

It is type of testing in which one will perform testing on the same funcatnality again and again with deferent sets of values, in order to confirm whether it is working fine or not.
Note: Retesting starts from 1st build and continuous up to last build.
Note: During Regression testing also Retesting will be conducted.

4. ALPHA TESTING:

It is type of user acceptance testing conducted in the software company by the test engineers just before delivering the application to the client.

5. BETA TESTING:

It is also a type of user acceptance testing conducted in the client's place either by the end users or third party experts, just before actual implementation of the application.

6. STATIC TESTING (Look and Feel Testing):

It is a type of testing in which one will perform testing on the application or its related factors without doing any actions.

EX: GUI Testing, Document Testing, Code Reviews etc…,

7. DYNAMIC TESTING:

It is a type of testing in which one will perform testing on the application or its related factors by doing some actions.

Ex: Functional Testing.

8. INSTALLATION TESTING:

It is a type of testing in which one will install the application in to the environment, following the guide lines provided in the deployment document(Installation Document), in order to confirm whether these guide lines are really suitable for installing the application into the environment or not.

9. PORT TESTING:

It is a type of testing in which one will install the application in to the original client's environment and check weather it is compatible with that environment or not.

10. USABILITY TESTING:

It is a type of testing in which one will test the user friendliness of the application.

11. COMPATABILITY TESTING:

It is a type of testing in which one will install the application into multiple environments, prepared with different configurations, in order to check whether the application is suitable with those environments or not.

Usually these types of testing will focused in product based companies.

12. MONKEY TESTING:

It is a type of testing in which one will perform abnormal actions on the application. Intentionally, in order to check the stability of the application.

13. EXPLORATORY TESTING:

EXPLORING: Having basic knowledge of about some concept, doing some thing and knowing more about the same concept is known as Exploring.

It is a type of testing in which the domain experts will perform testing on the application with out having the knowledge of requirements, just by parallel exploring the functionality.

14. END TO END TESTING:

It is a type of testing in which one will perform testing on the end to end scenarios of the application.

EX: Login---> Balance Enquiry ---> Withdraw ----> Balance Enquiry ---> Logout.

15. SECURITY TESTING:

It is a type of testing in which one will check whether the application is properly protected or not.

To do the same the BLACK BOX TEST Engineers will perform the following types of Testing:

1. **AUTHENTICATION TESTING:** In this type of testing one will enter different combination of user names and passwords and check whether only the authorized people are able to access application or not.
2. **DIRECT URL TESTING:** In this type of testing one will directly enter the URL's of secured pages, in order to check whether the secured pages are directly access or not with out login to the application.
3. **FIRE WALL LEAKEGE TESTING(or) USER PRIVILLAGES TESTING:** It is a type of testing in which one will enter in to the application as one level of user and will try to access beyond the user limits, in order to check whether the fire walls are working properly or not.

16. MUTATION TESTING:

It is a type of testing in which one will perform testing on the application are its related factors by doing some changes to them.

17. SOAK TESTING/REALIABILITY TESTING:

It is a type of testing in which one will use the application continuously for a long period of time, in order to check the stability of the application.

18. ADHOC TESTING:

It is a type of testing in which one will perform testing in their own style after understanding the requirements clearly.

Note: Usually in the final stages of the project, This type of Testing can be encouraged.

19. INPUT DOMAIN TESTING:

It is a part of Functionality Testing. Test engineers are maintaining special structures to define size and type of every input object.

20. INTER SYSTEM TESTING:

It is also known as end to end testing. During this test, testing team validates whether our application build co-existence with other existing software's are not?

21. PARALLEL TESTING:

It is also known as comparative testing and applicable to software products only. During this test, testing team compare our application build with competitors products in the market.

22. PERFORMANCE TESTING:

It is an advanced testing technique and expensive to apply because testing team have to create huge environment to conduct this testing. During this test, testing team validates Speed of Processing. During this performance testing, testing team conduct load testing and stress testing.

23. LOAD TESTING:

The execution of our application under customer expected configuration and customer expected load to estimate performance is called Load Testing.

24. STRESS TESTING:

The execution of our application under customer expected configuration and un interval load's to estimate performance is called stress testing.

25. STORAGE TESTING:

The execution of application under huge amounts of resources to estimate storage limitations is called storage Testing.

26. DATA VOLUME TESTING:

The execution of our application under customer expected configuration to estimate peak limits of data is called data volume testing.

27. BIG BANG TESTING/INFORMAL TESTING/SINGLE STAGE TESTING:

A testing team conducts single stage testing, after completion of entire system development instead of multiple stages.

28. INCREMENTAL TESTING/FORMAL TESTING:

A multiple stages of testing process from unit level to system level is called incremental testing. It is also known as formal testing.

SOFTWARE TESTING LIFE CYCLE (STLC):

STLC contains 6 phases:

1. **Test Planning.**
2. **Test Development.**
3. **Test Execution.**
4. **Result Analysis.**
5. **Bug Tracking.**
6. **Report.**

1. Test planning:

Plan: Plan is strategic document which describes how to perform a task in an effective, efficient and optimized way.

Test plan: Test plan is strategic document, which contains some information that describes how to perform testing on an application in an effective, efficient and optimized way.

Optimization: It is process of utilizing the available resources to their level best and getting maximum possible out put.

Note: Test plan is prepared by the Test Lead.

CONTENTS OF TEST PLAN (or) INDEX OF TEST PLAN:

1.0 Introduction
- **1.1 Objective.**
- **1.2 Reference documents.**

2.0 Test coverage
- **2.1 Features to be tested**
- **2.2 Features not to be tested**

3.0 Test strategy
- **3.1 Levels of testing**
- **3.2 Types of testing**
- **3.3 Test design technology**
- **3.4 Configuration management**
- **3.5 Test matrices**
- **3.6 Terminology**
- **3.7 Automation plan**
- **3.8 List of automated tools**

4.0 Base criteria
- **4.1 Acceptance criteria**
- **4.2 Suspension criteria**

5.0 Test deliverables
6.0 Test environment
7.0 Resource planning
8.0 Scheduling
9.0 Staffing and Training
10.0 Risk and Contingences.
11.0 Assumptions.
12.0 Approval information.

1.0 INTRODUCTION:

1.1 Objective: The purpose of the document will be clearly described clearly in this section.

1.2 Reference documents: The list of all the documents that are referred while preparing the test plan will be listed out here in this section.

2.0 TEST COVERAGE:

2.1 Features to be tested: The list of all the features with in the scope are to be tested will be listed out here in this section.

2.2 Features not to be tested: The list of all the features that are not for planned for the testing will be listed out here in this section.

EX:

a. Out of scope features.
b. Low risk features.
c. Features that are planed to in corporate in future.
d. Features that are skipped based on time constrains.

3.0 TEST STRATEGY:

It is an organizational level term which describes how to perform testing on all the projects in that organization.

Test plan: It is project level term which describes how to perform testing in a particular project in a detailed manner.

3.1 Levels of testing: The list of all the levels of testing that are maintained in that company will be listed out in this section.

3.2 Types of testing: The list of all the types of testing that are conducted in that company will be listed out here in this section.

Ex: Build Acceptance Testing, Retesting etc.

3.3 Test design technique: The list of all the techniques that are used in that company, while developing the test cases will be listed out in this section

EX: BVA- Boundary Value Analysis, ECP- Equalance Class Partition.

3.4 Configuration management: To be discussed

Note: Metrics: Measurements.

3.5 Test metrics: The list of metrics needs to be maintained in that company during the testing process will be maintained in this section.

Ex: Test case metrics, Defect metrics etc.

3.6 Terminology: The list of all the terms along with their meaning's that are used in that company will be maintained here in this section.

Ex: GUI/UI/COSMETIC etc.

3.7 Automation plan: The list of all the areas that are planed for automation will be listed out here in this section.

3.8 Automated tools: The list of automated tools that are used in that company will be listed out in this section.

Ex: LOAD RUNNER, WIN RUNNER, QTP etc.

4.0 BASE CRIATERIA:

4.1 Acceptance criteria: When to stop testing will be clearly specified in this section.

4.2 Suspension criteria: When to suspend the build will be clearly maintain in this section.

5.0 TEST DELEVERABELS:

The lists of all the documents that are to be prepared during testing process will be maintained here in this section.

Ex: Test case document, Defect profile document etc.

6.0 TEST ENVIRONMENT:

The clear details of the environment that is about to be used for testing the application will be clearly maintained in this section.

Ex: Depending upon the Project the Environment will be selected.

1. STAND-ALONE ENVIRONMENT (OR) ONE-TIER ARCHITECTURE.
2. CLIENT-SERVER ENVIRONMENT (OR) TWO-TIER ARCHITECTURE.
3. WEB ENVIRONMENT (OR) THREE-TIER ARCHITECTURE.
4. DISTRIBUTED ENVIRONMENT (OR) N-TIER ARCHITECTURE.

7.0 RESOURCE PLANING:

Who has to do what will be clearly planned and maintained in this section.

8.0 SCHEDULING:

The starting dates and ending dates of each and every list will be clearly planed and maintained here in this section.

9.0 STAFFING AND TRAINING (KTS-KNOWLEDGE TRANSFER SESSION'S):

To accomplish this project successfully any staff or training is required then that information will be clearly maintained in this section.

10.0 RISK & CONTINGENCIES:

The list of all the potential risks and corresponding plans will be maintained in this section.

RISKS:

a. Employees may leave the organization in the middle of the project.
b. Unable to deliver the project with in the dead lines.
c. Customers imposed dead lines.
d. Unable to test all the features with in the time.

CONTINGENCIES (SOLUTION'S):

a. Employees need to maintain on the bench.
b. Proper plan ensurance.
c. What to be skipped should be planed in case of customers imposed dead lines.
d. Priority based execution.

11.0 ASSUMPTION:

The list of all the assumption's need to be made by the testing people will be maintained here in this section.

Ex: Test Data.

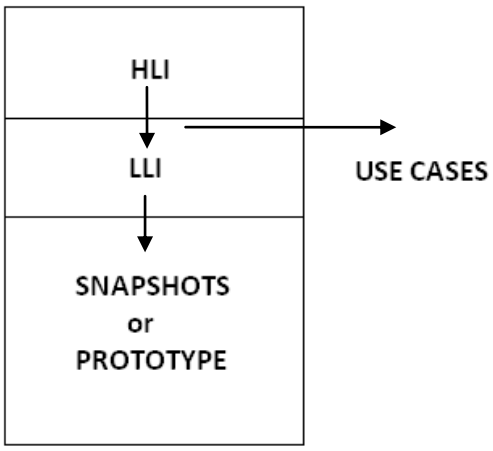
12.0 APPROVAL INFORMATION:

Who has approved this document, when it is approved will be maintained in this section.

Ex: Test Manager.

2. Test Development:

Requirement Document:

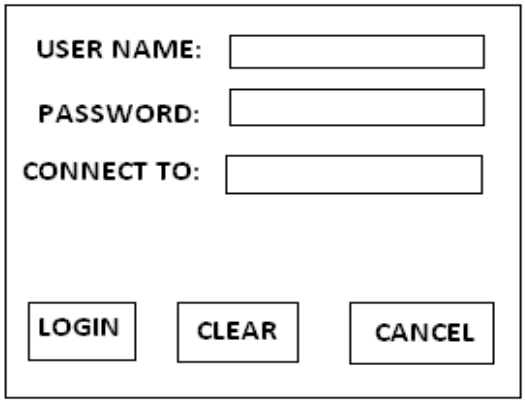


HLI-HIGH LEVEL INFORMATION, LLI-LOW LEVEL INFORMATION.

USE CASE: It describes the functionality of certain feature of an application in terms of actors, actions and responses.

Note: USECASE was prepared by Business Analyst (BA).

SNAPSHOT:



Functional Requirements:

1. Login screen should contain USER NAME, PASSWORD, CONNECT TO Fields, LOGIN, CLEAR, CANCEL Buttons.
2. Connect to field is not a mandatory field, but it should allow the user to select a database option, if required.
3. Upon entering the user name, password and clicking on login button, the corresponding page must be displayed.
4. Upon entering the information into any of the fields and clicking on clear button, all the fields must be cleared and the cursor must be available in the user name field.
5. Upon clicking on cancel button, login screen must be closed.

Special Requirements:

1. Upon invoking the application, login and clear buttons must be disable.
2. Cancel button must be always enabled.
3. Upon entering some information into any of the fields, the clear button must be enabled.
4. Upon entering some information into username and password login button must be enabled.
5. Tabbing order must be username, password, connect to, login, clear, and cancel.

USE CASE TEMPLATE:

NAME OF THE USE CASE :
DESCRIPTION OF THE USE CASE :
ACTORS INVOLVED :
SPECIAL REQUIREMENTS :
PRE CONDITIONS :
POST CONDITIONS :
FLOW OF EVENTS :

USE CASE DOCUMENT:

NAME OF THE USE CASE : LOGIN USE CASE
DSCRIPTION OF THE USE CASE : THIS USE CASE DESCRIBES THE FUNCTIONALITY OF ALL THE FEATURES PRESENT IN THE LOGIN SCREEN.
ACTORS INVOLVED : NORMAL USER/ADMIN USER.
SPECIAL REQUIREMENTS :

There are 2 types of Special Requirements:

1. **Implicit Requirements**
2. **Explicit Requirements**

1. **Implicit Requirements:** The requirements that are analyzed by the Bossiness Analyst and his team, which will add some value to the application, and will not affect any of the customer's requirement.
2. **Explicit Requirements:** The requirements that are explicitly given by the customer are known as Explicit Requirements.

Explicit Requirements:

1. Upon invoking the application, login and clear buttons must be disable.
2. Cancel button must be always enabled.
3. Upon entering some information into any of the fields, the clear button must be enabled.
4. Upon entering some information into both the username and password fields' login button must be enabled.
5. Tabbing order must be username, password, connect to, login, clear, and cancel.

Implicit Requirements:

1. Upon invoking the application the cursor must be available in the user name field.
2. Upon entering invalid user name, valid password, and clicking on login button the following error msg must be displayed "INVALID USER NAME Please try again".
3. Upon entering valid user name, invalid password and clicking on login button the following error msg must be displayed "INVALID PASSWORD Please try again".
4. Upon entering invalid user name, invalid password and clicking on login button the following error msg must be displayed "INVALID USER NAME & INVALID PASSWORD Please try again".

GENERIC REQUIREMENTS: Universal Requirements.
SPECIFIC REQUIREMENTS: Customer Requirements.

Pre Conditions : Before Starting any Task.
Post Conditions : After Completion of any Task.

Pre Conditions : Login screen must be available.
Post Conditions : Either Home page or Admin page for valid users and error msg's for invalid users.

Flow of Events:

Main Flow:

ACTION	REPONSE
*Actor invoke the application	* Application displays the login screen with the following fields: User name, password, connect to, login, clear, and cancel buttons.
*Actor enters valid user name, valid password and clicks on login button.	*Authentication's, application displays either home page or admin page depending upon the actor enter.
*Actor enters valid user name, valid password, selects a database option and clicks on login button.	*Authenticates, application displays either home page or admin page depending upon the actor enter with the mentioned database connection.
*Actor enters invalid user name, valid password and clicks on login button.	*Go to alternative flow table 1.
*Actor enters valid user name, invalid password and clicks on login button.	*Go to alternative flow table 2.
*Actor enters invalid user name, invalid password and click on login button.	*Go to alternative flow table 3.
*Actor enters some information into any of the fields and click on clear button.	*Go to alternative flow table 4.
*Actor clicks on cancel button.	*Go to alternative flow table 5.

Alternative flow table 1 (INVALID USER NAME):

ACTION	RESPONSE
*Actor enters invalid user name, valid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID USER NAME Please try again".

Alternative flow table 2 (INVALID PASSWORD):

ACTION	RESPONSE
*Actor enters valid user name, invalid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID PASSWORD Please try again".

Alternative flow table 3 (INVALID USER NAME & PASSWORD):

ACTION	RESPONSE
*Actor enters invalid user name, invalid password and clicks on login button.	*Authenticates, application displays the following error msg: "INVALID USER NAME & PASSWORD Please try again".

Alternative flow table 4 (CLEAR CLICK):

ACTION	RESPONSE
*Actor enters some information in any of the fields and clicks on clear button.	*Application clears the fields and makes the cursor available in the user name.

Alternative flow table 5 (CANCEL CLICK):

ACTION	RESPONSE
*Actor clicks on cancel button.	*Login screen is closed".

THE GUIDE LINES TO BE FOLLOWED BY A TEST ENGINEER, SOON AFTER THE USE CASE DOCUMENT IS RECEIVED:

1. Identify the module to which the use case belongs to.
A: Security module.

2. Identify the functionality of the use case with the request of total functionality.
A: Authentication.

3. Identify the actors involved in the use case.
A: Normal user/Admin user.

4. Identify the inputs required for testing.
A: Valid and invalid user names and passwords.

5. Identify whether the use case is linked with other use case or not.
A: It is linked with Home page and Admin page use cases.

6. Identify the pre conditions.
A: LOGIN Screen must be available.

7. Identify the post conditions.
A: Either Home page/Admin page for valid users, and error msgs for invalid users.

8. Identify the functional points and prepare the functional point document.

UNDERSTAND:

9. Understand the main flow of the application.
10. Understand the alternative flow of the application.
11. Understand the special requirements.

DOCUMENT:

12. Document the test cases for main flow.
13. Document the test cases for alternative flow.
14. Document the test cases for the special requirements.
15. Prepare the cross reference metrics or traceability metrics.

Functional Point:

The point at which the user can perform some actions in the application can be considered as Functional Point.

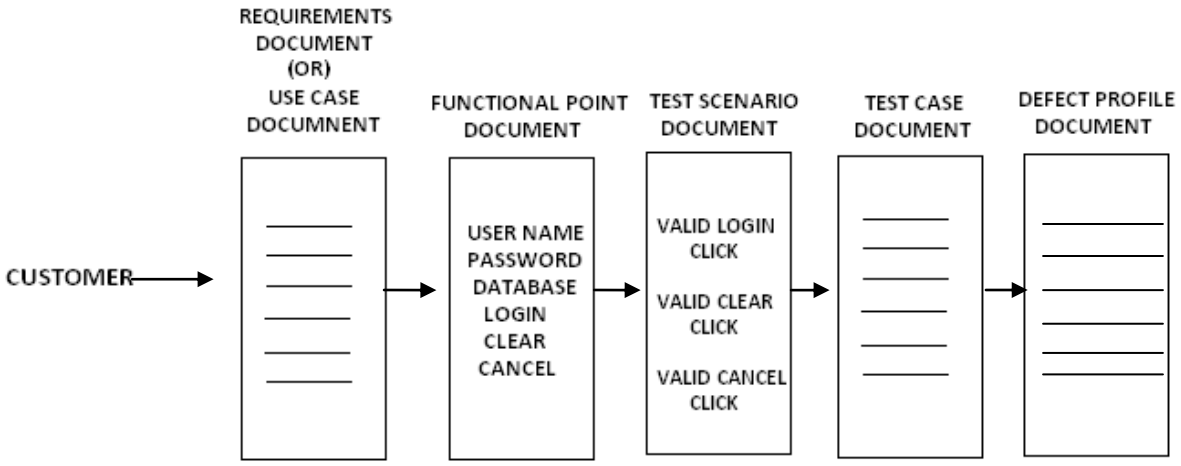
Test Scenario:

The situation where we can do testing.

There are 3 types of flow:

1. Main flow : Main page/Home Page.
2. Alternative flow : Error msgs page.
3. Exceptional flow : Server problems/Network problems.

Testing process related Documents:



TRACEABILITY MATRIX:

It is a document which contains a table of linking information used for tracing back for the reference. In any kind of confusion or questionable situation.

Note: Matrix: Combining different points in a document.

TM:

UCD ID	FPD ID	TSD ID	TCD ID	DPD ID
8.1	3	4	26	1
23.2	21	8	86	2
5.4	34	6	44	3

REQUIREMENT TRACEABILITY MATRIX:

RTM:

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

Note: The Test cases which are prepared related to which requirement is mentioned here in this table. If we mention Req. Id in the test case document, it same as the RTM, so we need not to maintain RTM separately for the application.

DEFECT TRACEABILITY MATRIX:

DTM:

DEFECT ID	TEST CASE ID
1	23
2	34
3	56
4	44

Note: The defects which are related to which test case is mentioned here in this table.

TYPES OF TEST CASES:

TEST CASES are broadly divided into 3 types:

1. GUI TEST CASES
2. FUNCTIONAL TEST CASES
3. NON FUNCTIONAL TEST CASES

NON FUNCTIONALITY TEST CASES:

Ex:
a. Compatibility Testing.
b. Performance Testing.
c. Usability Testing.
d. Installation Testing.

FUNCTIONAL TEST CASES are further divided into 2 ways:

1. POSITIVE TEST CASES
2. NEGATIVE TEST CASES

FUNCTIONALITY TESTING: It is a type of testing in which one will perform testing the functionality of an application, functionality means behavior of the application.

Guidelines for writing the GUI Test Cases:

1. Check for the availability of all the objects.
2. Check for the consistency of the objects.
3. Check for the alignment of the objects, in case of customer requirement only.
4. Check for the spellings and grammar.

Apart from the above guidelines any idea we get with which we can test something in the application, just by look and feel without doing any actions, and then all those ideas also can be considered as GUI Test Cases.

Guidelines for writing the Positive Test Cases:

1. A Test engineer should have positive mind set.
2. He should consider the positive flow of the application.
3. He should use only valid inputs from the point of the functionality.

Guidelines for writing the Negative Test Cases:

1. A Test engineer should have negative mind set.
2. He should consider the negative flow of the application.
3. He should use at least one invalid input for each set of data.

Guidelines for writing Test Cases:

1. Feel like Boss.
2. Action should be possible and expected value should be related to the actions based on the requirements.
3. Test Case should be clear and understandable.
4. It should be Simple, Easy and Powerful.

TEST CASE TEMPLATE:

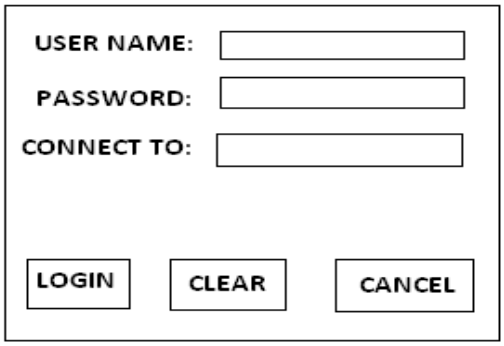
PROJECT NAME: MODULE: AUTHOR:										
REQ. ID	**TEST CASE ID**	**CATEGORY**	**PREREQUISITE**	**DESCRIPTION/ TEST STEPS**	**EXPECTED VALUE**	**ACTUAL VALUE**	**TEST DATA**	**RESULT (PASS/FAIL) BLOCKED**	**BUILD No.**	**PRIORITY**

TEST CASE DOCUMENT:

PROJECT NAME: MODULE: AUTHOR:										
REQ. ID	**TEST CASE ID**	**CATEGORY**	**PREREQUISITE**	**DESCRIPTION/ TEST STEPS**	**EXPECTED VALUE**	**ACTUAL VALUE**	**TEST DATA**	**RESULT (PASS/FAIL) BLOCKED**	**BUILD No.**	**PRIORITY**
	1	POSITIVE	NA	INVOKE THE APPLICATION	LOGIN SCREEN MUST BE DISPLAYED	LOGIN SCREEN DISPLAYED		PASS	1	
	2	GUI	NA	CHECK FOR THE AVAILABILITY OF ALL THE OBJECTS IN THE LOGIN SCREEN AS PER THE LOT	ALL THE OBJECTS MUST BE DISPLAYED AS PER THE **LOT**	ALL THE OBJECTS ARE AVAILABLE AS PER THE LOT	**LOT**	PASS	1	
	3	GUI	NA	CHECK FOR THE CONSISTANCY OF ALL THE OBJECTS IN THE LOGIN SCREEN	ALL THE OBJECTS MUST BE CONSISTANT WITH EACH OTHER	ALL THE OBJECTS ARE CONSISTANT WITH EACH OTHER		PASS	1	
	4	GUI	NA	CHECK FOR THE SPELLINGS IN THE LOGIN SCREEN	ALL THE SPELLINGS MUST BE CORRECT	ALL THE SPELLINGS ARE CORRECT		PASS	1	
	5	GUI	NA	CHECK FOR THE INTIAL POSITION OF THE CURSOR	THE CURSOR MUST BE AVAILABLE AT THE USERNAME FIELD	THE CURSOR IS NOT POSITIONED IN THE USERNAME FIELD		FAIL	1	
	6	GUI	NA	CHECK FOR THE ENABLED PROPERTY OF THE LOGIN, CLEAR & CANCEL BUTTONS	INITIALLY LOGIN & CLEAR BUTTONS MUST BE DISABLED AND CANCEL MUST BE ALWAYS ENABLED	LOGIN BUTTON IS DISABLED AND CLEAR & CANCEL BUTTONS ARE ENABLED		FAIL	1	
	7	POSITIVE	NA	ENTER SOME INFORMATION INTO ANY OF THE FIELDS & CHECK FOR THE ENABLED PROPERTY OF CLEAR BUTTON	CLEAR BUTTON MUST BE ENABLED	CLEAR BUTTON IS ENABLED		PASS	1	
	8	POSITIVE	NA	ENTER SOME INFORMATION INTO USERNAME & PASSWORD AND CHECK FOR THE ENABLED PROPERTY OF LOGIN BUTTON	LOGIN BUTTON MUST BE ENABLED	LOGIN BUTTON IS ENABLED		PASS	1	
	9	POSITIVE	NA	ENTER SOME INFORMATION INTO ANY OF THE FIELDS AND CLICK ON CLEAR BUTTON	ALL THE FIELDS MUST BE CLEARED AND THE CURSOR SHOULD BE DISPLAYED IN THE USERNAME FIELD	ALL THE FIELDS ARE CLEARED BUT CURSOR IS NOT PLACED IN THE USERNAME FIELD		FAIL	1	
	10	POSITIVE	NA	ENTER THE USERNAME, PASSWORD AS PER THE VIT AND CLICK ON LOGIN BUTTON	CORRESPONDING PAGE MUST BE DISPLAYED AS PER THE **VIT**	CORRESPONDING PAGES ARE NOT DISPLAYED AS PER THE VIT	**VIT**	FAIL	1	
	11	POSITIVE	NA	ENTER USERNAME, PASSWORD AS PER THE VIT & SELECT A DATABASE OPTION	CORRESPONDING PAGE MUST BE DISPLAYED AS PER THE **VIT** WITH THE MENTIONED DATABASE CONNECTION	CORRESPONDING PAGES ARE NOT DISPLAYED AS PER THE VIT, BUT THE MENTIONED DATABASE CONNECTION IS PROPERLY ESTABLISHED	**VIT**	FAIL	1	
	12	POSITIVE	NA	CLICK ON CANCEL BUTTON	LOGIN SCREEN MUST BE CLOSED	LOGIN SCREEN IS CLOSED		PASS	1	
	13	POSITIVE	LOGIN SCREEN MUST BE INVOKED	CHECCK FOR THE TABBING ORDER	TABBING ORDER MUST BE AS FOLLOWS: USERNAME, PASSWORD, CONNECT TO, LOGIN, CLEAR & CANCEL	TABBING ORDER IS AS FOLLOWS: USERNAME, PASSWORD, CONNECT TO, LOGIN, CLEAR & CANCEL		PASS	1	
	14	NEGATIVE	NA	ENTER THE USERNAME & PASSWORD AS PER IVIT AND CLICK ON LOGIN BUTTON	CORRESPONDING ERROR MSG SHOULD DISPLAYED AS PER **IVIT**	CORRESPONDING MSG`S ARE NOT DISPLAYED AS PER IVIT	**IVIT**	FAIL	1	
	15	NEGATIVE	NA	ENTER SOME INFORMATION ONLY INTO THE USERNAME FIELD AND CHECK FOR THE ENABLED PROPERTY OF LOGIN BUTTON	LOGIN BUTTON MUST BE DISABLED	LOGIN BUTTON IS ENABLED		FAIL	1	
	16	NEGATIVE	NA	ENTER SOME INFORMATION ONLY INTO THE PASSWORD FIELD AND CHECK FOR THE ENABLED PROPETY OF LOGIN BUTTON	LOGIN BUTTON MUST BE DISABLED	LOGIN BUTTON IS DISABLED		PASS	1	

Note: The underlined words in the Test Data column are the Hyperlinks to go for the Respective Data Table.

LOGIN SCREEN:



LOGIN OBJECTS TABLE (LOT):

S.No.	OBJECT NAME	OBJECT TYPE
1	USERNAME	TEXT BOX
2	PASSWORD	TEXT BOX
3	CONNECT TO	COMBO BOX
4	LOGIN	BUTTON
5	CLEAR	BUTTON
6	CANCEL	BUTTON

VALID INPUTS TABLE (VIT):

S.No.	USERNAME	PASSWORD	EXPECTED PAGE	ACTUAL PAGE	RESULT
1	SURESH	QTP	ADMIN	HOME	FAIL
2	ADMIN	ADMIN	ADMIN	ADMIN	PASS
3	CHIRU	SRIDEVI	HOME	CHIRU HOME PAGE	PASS
4	NAG	AMALA	HOME	NAG HOME PAGE	PASS
5	NTR	BALAKRISHNA	HOME	NTR HOME PAGE	PASS
6	VENKY	ILLU	HOME	VENKY HOME PAGE	PASS

INVALID INPUTS TABLE (IVIT):

S.No.	USERNAME	PASSWORD	EXPECTED MESSAGE	ACTUAL VALUE	RESULT
1	SURI	QTP	INVALID USER NAME PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	PASS
2	CHIRUTHA	SRIDEVI	INVALID USER NAME PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	PASS
3	VENKI	SAVITHRI	INVALID PASS WORD PLEASE TRY AGAIN	INVALID PASS WORD PLEASE TRY AGAIN	PASS
4	NTR	BALU	INVALID PASS WORD PLEASE TRY AGAIN	INVALID PASS WORD PLEASE TRY AGAIN	PASS
5	SRI	JAVA	INVALID USERNAME & PASS WORD PLEASE TRY AGAIN	SRI HOME PAGE	FAIL
6	RAJA	RANI	INVALID USERNAME & PASS WORD PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN	FAIL

TEST DESIGN TECHNIQUES (or) INPUT DESIGN TECHNIQUES :

- **Boundary Value Analysis BVA(Range / Size):**

Min	PASS
Min-1	FAIL
Min+1	PASS
Max	PASS
Max-1	PASS
Max+1	FAIL

- **Equivalence Class Partitions ECP (Type):**

VALID	INVALID
Pass	Fail

Ex:

A login process allows user ID and Password to validate users. User ID allows Alpha Numerics in lower case from 4 to 16 characters long. Password allows alphabets in lower case 4 to 8 characters long. Prepare BVA and ECP for user ID and password.

USER ID

BVA

4 --- PASS
3 --- FAIL
5 --- PASS
16 -- PASS
15 -- PASS
17 -- FAIL

ECP

VALID	INVALID
a to z 0 to 9	A to Z Special characters Blank space

PASSWORD

BVA

4 -- PASS
3 -- FAIL
5 -- PASS
8 -- PASS
7 -- PASS
9 -- FAIL

ECP

VALID	INVALID
a to z	A to Z 0 to 9 Special characters Blank space

TEST DESIGN TECHNIQUES: They are used for making the Test Engineers to write the Test Cases very easily and comfortably, even in the complex situations. Mainly 2 famous techniques used by most of the companies are:

1. BOUNDARY VALUE ANALYSIS (BVA)
2. EQUALANCE CLASS PARTITION (ECP)

1. BOUNDARY VALUE ANALYSIS (BVA): Whenever there is a range kind of input to be tested, it is suggested to test the boundaries of the range instead of whole range, usually one will concentrate on the following values:

LB-1	FAIL
LB	PASS
LB+1	PASS
MV	PASS
UB-1	PASS
UB	PASS
UB+1	FAIL

LB-LOWER BOUNDARY
MV-MEDIUM VALUE
UB-UPPER BOUNDARY

2. EQUALANCE CLASS PARTITION (ECP): Whenever there are more no of requirements for particular feature, for huge range of data need to be tested than it is suggested to, first divide the inputs into equal classes and then write the Test Cases.

Ex: Write the Test Cases for testing a Text Box, whose requirements are as follows:
1. It should accept Min 4 Characters and Max 20 Characters.
2. It should accept only @ and _ Symbols only.
3. It should accept only Small Alphabets.

BVA:

LB-1	3
LB	4
LB+1	5
MV	12
UB-1	19
UB	20
UB+1	21

ECP:

VALID	INVALID
4	3
5	21
12	A-Z
19	Except @ and _ all the remaining Special characters
20	0-9
a-z	Spaces
@,_	Decimal Points

VIT:

S.No.	INPUTS
1	abcd
2	ab@cd
3	abcdabcda__z
4	abcdabcdabcdabcd
5	abcdabcdabzdabcda_@z

IVIT:

S.No.	INPUTS
1	abc
2	ABCD
3	ABCD@__@abcd
4	<>@+-*/,.\abcdzyxw
5	12345
6	5.4
7	abcd ABCD z@/*
8	abcdabcdABCDABCD@<>+-ab
9	ABCD123, abcd123
10	abcdabcdABCD12345@<>ab @abcd

TEST CASE DOCUMENT:

TEST CASE ID	TEST CASE TYPE	DESCRIPTION	EXPECTED VALUE	TEST DATA
1	Positive	Enter the values into the Text Box as per the **VIT**	It should accept	**VIT**
2	Negative	Enter the values into the Text Box as per the **IVIT**	It should not accept	**IVIT**

3. Test Execution:

In this phase the test engineer will do the following:

1. They will perform the action as it is described in the Description column.
2. They will observe the actual behavior of the Application.
3. They will write the observed value in the Actual value column.

4. Result Analysis:

In this phase the test engineer will compare the actual value with the expected value. If both are matching, then he will decide the result as PASS otherwise FAIL.

Note: If at all the Test case is not executed in any reason, then the test engineer will specify BLOCKED in the result column.

Ex: If application has 5 pages, in each page a next button will be available, when we click on next button, we will enter into next page, if next page button of 4th page is not working, we can't enter into 5th page and all the test cases related to that 5th page can't test. The result to all that test cases will be BLOCKED.

5. Bug Tracking:

It is a process of identifying, isolating and managing the defects.

DEFECT ID: The sequence of defect no's will be mentioned here in this section.

TEST CASE ID: The test case id based on which defect is found will be mentioned here in this section.

ISSUE DISCRIPTION: What exactly the defect is will be clearly described here in this section.

REPRODUCABLE DEFECTS: The list of all the steps followed by the test engineer, to identify the defects will be listed out here in this section.

DETECTED BY: The name of the test engineer, who has identified will be mentioned here in this section.

DETECTED DATE: The date on which the defect is identified will be mentioned here in this section.

DETECTED BUILD: The build number in which the defect is identified will be mentioned here in this section.

DETECTED VERSION: The version number in which defect is identified will be mentioned here in this section.

VERSION: On which version the build was released will be mentioned here in this section.

Note: Depending upon the size of an application the version will be changed, but the build will be the same keep on changing. VERSION will be decided by the Software Configuration Management (SCM).

Ex: If the application version is 2.3.6, if the build 1 is released, the testing team will test the build, if they identified the defects and sent back for next build, if the new requirements from the customer are added to that application, then the version of an application changes from 2.3.6 to 2.3.7. Then the build 2 is released, the build number will be the same keeps on increasing.

DEFECT SEVERITY: Severity describes the seriousness of the application.

Severity is classified into 4 types:

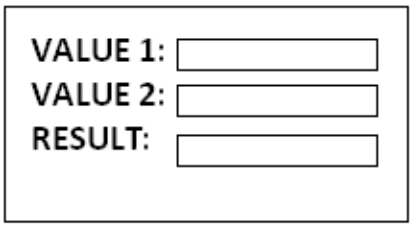
1.	**FATAL**	**SIVI 1**	**S1**	**1**
2.	**MAJOR**	**SIVI 2**	**S2**	**2**
3.	**MINOR**	**SIVI3**	**S3**	**3**
4.	**SUGGESTION**	**SIVI4**	**S4**	**4**

1. FATAL DEFECTS: If at all the problems are related to the navigational or unavailability if main functionality then such type of defects are treated as FATAL DEFECTS.

Ex:

PAGE1 → PAGE2 → PAGE3 → X.

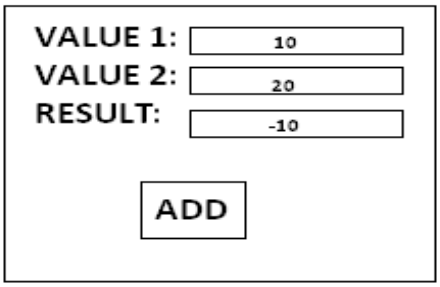
No navigation for next page.



ADD button is missing.

2. MAJOR DEFECTS: If at all the problems are related to working of the main functionality then such types of defects are treated as MAJOR DEFECTS.

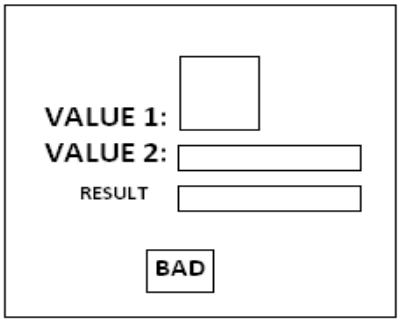
Ex:



Instead of 30 the result is -10.

3. MINOR DEFECTS: If at all the problems are related to look and feel of the application, then such type of defects are treated as MINOR DEFECTS.

Ex:



Instead of ADD button BAD button, and Text boxes size and Value names are not consistent with each other.

4. SUGGESTIONS: If at all the problems are related to value (User friendliness) of the application then such type of problems are treated as SUGGESTIONS.

Ex:

INVALID USERNAME PLEASE TRY AGAIN----POOR HELP
PLEASE ENTER ALPHANUMERIC ONLY----STRONG HELP

TOOL TIPS MUST BE HELPFUL.

DEFECT PRIORITY: It describes the sequence in which, the defects need to be rectify.

PRIORITY is classified into 4 types:

1.	**CRITICAL**	**PRI 1**	**P1**	**1**
2.	**HIGH**	**PRI 2**	**P2**	**2**
3.	**MEDIUM**	**PRI 3**	**P3**	**3**
4.	**LOW**	**PRI 4**	**P4**	**4**

Usually:

SEVERITY	PRIORITY
FATAL DEFECTS	CRITICAL
MAJOR DEFECTS	HIGH
MINOR DEFECTS	MEDIUM
SUGGESTIONS	LOW

Sometimes highest Severity defects will be given least Priority and sometimes least Severity defects will be given highest Priority.

CASE 1: LEAST SEVERITY-HIGHEST PRIORITY:

Whenever there is a customer visit, all the look and feel defects will be given highest priority.

Ex:

We are developing HDFC bank application, every page should have HDFC logo on top of it, but it shows HSBC, it is look and feel problem, even though we are not completed the main functionality of an application, customer will visit the company then we give all look and feel defects as high priority, even though the main functionality of an application is not completed.

CASE 2: HIGHEST SEVERITY-LEAST PRIORITY:

Whenever some part of the module or application is not released to the testing department as it is under construction the testers will usually raise it as a fatal defect but the development lead treats it as a least priority.

Ex:

Suppose 80% of an application developed and remaining 20% was not developed. The application is released to the testing department. The test engineer will raise this as a fatal defect in the 20% application, but developer lead will treated it as a least priority.

Note:

Sometimes highest Severity defects will be given least priority and sometimes least priority defects will be given highest priority, for this sake we use both SEVERITY and PRIORITY.

TYPES OF DEFECTS:

1. **User Interface Bugs: SUGGESTIONS**

Ex 1: Spelling Mistake → High Priority
Ex 2: Improper alignment → Low Priority

2. **Boundary Related Bugs: MINOR**

Ex 1: Does not allow valid type → High Priority
Ex 2: Allows invalid type also → Low Priority

3. **Error Handling Bugs: MINOR**

Ex 1: Does not providing error massage window → High Priority
Ex 2: Improper meaning of error massages → Low Priority

4. **Calculation Bugs: MAJOR**

Ex 1: Final output is wrong → Low Priority
Ex 2: Dependent results are wrong → High Priority

5. Race Condition Bugs: MAJOR
Ex 1: Dead Lock → High Priority
Ex 2: Improper order of services → Low Priority

6. Load Condition Bugs: MAJOR
Ex 1: Does not allow multiple users to operate → High Priority
Ex 2: Does not allow customer expected load → Low Priority

7. Hardware Bugs: MAJOR
Ex 1: Does not handle device → High Priority
Ex 2: Wrong output from device → Low Priority

8. ID Control Bugs: MINOR
Ex: Logo missing, wrong logo, version no mistake, copyright window missing, developers name missing, tester names missing.

9. Version Control Bugs: MINOR
Ex: Difference between two consecutive build versions.

10. Source Bugs: MINOR
Ex: Mistakes in help documents.

DEFECT RESOLUTION/STATUS: To set the status of the defect.
Ex: NEW, OPEN, DEFERRED…….etc.

FIX-BY/DATE/BUILD: The developer who has fixed the defect, on which date and build no will be mentioned here in this section.

DATE CLOSURE: The date on which the defect is rectified will be mentioned here in this section.

DEFECT AGE: The time gap between “Reported on” and “Resolved On”.

Note:

- Define NA in the Reproducible steps column if it is look and feel.
- Heart of the Defect profile is Issue description.
- Support column for Description is Reproducible steps.

DEFECT PROFILE TEMPLATE:

DEFECT ID	TEST CASE ID	ISSUE DISCRIPTION	REPRODUCABLE STEPS	DETECTION				DEFECT		DEFECT RESOLUTION (OR) STATUS	FIX			DATE CLOSURE
				BY	DATE	BUILD	VERSION	SEVERITY	PRIORITY		BY	DATE	BUILD	

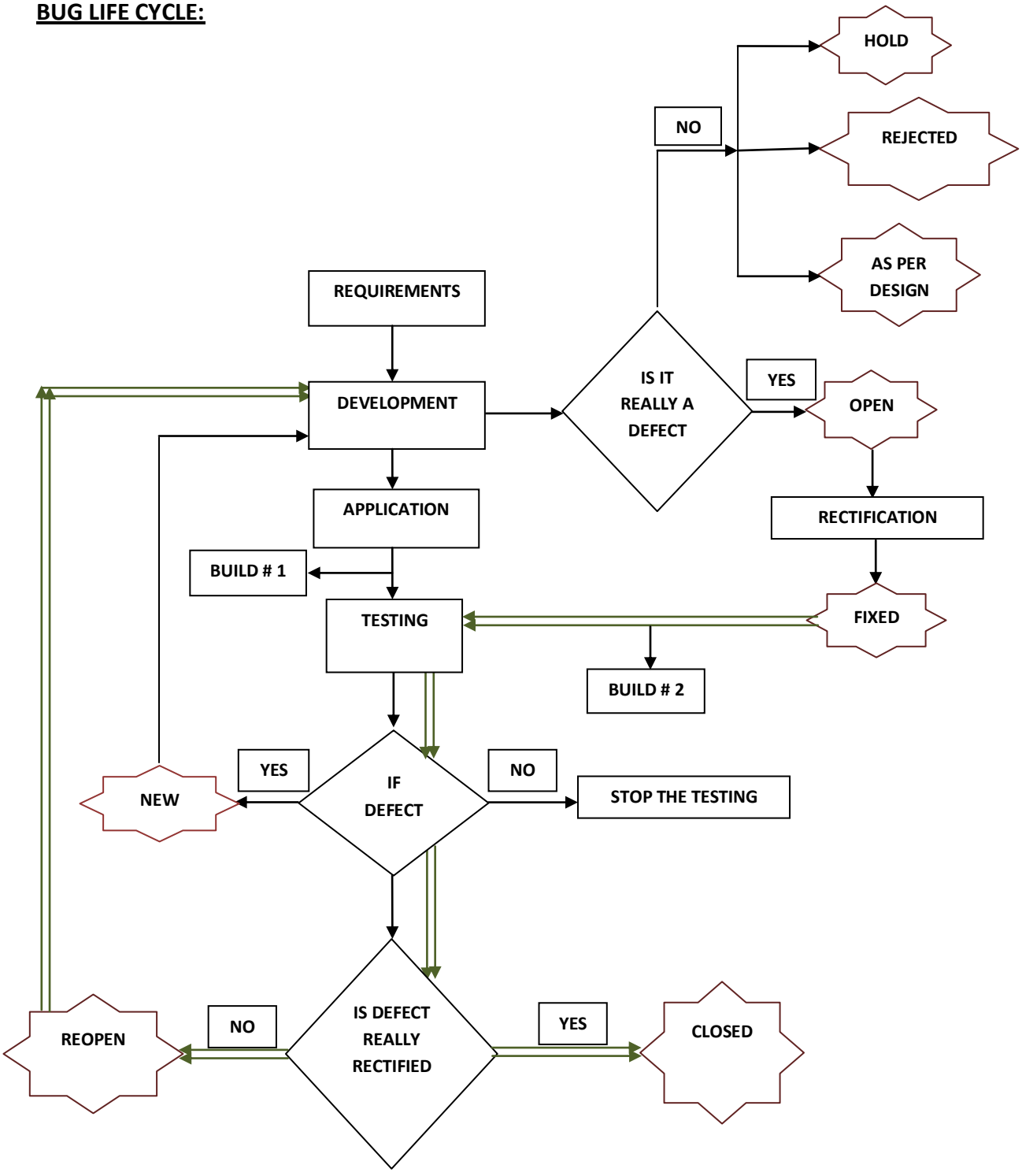
DEFECT PROFILE LOG:

DEFECT ID	TEST CASE ID	ISSUE DISCRIPTION	REPRODUCABLE STEPS	DETECTION				DEFECT		DEFECT RESOLUTION (OR) STATUS	FIX			DATE CLOSURE
				BY	DATE	BUILD	VERSION	SEVERITY	PRIORITY		BY	DATE	BUILD	
1	5	UPON INVOKING THE APPLIICATION, INITIALLY THE CURSOR IS NOT POSITIONED IN THE USER NAME FIELD	NA	SURI	16.12.09	1	2.3.6							
2	6	INITIALLY THE CLEAR BUTTON IS ENABLED, INSTEAD OF BEING DISABLED	NA	SURI	16.12.09	1	2.3.6							
3	9	UPON CLICKING ON CLEAR ALL THE FIELDS ARE CLEARED, BUT THE CURSOR IS NOT DISPLAYED IN THE USER NAME FIELD	1. ENTER SOME INFORMATION INTO ANY OF THE FIELDS. 2. CLICK ON CLEAR BUTTON. 3. OBSERVE THAT ALL THE FIELDS ARE CLEARED, BUT CURSOR IS NOT PLACED IN THE USER NAME FIELD.	SURI	16.12.09	1	2.3.6							
4	10	UPON ENTERING SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD THE PERSONAL HOME PAGE IS DISPLAYED, INSTEAD OF ADMIN PAGE	1. ENTER SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD. 2. CLICK ON LOGIN BUTTON. 3. OBSERVE THAT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE.	SURI	16.12.09	1	2.3.6							
5	11	UPON ENTERING SURESH INTO USER NAME FIELD, QTP INTO PASSWORD FIELD. SELECT A DATABASE OPTION AND CLICK ON LOGIN BUTTON. DATABASE CONNECTION IS PROPERLY ESTABLISHED BUT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE	1. ENTER SURESH INTO USER NAME FIELD & QTP INTO PASSWORD FIELD. 2. SELECT A DATABASE OPTION. 3. CLICK ON LOGIN BUTTON. 4. OBSERVE THAT DATABASE CONNECTION IS PROPERLY ESTABLISHED, BUT PERSONAL HOME PAGE IS DISPLAYED INSTEAD OF ADMIN PAGE.	SURI	16.12.09	1	2.3.6							
6	12	UPON ENTERING INVALID DATA INTO USER NAME & PASSWORD FIELDS. EXPECTED ERROR MESSAGES ARE NOT DISPLAYED. FOR DETAILS REFER TABLE 1.	1. ENTER USER NAME & PASSWORD AS PER THE TABLE 1. 2. CLICK ON LOGIN BUTTON. 3. OBSERVE THE ACTUAL VALUE AS PER THE TABLE 1.	SURI	16.12.09	1	2.3.6							
7	15	UPON ENTERING SOME INFORMATION ONLY INTO USER NAME FIELD, LOGIN BUTTON IS ENABLED INSTEAD OF BEING DISABLED	1. ENTER SOME INFORMATION ONLY INTO USER NAME FIELD. 2. OBSERVE THAT LOGIN BUTTON IS ENABLED INSTEAD OF BEING DISABLED.	SURI	16.12.09	1	2.3.6							

TABLE 1:

S.No.	USER NAME	PASSWORD	EXPECTED MESSAGE	ACTUAL VALUE
1	SRI	JAVA	INVALID USER NAME & PASSWORD PLEASE TRY AGAIN	PERSONAL HOME PAGE OF SRI
2	RAJA	RANI	INVALID USER NAME & PASSWORD PLEASE TRY AGAIN	INVALID USER NAME PLEASE TRY AGAIN

BUG LIFE CYCLE:



NEW: whenever the defect is newly identified by the tester, he will set the status as new.

OPEN: Whenever the developer accepts the defects then he will set the status as open.

DEFERRED: whenever the developer accept the defects and wants to rectify it in later then he will set the status as deferred.

FIXED: Once the defect is rectified then the developer will set the status as fixed, it is also called as rectified.

REOPEN & CLOSED: Once the next build is released the tester will check whether the defect is really rectified or not, if at all the defect is really rectified, then he will set the status as closed, otherwise reopen.

HOLD: Whenever the developer is confused to accept or reject, then he will set the status as hold.

Whenever the defect is in hold status there will be a meeting on that defect and if it is decided as a defect, then the developers will set the status as open otherwise the testers will close it.

REJECTED: Whenever the developers feel that, it is not at all a defect, then they will set the status as rejected.

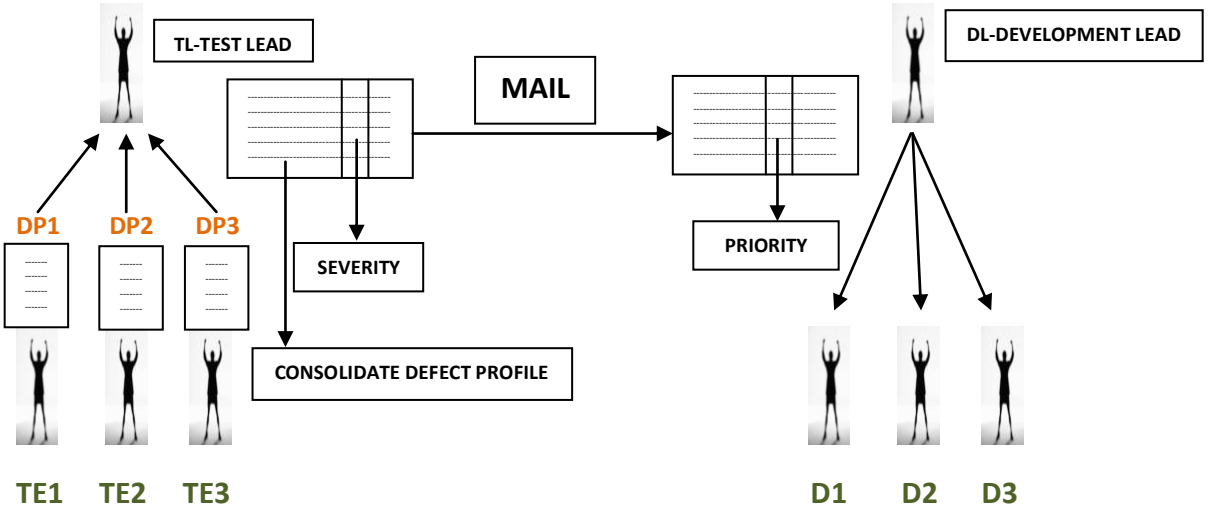
Whenever the defect is rejected, then the test engineers will once again check it, if at all they also feel it is not a defect then they will set the status as closed otherwise reopen.

AS PER DESIGN: This situation really occurs. Whenever developers feel the testers are not aware of latest requirements then they will set the status as as per design.

Whenever the defect is as per design status the testers will once again check it by going through the latest requirements, if at all they also feel it is as per design then they will set the status as closed otherwise reopen.

6. Reporting:

CLASSICAL BUG REPORTING PROCESS:

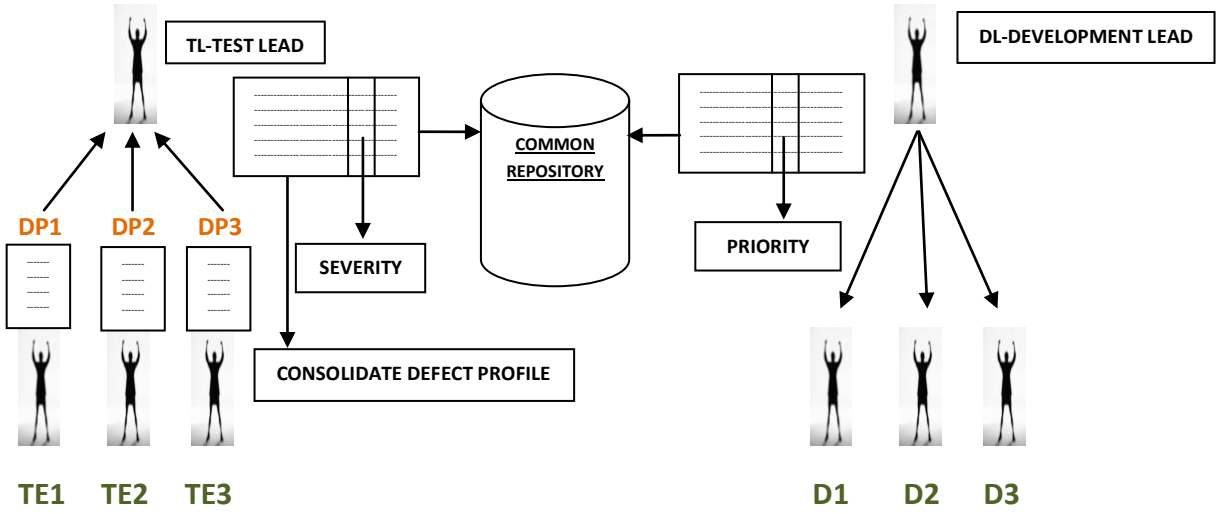


TE-TEST ENGINEER.
D-DEVELOPER.

DRAWBACKS:

1. Time consuming
2. No transparency
3. Redundancy
4. No security (Hackers may hack the Mails)

COMMON REPOSITORY ORIENTED BUG REPORTING PROCESS:



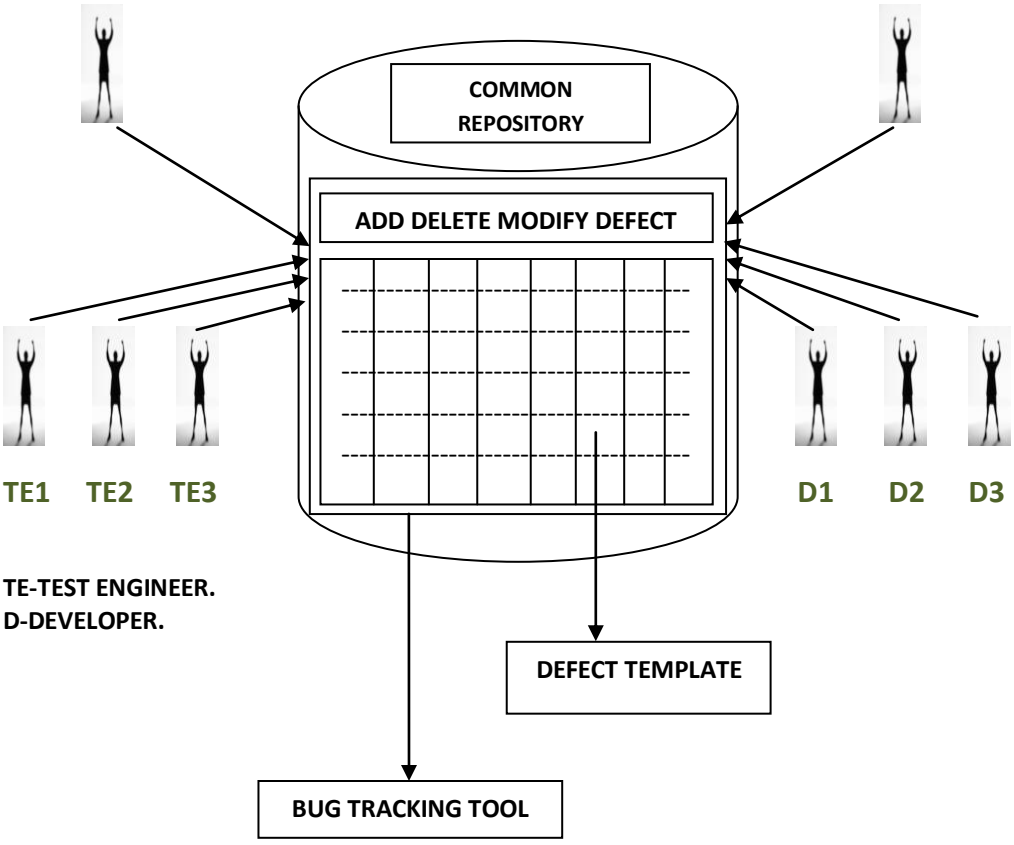
TE-TEST ENGINEER.
D-DEVELOPER.

COMMON REPOSITORY: It is a server which can allow only authorized people to upload and download.

DRAWBACKS:

1. Time consuming
2. No transparency
3. Redundancy (Repeating)

BUG TRACKING TOOL ORIENTED BUG REPORTING PROCESS:



BUG TRACKING TOOL: It is a software application which can be access only by the authorized people and provides all the facilities for bug tracking and reporting.

Ex: BUGZILLA, PR TRACKER, ISSUE TRACKER…..etc.

PR-PERFORMANCE REPORTER.

Ex:
No Transparency: Test engineer can't see what was happening in the development deportment and developer can't look what was the process is going in the testing deportment.
Redundancy: There is a chance that some defect will be found by all the test engineers.

Ex: Suppose all the test engineers found the defect of the login screen login button, and then they raise the same as a defect.

BUG TRACKING TOOL ORIENTED BUG REPORTING PROCESS:

The test engineer enter into bug tracking tool, he add defect to the template with add defect feature and writes the defect in corresponding columns, the test lead parallely observes it by bug tracking tool, and he assign Severity.

The development lead also enters into the bug tracking tool. He assigns the priority and assigns the task to the developer. The developer enters into the tool and understands the defect and rectifies it.

Tool: Something that is used to complete the work easily and perfectly.

Note: Some companies' use their own Bug Tracking Tool, this tool is developed by their own language, this tool is called 'INHOUSE TOOLS'.

TEST CLOSURE ACTIVITY: This is the final activity in the testing process done by the test lead, where he will prepare the test summary report, which contains the information like:

Number of cycles of execution,
Number of test cases executed in each cycle,
Number of defects found in each cycle,
Defect ratio and……etc.

TERMINOLOGY:

DEFECT PRODUCT: If at all the product is not satisfying some of the requirements, but still it is useful, than such type of products are known as Defect Products.

DEFECTIVE PRODUCT: If at all the product is not satisfying some of the requirements, as well as it is not usable, than such type of products are known as Defective Products.

QUALITY ASSURANCE: It is a dependent, which checks each and every role in the organization, in order to confirm whether they are working according to the company process guidelines or not.

QUALITY CONTROL: It is a department, which checks the develop products or its related parts are working according to the requirements or nt.

NCR: If the role is not following the process, the penalty given for him is known as NCR (Non Conformances Raised). Ex: IT-NCR, NON IT-MEMO.

INSPECTION: It is a process of sudden checking conducted on the roles (or) department, without any prior intimation.

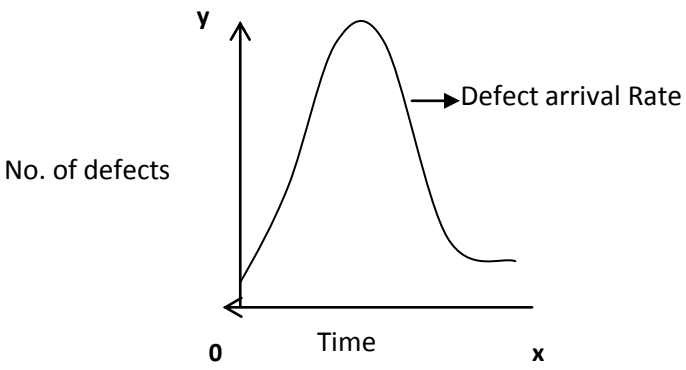
AUDIT: It is a process of checking, conducted on the roles (or) department with prior intimation well in advance.

There are 2 types of Audits:
1. INTERNAL AUDIT
2. EXTRNAL AUDIT

INTERNAL AUDIT: If at all the Audit is conducted by the internal resources of the company, than the Audit is known as Internal Audit.
INTERNAL AUDIT: If at all the Audit is conducted by the external people, than that Audit is known as External Audit.

AUDITING: To audit Testing Process, Quality people conduct three types of Measurements & Metrics.

1. QAM (Quality Assessment Measurement):
These measurements used by Quality Analysts / PM during testing process (Monthly once).



Stability:
20% testing → 80% defects
80% testing → 20% defects

Sufficiency:
→ Requirements Coverage
→Type-Trigger analysis

Defect Severity Distribution:
→ Organization- Trend limit check.

2. TMM (Test Management Measurement):
These measurements used by Test Lead during testing process (weekly twice).

Test Status:
→ Completed
→ In progress
→ Yet to execute

Delays in Delivery:
→ Defect arrival rate
→ Defect resolution rate
→ Defect age

Test Efficiency:
→ Cost to find a defect (No of defects / Person-Day)

3. PCM (Process Capability Measurement):
These measurements used by Project Management to improve capability of testing process depends on feed back of customer in existing maintenance software's.

Test Effectiveness:
→ Requirements Coverage
→Type-Trigger analysis

Defect Escapes (Missed defects):
→ Type-Phase analysis

Test Efficiency:
→ Cost to find a defect (No of defects / Person-Day)

CAPA (CORRECTIVE ACTIONS & PREVENTIVE ACTIONS):

CORRECTIVE ACTIONS: Whenever the role has committed repairable mistake, than the Corrective Actions will be taken care, in order to correct such type of mistakes.
PREVENTIVE ACTIONS: Whenever the role has committed irreparable mistake, than the Preventive Actions will be taken care, in order to correct such type of mistakes in future.
SCM (SOFTWARE CONFIGURATION MANAGEMENT): It is a process where in 2 tasks are perform:
1. CHANGE CONTROL
2. VERSION CONTROL

CHANGE CONTROL: It is a process of updating all the related documents, whenever some changes are made to the application, in order to keep the Documents and Applications in sink with each other.
VERSION CONTROL: It is a process in which one will take care of Naming conventions and Versions.

COMMON REPOSITORY: It is basically a server, which can be accessed only by the authorized people, where in they can store the information and retrieve the information.

CHECK IN: It is a process of uploading the information from the Common Repository.
CHECK OUT: It is a process of downloading the information from the Common Repository.
BASE LINE: It is a process of finalizing the documents.

PUBLISHING/PINNING: It is a process of making the finalized documents available to the relevant resources.

RELEASE: It is a process of sending the application from the development department to the testing department (or) from the company to the market.

DELIVERY: It is a process of sending the application from the company to the client (or) from the market to the client.

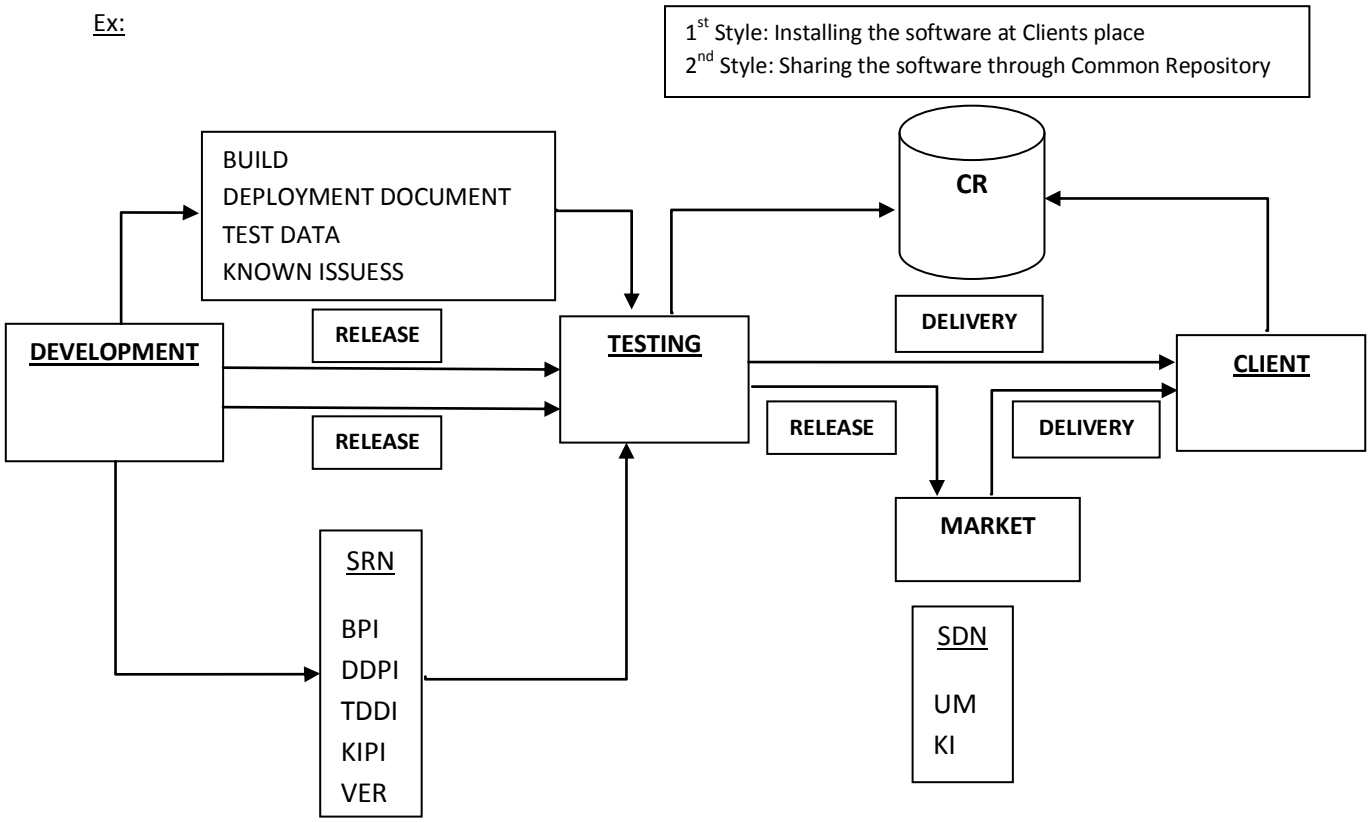
SRN (SOFTWARE RELEASE NOTE): It is a note prepared by the development department and sent to the testing department during the release, it contains the following information:

- Build path information.
- Deployment document path information.
- Test data path information.
- Known issues path information.
- Release manager name.
- Release date.
- Build number.
- Version number.
- Module name……etc

SDN (SOFTWARE DEVELOPMENT NOTE): It is a note prepared by team of members under the project managers guidance and given to the customer during the delivery, it contains the following information:

- User manual
- Known issues

Ex:



REVIEW: it is defined as either of process of studying (or) process of checking depending upon the role involved.

REVIEW REPORT: It is an outcome document of review, which may contain either list of doubts (or) list of comments.

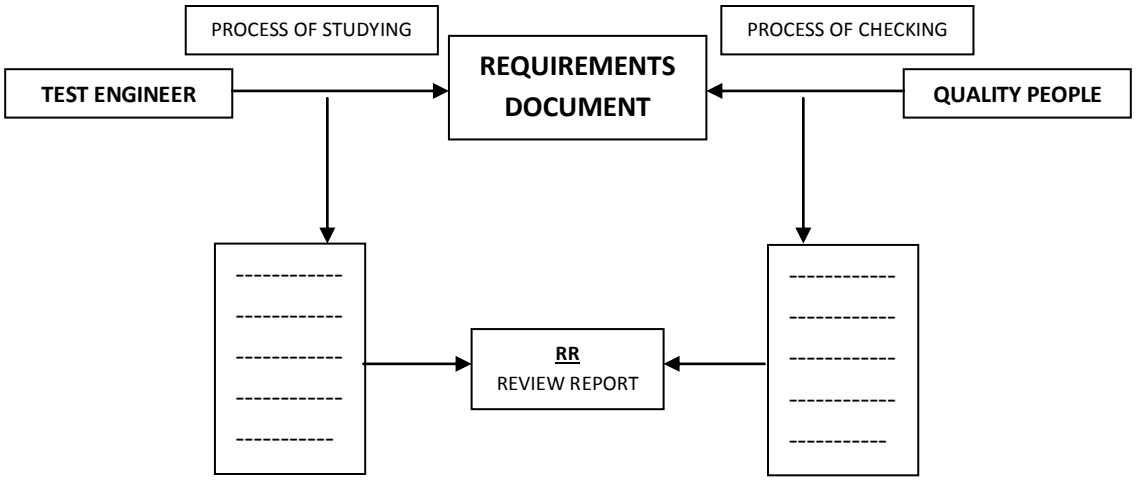
COLLEAGUES: People working in same company with different designations.

PEER: Colleagues with same designation in the company.

PEER REVIEW: It is a process of checking, conducted by the colleagues at same designation.

PEER REVIEW REPORT: It is an outcome document of peer review, which contains the list of comments.

Ex:



TEST SUIT: Combination of all the different types of test cases is known as Test Suit.

TEST BED: Combination of Test Environment and Test Suit.

HOT FIX: Fixing the defect immediately.

DEFECT AGE: The time gap between opening date and closing date of the defect is known as Defect Age.

LATENT DEFECT: The defect that is found late after some releases are known as Latent Defects.

SLIP AGE: The extra time taken to accomplish a task is known as Slip Age.

ESCALATION: It is a process of intimating the issue related information to the next level of authority.

METRICS: Clear measurement of any task is known as Metrics.

TRACEABILITY MATRIX: It is a document which contains a table of linking information used for tracing back for the reference in any kind of confusion or questionable situation.

PROTOTYPE: It is a roughly and rapidly developed model used for demonstrating to the client, in order to gather the clear requirements and also to win the confidence of ac customer.

TEMPLATE: It is predefined format, which is used for preparing a document easily and perfectly.

BENCH MARK: The standard with which usually we compare with is known as Bench Mark.

CHANGE REQUEST: It is a process of requesting the changes; to do the same customers will use the change request template.

REQ ID.	PURPOSE OF CHANGE	DESCRIPTION OF CHANGE	MODULE NAME

IMPACT ANALYSIS: Whenever the customer proposes some changes, the senior analyst will analyze, How much impact will fall on the already developed part? This process is known as Impact Analysis.

WALK THROUGH: It is defined as an informal meeting between two or more roles, may be for checking something (or) for transferring something.

CODE WALK THROUGH: It is a process of checking conducted on the source code document, in order to confirm whether it is developed according to the coding standards or not.

CODE OPTIMIZATION/FINE TUNING: It is a process of reducing the number of lines of code (or) complexity of code, in order to increase the performance.

PPM (PERIODIC PROJECT MEETING): It is a meeting conducted periodically, in order to discuss the status of the project. Usually they discuss the following points:

- Percentage covered in the project during the period
- Percentage not covered in the project during the period
- Tasks completed during that period.
- Defects found during that period.
- Slip ages.
- Reasons for the slip ages.
- Technical issues.
- HR related issues.

PPR (PERIODIC PROJECT REPORT): It is a report prepared by Team Lead by interacting with the team members before the PPM is conducted. This document contains the information related to all the above said points.

MRM (MANAGEMENT REPRESENTATIVE MEETING): It is a meeting conducted, in order to discuss the status of the company. Usually they discuss the following points:

- Success rate and growth rate of the company.
- Projects that are recently signed off.
- Projects that are in pipe line.
- Customers' appraisals (Good comments).
- Future plans.
- Internal audit reports.
- Individual appraisals.

PATCH: Whenever the test engineers suspend a build the developers will rectify the problems and release the same build as Patch.

WORK AROUND: A workaround is a method, sometimes used temporarily, for achieving a task or goal when the usual or planned method isn't working. In information technology, a workaround is often used to overcome hardware, programming, or communication problems. Once a problem is fixed, a workaround is usually abandoned.

WAYS OF TESTING:

There are 2 ways of Testing:

1. **MANUAL TESTING**
2. **AUTOMATION TESTING**

1. MANUAL TESTING: Manual Testing is a process, in which all the phases of STLC (Software Testing Life Cycle) like Test planning, Test development, Test execution, Result analysis, Bug tracking and Reporting are accomplished successfully and manually with Human efforts.

DRAWBACKS:

1. More no of human resources are required.
2. Time consuming.
3. Less accuracy.
4. Tiredness.
5. Simultaneous actions are almost impossible.
6. Repeating the same task again and again in same fashion is almost impossible.

2. AUTOMATION TESTING: Automation Testing is a process, in which all the drawbacks of Manual Testing are addressed properly and provides speed and accuracy to the existing testing process.

DRAWBACKS:

1. Automated tools are expensive.
2. All the areas of the application can't be tested successfully with the automated tools.
3. Lack of automation Testing experts.

Note: Automation Testing is not a replacement for Manual Testing, it is just continuation for Manual Testing.

Note: Automation Testing is recommended to be implemented only after the application has come to a stable stage.

PC SURENDRA REDDY M.C.A.

Suri.poluchalla@live.com