

Software

A set of Executable programs is called software.

Application / Project

If a software is developed based on specific customer requirements, then that software is called as Application / Project.

if a software is developed based on proposal from specific companies, then that software is called as Application / Project.

Product

If a software is developed depending on overall requirements in the market, then that software is called as Product.

with Company's Self Interest / own proposal, a new software is developed for the market, that software is called product.

SDLC : Software Development Life Cycle

During project or product development, organizations follow an engineering process called as SDLC. SDLC Models are classified in to old models and new models.

Software Product Development Models	
Old Models	New Models
a) Waterfall	a) FISH Model
b) Spiral	b) V Model
c) Prototype	c) AGILE Model
d) Incremental	
e) RAD	
These models are without Testing Team.	These models are with Testing Team.

Old SDLC Models

a) Waterfall Model

This model is also called as linear sequential model. In this model, one stage will be starting only after completion of previous stage. This model is followable in companies when customer requirements are clear (when there is no confusion in customer requirements)

Customer (Project / Application)		Software Bidding / Proposal (Product)
Kick of Meeting		CEO
PM - Project Manager	PIN (Project / Product Initiation Note)	(overall plan)
BA - Business Analyst	Requirements Gathering (BRS)	
SA - System Analyst	Analysis & Planning	
approval on Analysis & Planning		PM
SRS		SA
Design	Designer / Technical Architect	
Coding	Programmers	
Testing	Programmers	
Release & Maintenance	CCB	

Software Bidding

Interest to develop new software's is called Software Bidding.

Kick of Meeting

Selection process of PM (Selecting the reasonable person to handle the project)

PM

Project Manager / Product Manager

PIN

Project / Product Initiation Note (Overall Plan)

I

Investment

ROI

Return of Investment

BA

Business Analyst

Responsibilities of BA

Talking to the client

Gathering the real requirements of client

Knowledge of Banking / Insurance / Health Care Domain

SME

Subject Matter Expert / Subject Matter Expertise

BRS

Business requirement specification

SA

System Analyst

SRS

Software Requirement Specifications

CCB

Change Control Board

Responsibilities of CCB

Support / Maintenance / Service

Releasing software to customer

solve problems in future

b) PROTOTYPE Model

Organizations can follow this model, when customer requirements are not clear. To get clear and complete requirements, organizations are developing prototypes before going to real software development.

Prototype

Screens without functionality (Sample / dummy screens)

- * Sample screens like power point slide show are shown to customers when requirements not clear.
- * completeness and correctness is very important.
- * nice look & feel screens are shown to customers to collect accurate information.

Software

Screens with functionality

c) Incremental Model

Organizations are following this model when customer requirements are huge. Here, organizations are releasing a software installment by installment.

d) Spiral Model

Organizations are following this model when customer requirements are enhancing regularly.

e) RAD Model : Rapid Application Development

- * releasing new software using existing software coding.
- * organizations are following this model, when customer requirements are known.
- * here, new software will be integrated by using existing components of old projects.

Note :

1. In all the old SDLC models, testing stage comes after coding only.
 2. This single stage of testing is conducted by the same programmers / developers who perform coding. Due to this, organizations are not able to release quality software to customer or to market.
- * To develop and release quality software, organizations are following new SDLC models. In new SDLC models, organizations conduct multiple stages of testing and maintain separate testing teams.

New SDLC Models

(a) FISH Model

This model is followed in organizations, when customer requirements are clear. But, this model is time consuming & costly. Advantage is 100% quality.

(b) V Model

V stands for verification and validation. In this model, organizations are conducting multiple stages of testing. But are maintaining separate testing team for software testing stage only in order to save time and reduce project cost.

Quality Software

- * meet customer requirements (functionality)
- * meet customer expectations (performance, usability, compatibility, security, ...)
- * cost to produce license.
- * Time to release

To develop a quality software, organizations follow advanced / new SDLC models instead of old SDLC models. In new SDLC models, V model is reasonable to follow as it is time saving and cost effective.

Testing Stages or Phases in 'V' Model

To decrease project cost, save time & release quality software, organizations are following 'V' Model. In this model, multiple stages of development is mapping with multiple stages of testing.

(i) Documents Testing / Verification

In Software Development Process, requirements gathering will be done by Business Analyst (BA). Here, BA prepares a document based on customer requirements called **BRS** / CRS / URS.

BRS : Business Requirement Specifications.

CRS : Customer Requirement Specifications.

URS : User Requirement specifications.

This BRS will be reviewed by the same BA for completeness & correctness.

Once BRS is base-lined, system analyst (SA) prepares SRS (Software Requirement Specification) by analyzing BRS.

Example

BRS	SRS
Addition of Two Numbers	FRS : Functional Requirement Specifications
	* 2 inputs * + operator * 1 output
	NFRS : Non Functional Requirement Specifications
	* Blue Color in Screen. * Run on Windows & Linux. * Addition within 0.5 Seconds.
What to Develop	How to Develop

In the above example, we can observe that SRS is divided in to FRS & NFRS. Here, FRS consists of customer requirements to be developed and NFRS consists of customer expectations to be achieved.

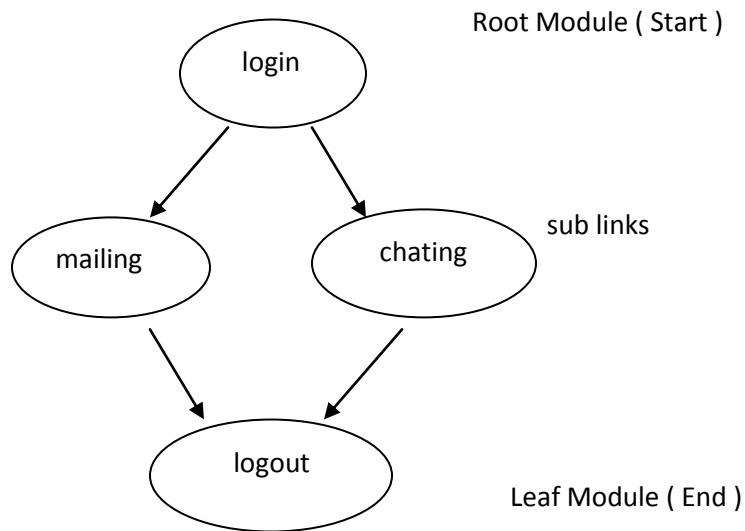
After completion of SRS preparation, same system analyst reviews the SRS for its completeness & correctness.

Once SRS is base-lined, Designers / Technical Architects concentrate on Software Design in High Level & Low Level.

High Level Design shows the total architecture of Software. Low Level Design provides the internal architecture or logic for a specific module. That's why one project has one HLD and multiple LLD's.

Example

We can observe 4 modules in the project below



Project Level : HLD

Module Level : LLD's

After completion of HLD & LLD's, corresponding designers review those designs for completeness & correctness.

Case Study

Document	Prepared by	Reviewed by	Techniques
BRS	BA	Same BA	* walk-through * Inspection * Peer review
SRS	SA	Same SA	* walk-through * Inspection * Peer review
HLD	Designer / TA	Same Designer / TA	* walk-through * Inspection * Peer review

walk-through : walk-through means that a document studied from first to last.

Inspection : Inspection means that searching a factor in document

peer review : comparison of 2 similar documents.

Above 3 Techniques are also called as Documents Testing Techniques or Verification Techniques or Review Techniques or Static Testing Techniques.

(ii) Unit Testing

Once the Design Documents are base-lined, programmers can start coding by writing programs. The same programmers test these programs for completeness and correctness. This process is called Unit Testing.

In Unit Testing, White Box Testing Techniques are followed by Programmers.

a) Basic Paths Coverage

This Technique is used by programmers to confirm that the program is running without any run time errors.

- * Write the program
- * convert the program logic in to flow graph
- * calculate the number of paths in that graph
- * run the program that many times to cover entirely.

b) Control storage coverage

Programmers use this technique to confirm that program is running correctly in terms of inputs and outputs.

Example :

Debugging : Run a program line by line and check correction of inputs and outputs.

c) Program Technique Coverage

Programmers use this technique to confirm whether the program is running correctly or not.

d) Mutation coverage

Mutation means a change in program. Programmers use this technique to confirm that the program is tested completely or not.

Example :

Debugging : Testing of Testing on Program.

(Intentionally do some mistakes in program and perform testing)

* In this technique, programmers perform a change in tested program and repeat testing once again. This mutation coverage is also called as Bebugging.

Unit Testing	
Basic Paths coverage	Control Storage coverage
Program Technique coverage	Mutation coverage

These 4 white box testing techniques used by programmers in unit testing are also called as Glass Box testing Techniques or open box testing techniques or clear box testing techniques

(iii) Integration Testing

After completion of related programs writing and unit testing, corresponding programmers can integrate those programs and conduct integration testing.

Integration : connecting 2 programs

Integration testing : testing whether programs connected correctly or not

while integrating programs, programmers follow any one of the approaches mentioned below

a) top - down approach

In this approach, programmers are integrating programs without involvement of under constructive sub modules. STUB is a temporary program to send back run control to main module, instead of under constructive sub module.

b) bottom - up approach

In this approach, programmers are integrating sub modules without involvement of under constructive main module. DRIVER is a temporary program that is used instead of under constructive main module.

c) hybrid approach

The combination of top - down & bottom - up approaches is called as hybrid approach or sand witch approach.

Note : The 3 approaches mentioned above are also called as Incremental Integration approaches.

d) system approach or big bang approach

In this approach, Integration will be starting only after the coding is completely completed 100%.
(No need of STUB or DRIVER in this approach)

(iv) Software Testing

After completion of Integration and Integration testing. corresponding programmers release software to Testing Team. Testing Team conducts Functional Testing and Non Functional Testing on that software with respect to System Requirement Specifications.

Software testing	
Functional testing on software with respect to (F.R.S.) Functional requirement specifications in (S.R.S) software requirement specifications.	Non Functional testing / system testing / characteristics testing on software with respect to (N.F.R.S.) Non Functional requirement specifications in (S.R.S) software requirement specifications.

1. Functional Testing

It is a mandatory testing topic in software testing. During this text, testers can apply the following sub tests on software to validate.

a) GUI testing (Graphical User Testing) / Control flow testing / behavioral testing

Testing whether the software screens are behaving / responding correctly or not while navigation (operating them).

b) Input domain testing

Testing whether software screens are taking correct inputs or not.

c) Error handling testing

Testing whether software screens are returning error messages or not when we operated those screens in wrong manner.

d) Manipulations testing

Testing whether software screens are proving correct outputs or not with respect to given inputs.

e) Database testing (back end)

Testing whether software screens are storing and manipulating data in corresponding database or not.

Data Validation : New data correctly stored or not.

Data Integrity : due to new data storage, existing data got modified or not.

* correctness of change in existing data based on given input is called as data integrity.

* Perform operation on front end - observation on back end

Note :

In the above functional testing topics, first 4 sub-tests are considered as front end testing or screens testing.

Database testing topic is called as Back End Testing.

2. Non Functional Testing

Non Functional Testing is also called as System testing or Characteristics testing.

After completion of functional testing, corresponding testing team is concentrating on non functional testing. In non functional testing, we have 10 sub tests as mentioned below.

a) Usability testing

During this test, testing team is concentrating on the following expectations from corresponding SUT. SUT : Software under Testing.

- * ease of use
- * look and feel
- * short navigations
- * understandable HELP (user manuals - HELP documents of that software)

b) compatibility testing

This testing is also called as portability testing. During this test, testing team can operate SUT in various platform based computers. Here, platform means that computer operating system, browsers and other system software's.

c) configuration testing

This testing is also called as hardware compatibility testing .

During this test, testing team can confirm whether our SUT supports hardware devices belonging to different technology or not.

Example :

Different types of Networks, Different types of printers, Different types of fax machines, . . .

d) Intersystem testing

This testing is also called as inter-operability testing or end to end testing (front end one software to back end of other software) or web services testing or SOA testing (service oriented application testing)

- * During this test, testing team can confirm that whether our software is sharing resources of other software's or not.
- * Due to globalization in the world, companies are using inter connecting software's to share resources.
- * software's that are helping the software's are called as service oriented software's.
- * software's that are using the services of other software's are called as service utilizers.
- * testing module to module of 2 different software's is inter system testing
- * testing module to module of same software is integration testing.

e) data volume testing

This testing is also called as capacity testing or memory testing.

During this test, testing team is calculating the capacity of corresponding software database by entering sample data (bulk dummy data) to the database.

This test is time consuming, but not costly.

f) performance testing

During this test, testing team is concentrating on 4 sub tests

1	Load testing	Reasonable load applied on software
2	Stress testing	More than reasonable load applied on software
3	Spike testing	huge load applied on software
4	Endurance testing (longevity testing)	Software Working for a long time or not, even after applying reasonable load

load testing

The execution of SUT under customer expected configuration and customer expected load (number of concurrent users) to estimate speed in processing is called as load testing.

Stress testing

The execution of SUT under customer expected configuration by applying more than customer expected load to estimate peak load (maximum load, the project can handle) is called as stress testing.

spike testing

The execution of SUT under customer expected configuration and huge load to estimate reliability is called as spike testing.

endurance testing

The execution of SUT under customer expected configuration and customer expected load repeatedly or iteratively to estimate longevity or durability is called as endurance testing.

* longevity means that a software is working for a longer period of time without any memory leakages.

g) multi languity testing

During this test, testing team is validating SUT functionalities by entering inputs in various languages, when that software screens are developed in Java or .Net or other Unicode technologies.

* There are two ways in multi languity testing

Localization	Globalization
--------------	---------------

h) security testing

This testing is also called as penetration testing.

During this test, testing team concentrates on the following sub tests

Authorization / Authentication testing

means that the software can allow valid users and prevent invalid users.

Access control testing

means that the software can provide permissions to restrict users.

Encryption Decryption testing

one needs to have knowledge of hacking to conduct this test.

i) Installation testing

During this test, testing team is checking whether the software is easy to install on customer expected configured system or not.

* " setup" program execution to start initialization.

* easy screens during installation.

* occupied disk space after installation.

* easy to uninstall.

j) parallel testing

This testing is also called as competitive testing or comparison testing.

During this test, testing team is comparing SUT with previous versions of same software or with competitive products in the market to find weaknesses and strengths.

parallel testing is applicable only for products (product based software's)

(V) Acceptance testing

After completion of Software testing, corresponding project manager concentrates on acceptance and acceptance testing.

In this stage, developers and testers are also involved either directly or indirectly.

During this test, real customers or model customers will give feedback on corresponding software.

There are 2 ways of acceptance testing

Alpha test	Beta test
For software applications / projects	For software products
Invite real customer site people to collect feedback on software (face to face)	Release a trial version of the software to model customers to collect feedback (online)
Developers and testers can involve with real customer site people directly.	Developers and testers can involve indirectly to read model customers feedback mails.
Yellow box testing	

From this acceptance testing, project management is getting an idea on changes to be needed in the developed software.

if changes are needed, then developers are performing the corresponding changes in that software and testers are approving those changes for completeness and correctness.

(VI) Software release and release testing

After completion of acceptance, corresponding project management can select few developers and few testers along with few hardware engineers to go to real customer site or license purchased customer site for software release.

The remaining developers and testers in the team can go to bench (wait for next project)

The selected developers and testers chosen for release of software are called as release team or onsite team or delivery team.

This team performs the tasks mentioned below in the corresponding customer site

- * Complete installation of software.
- * Verify overall functionality of software.
- * support from input devices (Keyboard, mouse, . . .)
- * support from output devices (monitor, printer, . . .)
- * support from operating system and other system software's.
- * co-existence with other software's.
- * support from secondary storage devices (external hard disk, pen drive, cd drive)

The tasks mentioned above are referred as green box techniques.

After completion of above observations in customer site, corresponding release team can conduct training sessions for end users or customer site people.

at the end, release team will come back from customer site and go to bench.

(VII) maintenance / support / service level testing

During utilization of a software, customer site people send change requests to corresponding company.

To handle those change requests, project management establishes a team called CCB (Change Control Board)

CCB members have both development and testing skills.

Software Change Request (SCR)	
Enhancement	Mistake / missed defect / latent defect
Analyze that enhancement	Route cause analysis to identify wrong coding areas
Perform changes in coding to achieve enhancement	Change coding to fix defect
Test those changes for correct enhancement	Test correctness of fixed defect
Release patch to customer site	Release patch to customer site
	Improve testing process and development process for future projects / products
Adaptive maintenance / enhance maintenance	Corrective Maintenance

This testing involves Grey Box Techniques (White Box + Black Box)

Case Study

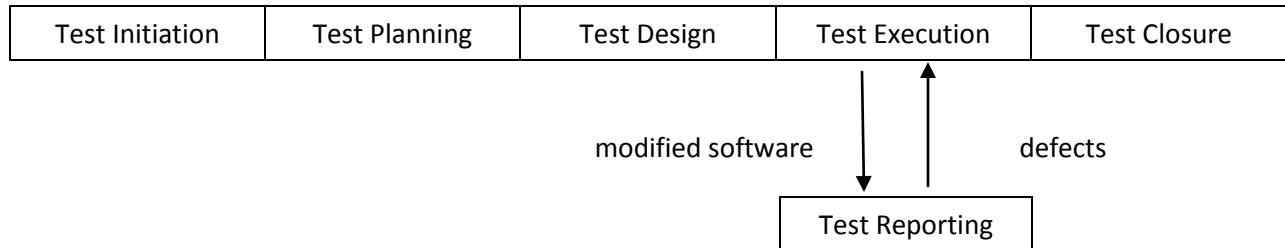
Testing stage / phase	Purpose	Conducted by	Testing techniques
Documents testing	For completeness & correctness of BRS, SRS, HLD, LLD's	Authors of Documents	Static testing techniques
Unit testing	Testing programs with respect to LLD's	Programmers	White box techniques
Integration testing	Testing on programs interconnection with respect to HLD	Programmers	White box techniques
Software testing	Testing on software with respect to SRS	Testers	Black box techniques
Acceptance testing	Collecting feedback	Real / model customer site people	Yellow box techniques
Release testing	Correctness and completeness of software in customer site	Release team	Green box techniques
Maintenance testing	Test software changes	CCB	Grey box techniques

SOFTWARE TESTING PROCESS

To release quality software to customers, organizations are conducting multiple stages of testing while developing software.

In those testing stages, a separate testing team is conducting software testing i.e. testing on a complete software with respect to customer requirements and customer expectations as specified in SRS.

During this software testing, testing team follows the process as shown below



Software development process v/s software testing process

Note

1. Programmers and testers will sign-in / role-in to a project / product, when SRS is base lined.
2. Testers will start test execution to detect defects in software, when developers release initial software build and when these testers finished writing test cases.
3. Programmers and Testers will roll-off / sign-off, when acceptance testing is finished by real customers / model customers.

SOFTWARE TEST INITIATION

Software testing process starts with Test Initiation stage by (PM) Project Manager or (TM) Test Manager.

Project Manager prepares a document in this stage which is called as Test Strategy or Test Methodology.

This document specifies required testing approach to be followed by testing team in current project / product.

while preparing this strategy document for current project testing, PM can follow any one of the following 3 ways.

1. Exhaustive testing
2. optimal / formal testing
3. Adhoc / informal testing

* From the above 3 strategies, exhaustive testing is impossible to follow because this type of testing needs huge time and resources. From testing principles also, exhaustive testing is considered as of no need.

In most of the organizations, testing teams are going for optimal testing strategies to release quality software to customers. In rare cases, some organizations testing teams are following adhoc testing strategy due to some risks.

To prepare optimal test strategy, PM follows IEEE (Institute of Electrical and Electronic Engineering) 829 format. According to IEEE format, 829 stands for software testing and 830 stands for software development. IEEE format is applied for, it is universal and easily understandable.

from that format

1. Scope and objective : importance of testing in current project.

2. Business Issues : Budget allocated for software testing.

100 % project cost	
64 %	36 %
Development & Maintenance	Testing

3. Test Responsibility Matrix (TRM)

Software Testing Topic	Yes / No	Comments
Functional Testing	Yes	-
Usability Testing	Yes	-
Compatibility Testing	No	No Resources
Security Testing	No	No Skills
Configuration Testing	No	No Requirements

This table can list out all reasonable testing topics to be conducted in current project.

4. Roles and Responsibilities

Roles	Responsibilities
Test Lead	Prepare Test Plan(s)
	Review Test Cases and Track Defects
	Coordinate Testers (Daily, Weekly, Monthly)
Sr. Tester	Prepare Test Cases
	Defect Reporting
	Coordinate with Jr. Testers
Jr. Tester	Execute Test Cases on Software under Testing (SUT)
	Defects Identification

5. Communication and Status Reporting

Required negotiations between Testers and Negotiation Channels

Example : Personal Meetings, e - mails, Chating, . . .

6. Manual Testing vs Automated Testing

Need for Test Automation in Current Project and available Testing Tools in our Organization

Manual Testing	Automated Testing
Tester	Tester
Test Cases	Test Cases
Software	Test Scripts
	Software

7. Defect Reporting and Tracking

Required Negotiations in between Testers and Developers during Defects Reporting and Tracking.

8. Testing Measurements and Metrics

List of Measurements and Metrics to estimate Tester's Performance / Tester's Efficiency.

Example :

30 Test Cases prepared per day (per day means 8 working hours in an organization)

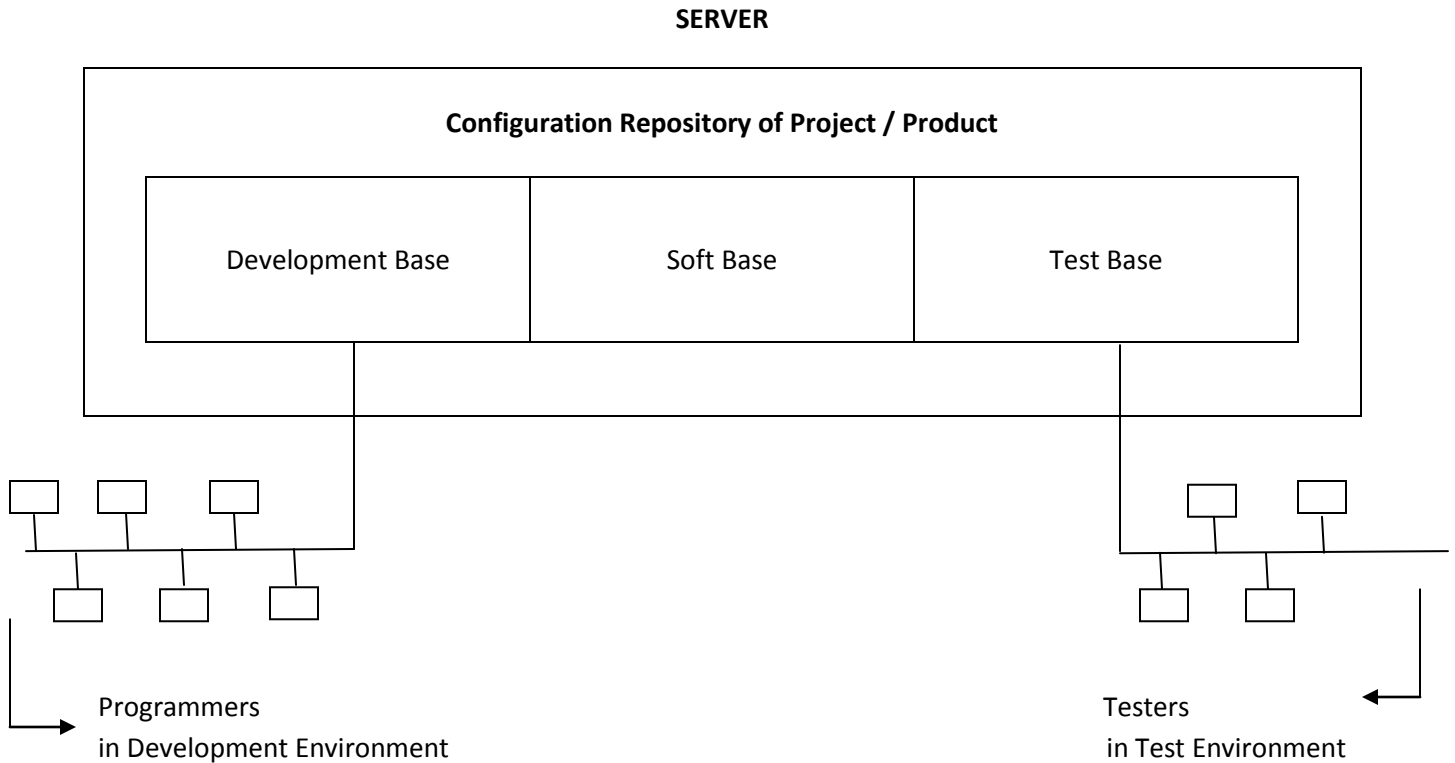
20 Test Cases executed per day.

5 defects detected per day.

9. Configuration Management

A folder is created in company's server, which will be used by developers and testers to store their documents.

Example



10. Risk and Mitigations

list out challenges that will come during testing and solutions to overcome their risks.

11. Training plan

Need for training in current project for Testing People.

(KT : Knowledge Transfer)

Note : The above IEEE 829 format Optional Test Strategy document will be forwarded to test lead by PM.

ii) TEST PLANNING

Depending on PM's Test strategy, corresponding Test Lead can prepare Test Plan / Plans in the process as shown below

S.R.S. of S.A.	* Team Formation * Identify Tactical / Practical Risks * Prepare Test Plan(s) * Review Test Plan(s)	Test Plan(s)
Project Plan of PM		
Test Strategy of PM		

a) Team Formation

Test Lead job starts with Testing Team Formation in a project. In this stage,

Test Lead can take the help of PM and HR people to form a Testing Team depending on the factors mentioned below.

- * Project Size

Example : Number of Functional Points)

- * Test Duration (available time for testing)

- * Availability of Testers on the Bench.

- * Availability of Testing Tools.

Note : Software's used in specific devices are called as Embedded Systems or Networking based Software's.

Case Study on the Formation of Testing Team

Type of Software Project / Product	Developers : Testers	Test Duration
Client / Server and Web based Software (Banking, Insurance, e-commerce, other Commercial Software's)	3 : 1	3 to 5 Months
Embedded and Networking based Software's	1 : 1	7 to 9 Months
Artificial Intelligence (A.I.) and Expert Systems (Satellite, Robotics, Expert Games, Aviation Projects, . . . e.t.c.)	1 : 7	12 to 15 Months

Note : when Complexity is increasing in projects, number of testers in a project are increasing.

b) Identify Tactical Risks

Risk indicates a Future Problem.

Test Lead can identify some Risks with respect to the Testing Team that is formed in Current Project / Product.

Few Examples of RISK	
Risk 1	Lack of Time
Risk 2	Lack of Documentation
Risk 3	Lack of Resources
Risk 4	Delays in Delivery
Risk 5	Lack of Seriousness in Development People
Risk 6	Lack of Skills
Risk 7	Lack of Communication

c) Prepare Test Plan(s) in IEEE 829 Format

After completion of Testing Team Formation and Risks Identification, corresponding Test Lead can prepare Test Plan document in IEEE 829 Format as shown below.

1. Test Plan Id

Unique Number or Name for Future Reference.

2. Introduction

about Current Project / Product.

3. Features / Modules

List of all Modules in Current Project / Product.

3	4	5	What to Test
---	---	---	--------------

4. Features to be Tested

List of Modules to be tested in Current Project / Product.

5. Features not to be Tested

List of Modules which are already tested.

6. Approach

Attach Test Strategy given by PM.

7. Test Environment

Required Hardware and Software for Testing Team in Current Product / Project testing including Testing Tools.

8. Test Deliverables

Documents to be prepared by Testers during Current Project / Product Testing.

Examples :

- * Test Scenarios
- * Test Cases Documents *
- * Automation Test Scripts
- * Test Logs
- * Defect Reports* & Review Reports (Daily, Weekly and Monthly)

9. Entry Criteria

To start Test Execution by Testers.

- * After Test Cases Preparation and Review.
- * After establishing Test Environment.
- * After receiving Initial Software Build from Developers.

10. Suspension Criteria

To interrupt Test Execution by Testers.

- * More defects are in pending (queue) without getting fixed.
(Technical word used for this is Quality Gap)
- * Show Stopper in Software (High Severity in Defect)
- * Test Environment abandoned (System Problem)

6	7	8	9	10	11	How to Test
---	---	---	---	----	----	-------------

11. Exit Criteria

To Stop Test Execution by Testers.

- * All Major Defects are Fixed and Closed.
- * All Features / Modules are Tested.
- * Test Duration Exceeded.

12. Staff and Training Needs

Names of Selected Testers and need for training to them to understand current project or product registration.

13. Responsibilities

Work Allocation to above listed testers in modules wise or in Testing Topics wise.

14. Schedule

Dates and time for Testing in Current Project / Product.

12	13	Who to Test	14	When to Test
----	----	-------------	----	--------------

15. Risks and Assumptions

previously analyzed Risks and Solutions to overcome them.

16. Approvals

Signatures of Test Lead and PM.

Scenario	What to Test	Case	How to Test
----------	--------------	------	-------------

d) Review Test Plan

After completion of Test Plan Document preparation, corresponding Test Lead can get approval from PM and forward that Plan Document to all selected testers in the current project / product (Juniors and Seniors included)

iii) Software Test Design

All Selected Test Engineers can follow Test Plan given by Test Lead and do below Tasks in Test Design Stage.

- a) Understand all Requirements.
- b) Prepare Test Scenario's for responsible Modules and responsible Testing Topics (High Level Test Design - HLTD)
- c) implement Test Scenario's as detailed Test Cases (Low Level Test Design - LLTD)

a) Requirement Study

In an organization, every tester job starts with corresponding project / product requirement study because software testing depends on requirements described in S.R.S.

To understand requirements in S.R.S., testers can follow

- * walk through S.R.S.
- * Training on S.R.S. from S.A.
- * Training on S.R.S. by S.M.E. (Subject Matter Expert - outsider)
- * Discussions with Sr. Testers, Developers, Leads . . . (K.T. - Knowledge Transfer from Colleagues)
- * Internet Browsing.

b) High Level Test Design

After understanding corresponding Product / Project requirements, every Test Engineer can prepare Test Scenario's for responsible Modules and responsible Testing Topics.

There are 4 methods to prepare Test Scenario's and Test Cases

One) Functional Specifications based Test Design.

Two) Use Cases based Test Design.

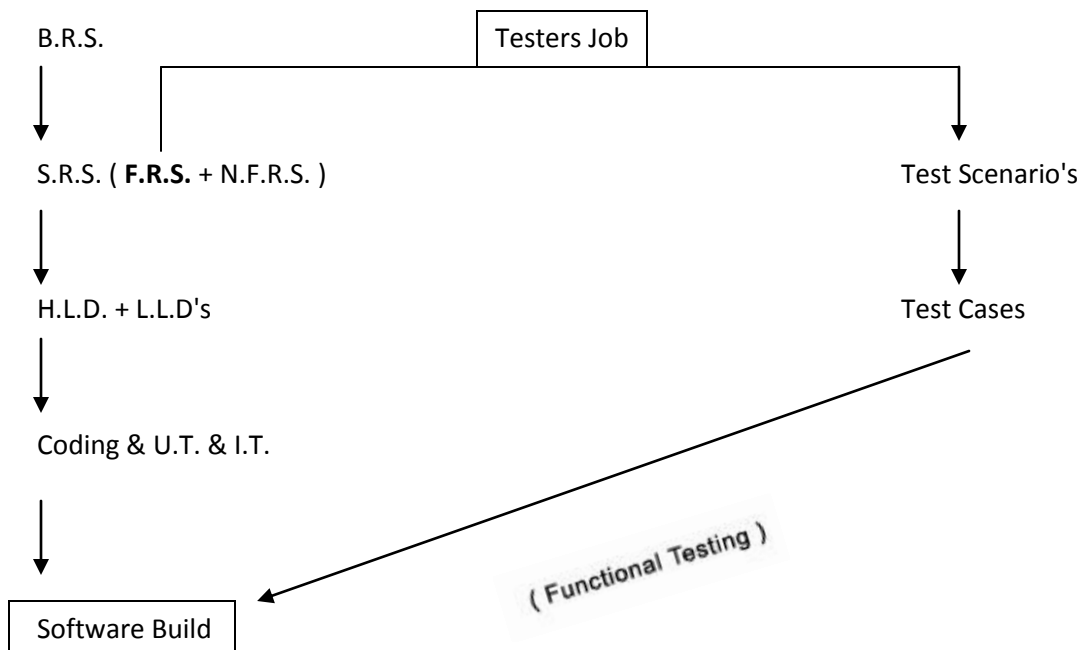
Three) Screens based Test Design.

Four) Non Functional Specifications based Test Design

For Software Testing Design			
For Functional Testing			For Non Functional Testing
One	Two	Three	Four

One) Functional Specifications based Test Design

Testers are following this method to prepare Test Scenario's and Cases for responsible Modules Functional Testing.



Black Box Testing Techniques

while Preparing Test Scenario's and Cases to conduct Functional Testing Optimally, Testers are following Black Box Testing Techniques. These Techniques are also called as Closed Box Testing Techniques.

1. BVA - Boundary Value Analysis

BVA Technique is used to define Range or Size of Inputs or Outputs.

2. ECP - Equivalence Class Partitions

ECP Technique is used by Testers to specify Valid and Invalid Type of Inputs and Outputs.

3. DT - Decision Table

DT Technique is used to define Mapping in between Inputs and Outputs.

4. OA - Orthogonal Arrays

OA Technique is used to avoid redundancy / repetitions in Decision Table to Save Time.

5. STF - State Transition Flow

STF Technique is used by Testers to write Scenario's in an Order with respect to Functionality.

6. EG - Error Guessing

EG Technique is used by Testers to guess defects before conducting Testing, depending on Past Experience.

Functional Specification 1

In an Insurance Software, Agents are doing **Login** by entering **Username** and **Password**

From requirements, **Username** is taking Alphanumerics in lower case from 4 to 16 characters long.

Password is taking alphabets in Lower Case from 4 to 8 Characters.

Insurance Agents can Click **OK** Button to do **Login** and Click **Cancel** Button to **Close Login window** at any time.

Prepare **Test Scenario's** for **Login** Module **Functional Testing**.

Test Scenario 1

Check / Verify / Validate **Username** Field

B.V.A. (Size)

Minimum	4 Characters	Passed
Minimum - 1	3 Characters	Failed
Minimum + 1	5 Characters	Passed
Maximum	16 Characters	Passed
Maximum - 1	15 Characters	Passed
Maximum + 1	17 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Alphanumerics in Lower Case	Alphabets in Upper Case
	Special Characters
	Blank Field

Test Scenario 2Check / Verify / Validate **Password** Field**B.V.A. (Size)**

Minimum	4 Characters	Passed
Minimum - 1	3 Characters	Failed
Minimum + 1	5 Characters	Passed
Maximum	8 Characters	Passed
Maximum - 1	7 Characters	Passed
Maximum + 1	9 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Alphabets in Lower Case	Alphabets in Upper Case
	Numbers
	Special Characters
	Blank Field

Test Scenario 3Check / Verify / Validate **Login** Operation by Clicking **OK** Button

D.T. (based on Permutations and Combinations)

User Id	Password	Expected Output after Clicking OK
Valid	Valid	Next Window
Valid	Invalid	Error Message
Invalid	Valid	Error Message
Invalid	Invalid	Error Message
Blank Field	Value	Error Message
Value	Blank Field	Error Message
Blank Field	Blank Field	Error Message

Applying 2 Scenario's in the above Table : **1. Decision Table 2. Orthogonal Arrays**

User Id	Password	Expected Output after Clicking OK
Valid	Valid	Next Window
Valid	Invalid	Error Message
Invalid	Valid	Error Message
Blank Field	Value	Error Message
Value	Blank Field	Error Message

We can observe that after applying Orthogonal Arrays, the size of Decision Table is reduced by 2 Rows.

Test Scenario 4Check / Verify / Validate **Login Window Closing** by Clicking **Cancel** Button

D.T.

User Id	Password	Expected Output after Clicking Cancel Button
Value	Value	Window Closed
Value	Blank	Error Message
Blank	Value	Error Message
Blank	Blank	Error Message

Functional Specification 2

After Successful **Login**, Insurance Agents are Creating Policies.

While Creating a Policy, Agent need to fill the following fields

Policy Holder Name

Alphabets in Upper Case as one or more words

Age

15 to 60

Telephone Number

Valid in India

Address

Alphabets as one or more words and having Numbers and Special Characters like , / -

After filling above fields, corresponding Agent can click **SUBMIT** Button to get **Policy Number** as Output.

Prepare **Test Scenario's** for **Policy Creation** Module **Functional Testing**.

Note
<ul style="list-style-type: none">* When Document is not providing clear information in terms of Size of Input, we can take minimum size as 1 character and maximum size as 256 characters.* Blank Space : One Space gap between / separating two words.* Blank Field : When an Input Field is Left Empty.

Test Scenario 1Check / Verify / Validate **Policy Holder Name** Field**B.V.A. (Size)**

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Alphanumeric in Upper Case	Alphabets in Lower Case
	Numbers
	Special Characters excluding Blank Space
	Blank Field

Test Scenario 2Check / Verify / Validate **Age** Field**B.V.A. (Range)**

Minimum	15	Passed
Minimum - 1	14	Failed
Minimum + 1	16	Passed
Maximum	60	Passed
Maximum - 1	59	Passed
Maximum + 1	61	Failed

E.C.P. (Type)

Valid	Invalid
Numbers	Alphabets
	Special Characters
	Blank Field

Test Scenario 3Check / Verify / Validate **Telephone Number** Field**B.V.A. (Size)**

Minimum	10 digits	Passed
Minimum - 1	9 digits	Failed
Minimum + 1	11 digits	Passed
Maximum	12 digits	Passed
Maximum - 1	11 digits	Passed
Maximum + 1	13 digits	Failed

E.C.P. (Type)

Valid	Invalid
Numbers	Alphabets
	Special Characters
	Blank Field

Test Scenario 4Check / Verify / Validate **Address** Field**B.V.A. (Size)**

Minimum	1 Character	Passed
Minimum - 1	0 Character/ Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Numbers	Alphabets
	Special Characters
	Blank Field

Test Scenario 5Check / Verify / Validate **Policy Creation Operation** by Clicking **Submit** Button**D.T.**

Input Fields	Expected Output after Clicking Submit Button
All are Valid	Policy Number Returned
Any One Invalid	Error Message
Any One Blank Field	Error Message

Test Scenario 6Check / Verify / Validate **Policy Number** after **Successful Policy Creation****B.V.A. (Size)**

Minimum = Maximum	6 Digits	Passed
Minimum - 1	5 Digits	Failed
Minimum + 1	7 Digits	Failed

E.C.P. (Type)

Valid	Invalid
Numbers	Alphabets
	Special Characters
	Blank Field

Functional Specification 3

In a Library Management Software, Employees are Registering themselves to get **User id** and **Password**.

During Registration Process, Employee can fill the fields mentioned below

Username : Alphanumeric from 4 to 16 Characters long.

Password : Alphanumeric from 4 to 8 Characters long.

Confirm Password : Equal to given Password.

After filling the above 3 fields, employee can click **Register** Button to get **Employee User id** and **Password** for future **Login**.

Here, **Usernames** or **User Id's** must be unique.

Prepare **Test Scenario's** for **Employee Registration** Module **Functional Testing**.

Test Scenario 1

Check / Verify / Validate **Username** Field

B.V.A. (Size)

Minimum	4 Characters	Passed
Minimum - 1	3 Characters	Failed
Minimum + 1	5 Characters	Passed
Maximum	16 Characters	Passed
Maximum - 1	15 Characters	Passed
Maximum + 1	17 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Alphanumeric	Special Characters
	Blank Field

Test Scenario 2Check / Verify / Validate **Password** Field**B.V.A. (Size)**

Minimum	4 Characters	Passed
Minimum - 1	3 Characters	Failed
Minimum + 1	5 Characters	Passed
Maximum	8 Characters	Passed
Maximum - 1	7 Characters	Passed
Maximum + 1	9 Characters	Failed

E.C.P. (Type)

Valid	Invalid
Alphanumeric	Special Characters
	Blank Field

Test Scenario 3Check / Verify / Validate **Confirm Password** Field**D.T.**

Confirm Password Field Value	Criteria
Equal to Password Field Value	Passed
Other Value	Failed

Here, **Confirm Password** field is depending on already given **password** field (already stored in backend)

Test Scenario 4Check **Employee Registration Operation** by Clicking **Register** Button**D.T.**

Input Fields	Expected Output after Clicking Register Button
Username is Valid and Unique Password is Valid Confirm Password is equal to Password	Registered Successfully
Username is Valid and not Unique Password is Valid Confirm Password is equal to Password	Username already Exists (Prompt Message)
Username is Valid and Unique Password is Valid Confirm Password is not equal to Password	Error Message
Any One Invalid Field	Error Message
Any One Blank Field	Error Message

Regular Expressions

while preparing Test Scenario's and Test Cases, Testers are using Regular Expressions to simplify Valid and Invalid Data Types of Fields.

Regular Expressions is a Universal Mathematical Concept. Regular Expressions is Optional for Manual Testing, But Mandatory for Automation Testing.

To write Regular Expressions, we can follow the Notations / Patterns as shown below.

One Pair of Square Brackets [] indicates One Position

One Pair of Flower Brackets / Curly Brackets specify the Size / Range of Character

[]	One Digit or One Character (One Position)
[0 - 9]	One Digit
[a - z]	One Lower Case Alphabet
[A - Z]	One Upper Case Alphabet
[0 - 9 a - z]	One Digit or One Lower Case Character
[0 - 9 A - Z]	One Digit or One Upper Case Character
[0 - 9 A - Z a - z]	One Digit or One Alphabet
[0 - 9 A - Z a - z _]	One Digit or One Alphabet or one Underscore
[R A M]	R or A or M
[R][A][M]	R A M
[A - Z][a - z]	One Upper Case Character and One Lower Case Character
[A - Z][a - z 0 - 9]	One Upper Case Character and One Lower Case Character or Digit
[A - Z][a - z][a - z]	One Upper Case and Next Two Lower Case Characters
[A - Z][a - z]{ 2 }	One Upper Case and Next Two Lower Case Characters

[A - Z] { 2 } [a - z] { 4 }	
First Two Characters Upper Case and Next 4 Characters Lower Case	
[a - z] { 4 , 16 }	
Alphabets in Lower Case from 4 Characters to 16 Characters Long	
[A - Z] [a - z] { 3 , 10 }	
Start with Upper Case Alphabet and remaining 3 to 10 Characters Lower Case Alphabets	
[A - Z a - z] [a - z] { 3 , 10 }	
Start with Upper or Lower Case Alphabet and remaining 3 to 10 Characters in Lower Case	
[A - Z] [a - z 0 - 9 _] { 3 , 5 }	
One Upper Case Character and remaining 3 to 5 Characters in Lower Case Alphabets or Numbers or Underscore	
[0 - 9] { 4 , }	4 Digits to Infinite Number of Digits
[0 - 9] { 1 , }	[0 - 9] +
One Digit to Infinite Number of Digits	
[0 - 9] { , 5 }	No Digit to 5 Digits Numbers
[0 - 9] { 0 , }	[0 - 9] *
No Digit to Infinite Number of Digits	
[0 - 9] { 0 , 1 }	[0 - 9] ?
No Digit to One Digit	
[0 - 9] { }	[0 - 9]
One Digit or One Position	

[a - z 7 - 8]	One Lower Case Alphabet or 7 or 8
[a - z 7 8 -]	One Lower Case Alphabet or 7 or 8 or -
[0 9 -]	0 or 9 or -
[0 - 9]	One Digit (0 or 1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9)

[a - z A - Z 0 - 9 _]	[.]
One Upper Case Alphabet or One Lower Case Alphabet or One Digit or Underscore	

[.] +	One or More (Infinite) Alphabets or Digits or Underscore
---------	--

Note :

Hyphen (-) when used between Two Numbers, it behaves as range. But when it is given after Numbers, behaves as itself i.e. Hyphen.

Any Special Characters need to be mentioned Individually as there is no short form to mention them in Regular Expressions

[.] *	No or More (Infinite) Alphabets or Digits or Underscore	
[.] ?	No or One Alphabets or Digits or Underscore	
[\ .]	One .	One Dot
[\\ .]	One \ or One .	One Backslash or One Dot
[. \]	One Alphabet or One Digit or One Underscore or One Backslash	
[\ s]	One Blank Space	

One Uppercase Alphabet, No or More (Infinite) Lower Case Alphabets, One Blank Space, One Upper Case Alphabet, No or More (Infinite) Lower Case Alphabets	
[A - Z] [a - z] * [\ s] [A - Z] [a - z]	([A - Z] [a - z] * [\ s] ?) { 2 }
Two Words starting with Upper Case Alphabet and remaining No or More Lower Case Alphabets, separated by a Blank Space	

*	+	?
No or More Positions	One or More Positions	No or One Position

(){}	[]*	()*
Correct	Correct	Incorrect

{,2}	{2}	{2,}
No or Two	Two	Two or More

([A-Z][a-z]*[\s]?){1,}	Universal Expression
One Sentence with One or More words, but every word should start with Upper Case Alphabet and remaining are lower Case Alphabets.	

([a-z]+[\s]?){1,}
One Sentence with One or More words and Every word in Lower Case.

[A-Z][a-z]+[\s]([a-z]+[\s]?){1,}[\.]	
First word First Position Upper Case, Remaining positions lower After first word, remaining all words in lower case Sentence ends with a Full stop	Example : The business man

[A-Z][a-z]+[\s]([a-z]+[\s]?){0,}[\.]	
One Sentence with One or More words, but First word should start with Upper Case Alphabet and remaining are Lower Case Alphabets. Total Sentence must end with a Full Stop.	

([\w][\s]?){1,}	One Word and One Space Until Infinite	
\w	Only for Alphabets	
.	For Alphabets, digits and Underscore	
[\s]{8}	One Tab	8 Blank Spaces

<code>[\ w]</code>	<code>([A - Z a - z] +)</code>
One or More Upper or Lower Case Alphabet	

<code>[\ w] +</code>	<code>([A - Z a - z] + [\ s] ?) { 1 , }</code>
Both the Expressions written above convey the same meaning	

<code>[^]</code>	<code>[^ a]</code>	<code>[^ 0 - 9]</code>
Word with any	All except a	All except 0 to 9

<code>[\ ^]</code>	<code>[^ a - z]</code>	<code>[^ A - Z]</code>
One Cap	All except a to z	All except A to Z

<code>[^]</code>	<code>[^ o 1]</code>
Not Any (or) All	Not 0 and 1

Functional Specification 4

After Successful **Login**, Library Employees can store Information of Books in to Library Management Software by filling the following Fields.

Book Title : one or more words in Upper Case only.

Book Author : One or More words, but every word starts with Uppercase Character and remaining Lower case.

Publisher : One or More words, but every word in Lower Case.

Number of Copies : Up to 10

After filling above fields, Library Employee can Click **Feed** Button to get **Book id** as **Output**.

Here, **Book id** is in the following Format : **B O O K - X X X X X** (a 5 digit Number after **B O O K -**)

Prepare **Test Scenario's** for **Book Feeding** Module **Functional Testing**.

Test Scenario 1

Check / Verify / Validate **Book Title** Field

B.V.A. (Size)

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
$([A-Z] + [\backslash s]?)\{1, \}$	$[a-z0-9]$
	Special Characters except $[\backslash s]$
	Blank Field

Test Scenario 2Check / Verify / Validate **Book Author** Field**B.V.A. (Size)**

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
([A - Z] [a - z] * [\ s] ?) { 1 , }	[0 - 9]
	Special Characters except [\ s]
	Blank Field

Test Scenario 3Check / Verify / Validate **Publisher** Field**B.V.A. (Size)**

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
([a - z] + [\ s] ?) { 1 , }	[0 - 9 A - Z]
	Special Characters except [\ s]
	Blank Field

Test Scenario 4Check / Verify / Validate **Number of Copies** Field**B.V.A. (Range)**

Minimum	1	Passed
Minimum - 1	0	Failed
Minimum + 1	2	Passed
Maximum	10	Passed
Maximum - 1	9	Passed
Maximum + 1	11	Failed

E.C.P. (Type)

Valid	Invalid
[0 - 9] + Generic Expression	[a - z A - Z]
Or	Special Characters
([1 - 9]) / ([1] [0]) Detailed Expression	Blank Field

Test Scenario 5Check / Verify / Validate **Book Feeding Operation** by Clicking **Feed** Button**D.T.**

Input Fields	Expected Output after Clicking Submit Button
All are Valid	Book id Returned
Any One Invalid	Error Message
Any One Blank Field	Error Message

Test Scenario 6Check / Verify / Validate **Book id** Format after Successful Data Feeding**B.V.A. (Size)**

Minimum = Maximum	10 Positions	Passed
Minimum - 1	9 Positions	Failed
Minimum + 1	11 Positions	Failed

E.C.P. (Type)

Valid	Invalid
[B][O] { 2 } [K] [-] [0 - 9] { 5 }	[a - z]
	[A - Z] except B O K
	Special Characters except -
	Blank Field

Functional Specification 5

After Successful Login, Corresponding Library Employee can Register Readers by filling the following fields.

Reader Name : Alphabets in Lower Case as one or more words, but only first word starts with Upper Case Alphabet.

Age : should be less than 60 years and greater than 12 years.

Telephone Number : Valid in India

Address : Alphanumeric along with , : / -

After Filing above fields, corresponding Library Employee can click **Register** Button.

Here, **Reader id** is coming as **Output** in the following Format.

mm - yy - x x x x x x

Prepare **Test Scenario's** for **Reader Registration Module Functional Testing**.

Test Scenario 1

Check / Verify / Validate **Reader Name** Field

B.V.A. (Size)

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
[A - Z] [a - z] * [\ s] ([a - z] + [\ s] ?) { 0 , }	[0 - 9]
	Special Characters except [\ s]
	Blank Field

Test Scenario 2Check / Verify / Validate **Age** Field**B.V.A. (Range)**

Minimum	13	Passed
Minimum - 1	12	Failed
Minimum + 1	14	Passed
Maximum	59	Passed
Maximum - 1	58	Passed
Maximum + 1	60	Failed

E.C.P. (Type)

Valid	Invalid
[0 - 9] +	[a - z A - Z]
	Special Characters
	Blank Field

Test Scenario 3Check / Verify / Validate **Telephone Number** Field**B.V.A. (Size)**

Minimum	10 digits	Passed
Minimum - 1	9 digits	Failed
Minimum + 1	11 digits	Passed
Maximum	12 digits	Passed
Maximum - 1	11 digits	Passed
Maximum + 1	13 digits	Failed

E.C.P. (Type)

Valid	Invalid
[0 - 9] +	[a - z A - Z]
	Special Characters
	Blank Field

Test Scenario 4Check / Verify / Validate **Address** Field**B.V.A. (Size)**

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	256 Characters	Passed
Maximum - 1	255 Characters	Passed
Maximum + 1	257 Characters	Failed

E.C.P. (Type)

Valid	Invalid
([A - Z a - z 0 - 9] + [\ s , : / -] ?) { 1 , }	Special Characters except [\ s] , / - :
	Blank Field

Test Scenario 5Check **Reader Registration Operation** by Clicking **Register** Button**D.T.**

Input Fields	Expected Output after Clicking Register Button
All are Valid	Reader id Returned
Any One Invalid	Error Message
Any One Blank Field	Error Message

Test Scenario 6Check / Verify / Validate **Reader id** Format after Successful Registration**B.V.A. (Size)**

Minimum = Maximum	12 Positions	Passed
Minimum - 1	11 Positions	Failed
Minimum + 1	13 Positions	Failed

E.C.P. (Type)

Valid	Invalid
([0][1-9])/([1][0-2])[-][0-9]{2}[-][0-9]{6}	[a - z A - Z]
	Special Characters except -
	Blank Field

Different Circumstances in Work Culture

Without Document	Partial Document	Complete Document
------------------	------------------	-------------------

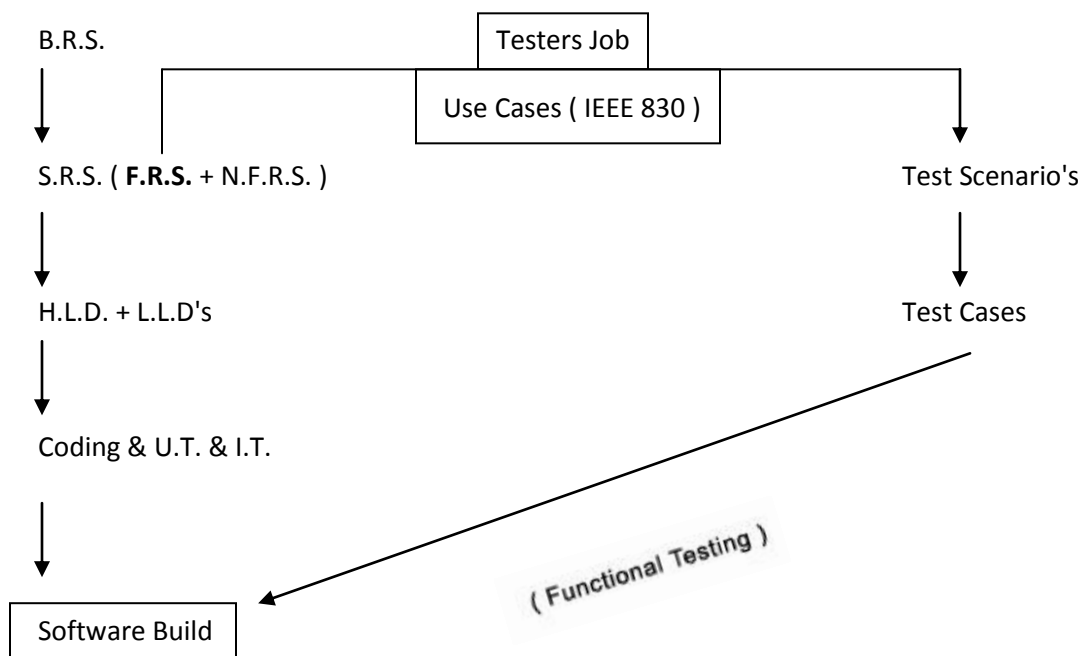
Two) Use Cases based Test Design.

Some Times, Test Engineers can prepare Test Scenario's and Cases for responsible Modules Functional Testing depending on Use Cases instead of Functional Specifications.

In General, Use Cases are more elaborative than F.R.S. in S.R.S.

These Use Cases are also prepared by S.A.

Testers are expecting Use Cases instead of F.R.S. to understand **Complex requirements** in **Project** or to understand **requirements** in **Outsourced Project**.



Use Case 1 (IEEE 830)

1. **Use Case id** : UC_BOOK_ISSUE

2. **Description** : Valid Readers can apply for available book in Library for issue.

3. **Actors** : Reader id is in mm - yy - x x x x x x and Book id is in BOOK - X X X X X

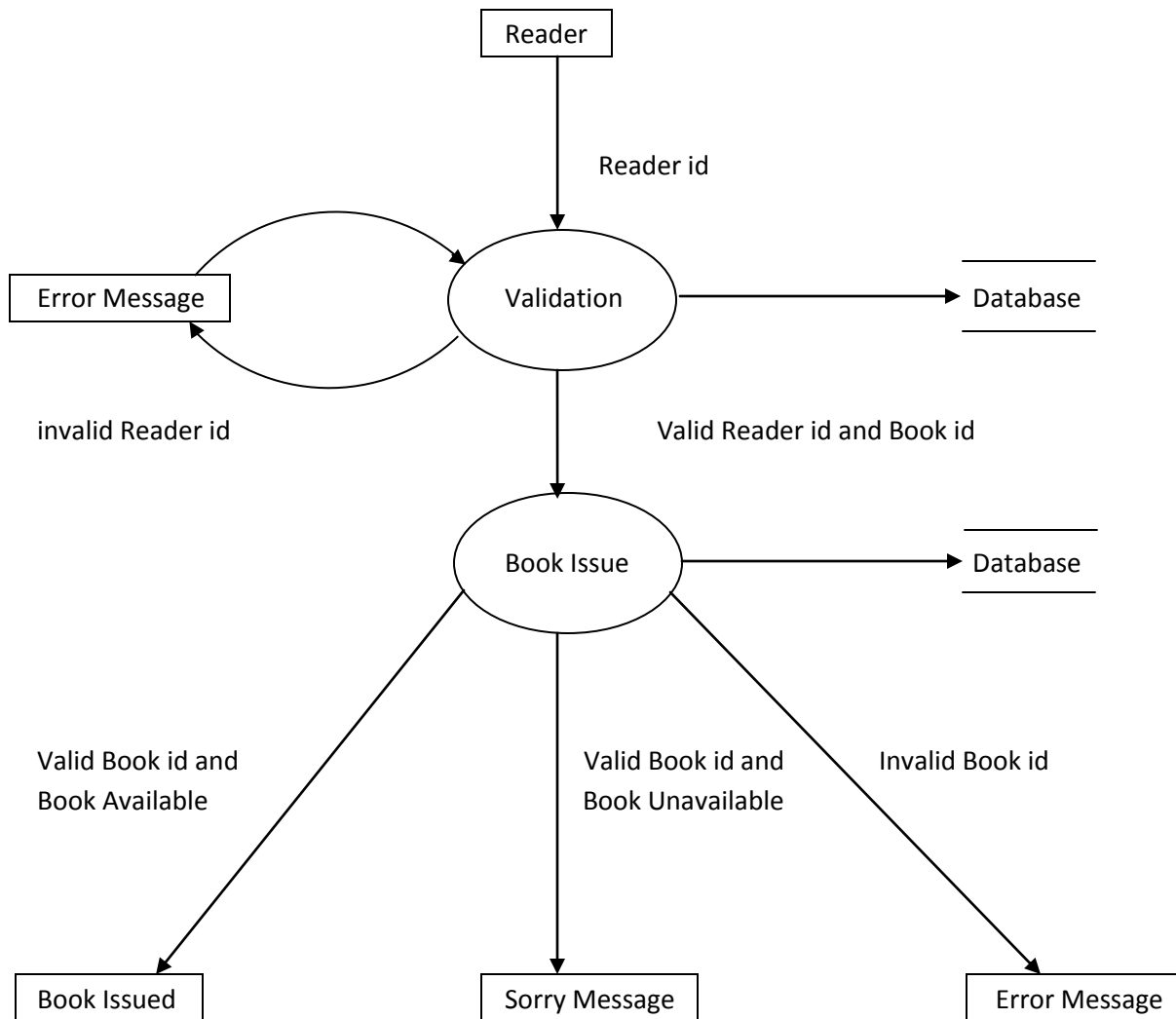
4. **Precondition** : Before Issue, Readers must be registered and books information must be feeded.

5. **Events List** :

Event	Output / Outcome
Enter Reader id and Click Go	If Reader id is valid, then Book id can focus
	If Reader id invalid, then Error Message Appears
Enter Book id and Click Go Button	If Book id is valid and Book is available, then Book Issued Message appears
	If Book id is valid and Book is unavailable, then Sorry, Book Issued to Others Message appears
	If Book id is invalid, then Error Message appears

P.T.O.

6. AFD : Activity Flow Diagram



7. Prototype or Screen Shot

BOOK ISSUE		- □ X
Reader id	<input type="text"/>	<input type="button" value="Go"/>
Bood id	<input type="text"/>	<input type="button" value="Go"/>

8. Post Conditions

Corresponding Library Employee can go to **Next Transaction** or can go to **logout**.

9. Alternative Events

No Alternate Events for Book Issue.

10. Related Use Cases

UC_BOOK_FEED

UC_Reader_Register

UC_Emp_Login

Uc_Emp_Logout

Prepare **Test Scenario's** for **Book Issue** Module **Functional Testing**.

Test Scenario 1	Check / Verify / Validate Reader Id Field
------------------------	--

B.V.A. (Size)

Minimum = Maximum	12 Positions	Passed
Minimum - 1	11 Positions	Failed
Minimum + 1	13 Positions	Failed

E.C.P. (Type)

Valid	Invalid
([0][1-9])/([1][0-2])[-][0-9]{2}[-][0-9]{6}	[a - z A - Z]
	Special Characters except -
	Blank Field

Test Scenario 2Check **Reader id** Validation by Clicking **Go** Button**D.T.**

Reader id Field	Expected Output after Clicking Go Button
Valid	Focus on Book id
Invalid	Error Message
Blank Field	Error Message

Test Scenario 3Check / Verify / Validate **Book Id** Field after **Reader id** Field Validation**B.V.A. (Size)**

Minimum = Maximum	10 Positions	Passed
Minimum - 1	9 Positions	Failed
Minimum + 1	11 Positions	Failed

E.C.P. (Type)

Valid	Invalid
[B][O] { 2 } [K] [-] [0 - 9] { 5 }	[a - z]
	[A - Z] except B O K
	Special Characters except -
	Blank Field

Test Scenario 4

Check Book Issue Operation by clicking **Go** Button after successful **Reader id** Validation

D.T.

Reader id	Book id	Expected Output after click Go Button
Valid	Valid and Copies Available	Book Issued
Valid	Valid, But Copies Unavailable	Sorry, Book already issued to Others
Valid	Invalid	Error Message
Valid	Blank Field	Error Message

in the above Decision Table, **Book id** depends on **Reader id**. That's why, **Reader id** column is also included along with **Book id**.

Test Scenario 5

Check / Verify / Validate **Minimize** Icon Functionality

D.T.

Fields	Expected Output after Clicking Minimize
All Filled	Window Minimized
Some Filled	Window Minimized
All Blank Fields	Window Minimized

Note : The window should be in **Active Mode** or **Maximized Mode** to Check whether **Minimize** Icon is working properly or not.

Test Scenario 6Check / Verify / Validate **Maximize** Icon Functionality**D.T.**

Fields	Expected Output after Clicking Maximize
All Filled	Window Maximized
Some Filled	Window Maximized
All Blank Fields	Window Maximized

Note : The window should be in **Inactive Mode** or **Minimized Mode** to Check whether **Maximize** Icon is working properly or not.

Test Scenario 7Check / Verify / Validate **Close** Icon Functionality**D.T.**

Fields	Expected Output after Clicking Close
All Filled	Window Closed
Some Filled	Window Closed
All Blank Fields	Window Closed

Note : The window should be in **Active Mode** or **Open Mode** to Check whether **Close** Icon is working properly or not.

Extra Notes

O : Operation

OUT : Operation under Testing.

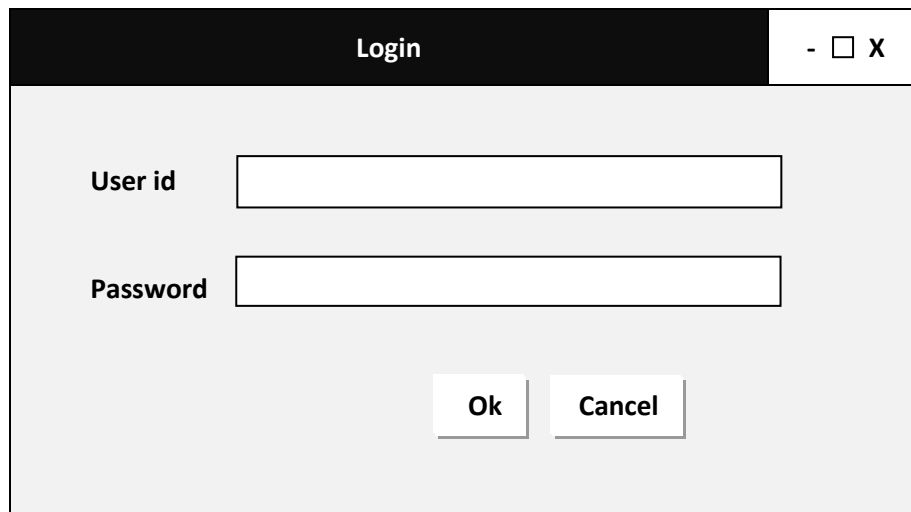
Dependency

The Fields in a Dialog Box can either be Independent or Dependent. Some Fields need to be filled in a Specific Order due to Dependency.

Example :

In the Dialog Box shown below, we can enter **User id** first or enter **Password** first, as they are two independent fields.

But **Ok** Button Operation depends on both the inputs to be physically present and valid to go to the next operation or Next Screen.



Note

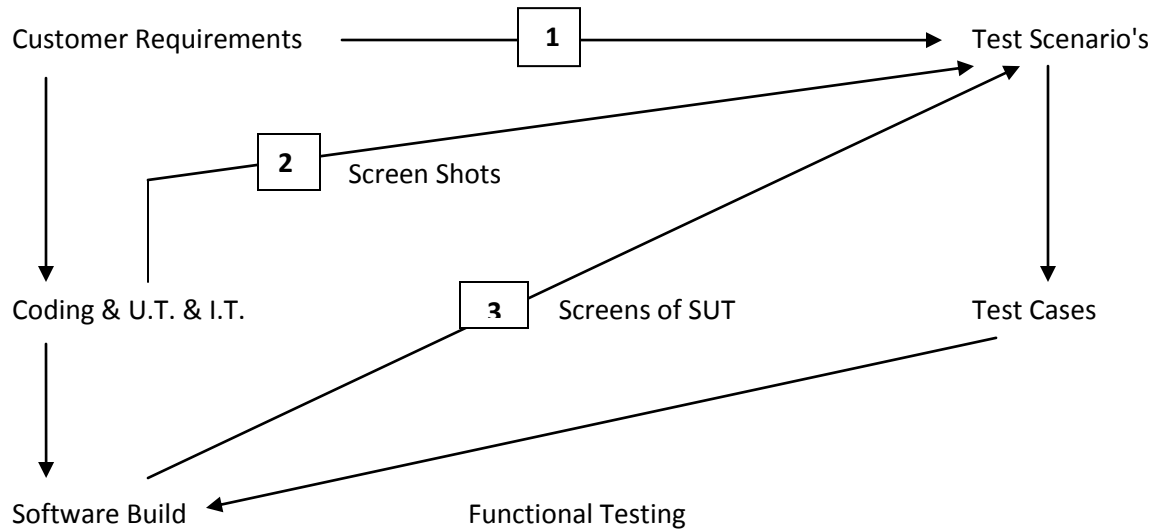
Black Box Techniques and Regular Expressions are useful to Testers while preparing Test Scenario's and Cases for Functional Testing Only.

In Regular Expressions, Pipe Symbol " | " is used for " or " and Cap Symbol " ^ " is used for " not " (Negation)

Functional Testing is Module Level where as Non Functional Testing is Software Level.

Three) Screens Based Test Design

Sometimes Due to unavailability of Use Cases and F.R.S. i.e. No Documentation at all, Testers are preparing Test Scenario's and Test Cases depending on Screen Shots or Screen Soft Software (Screens in Software)

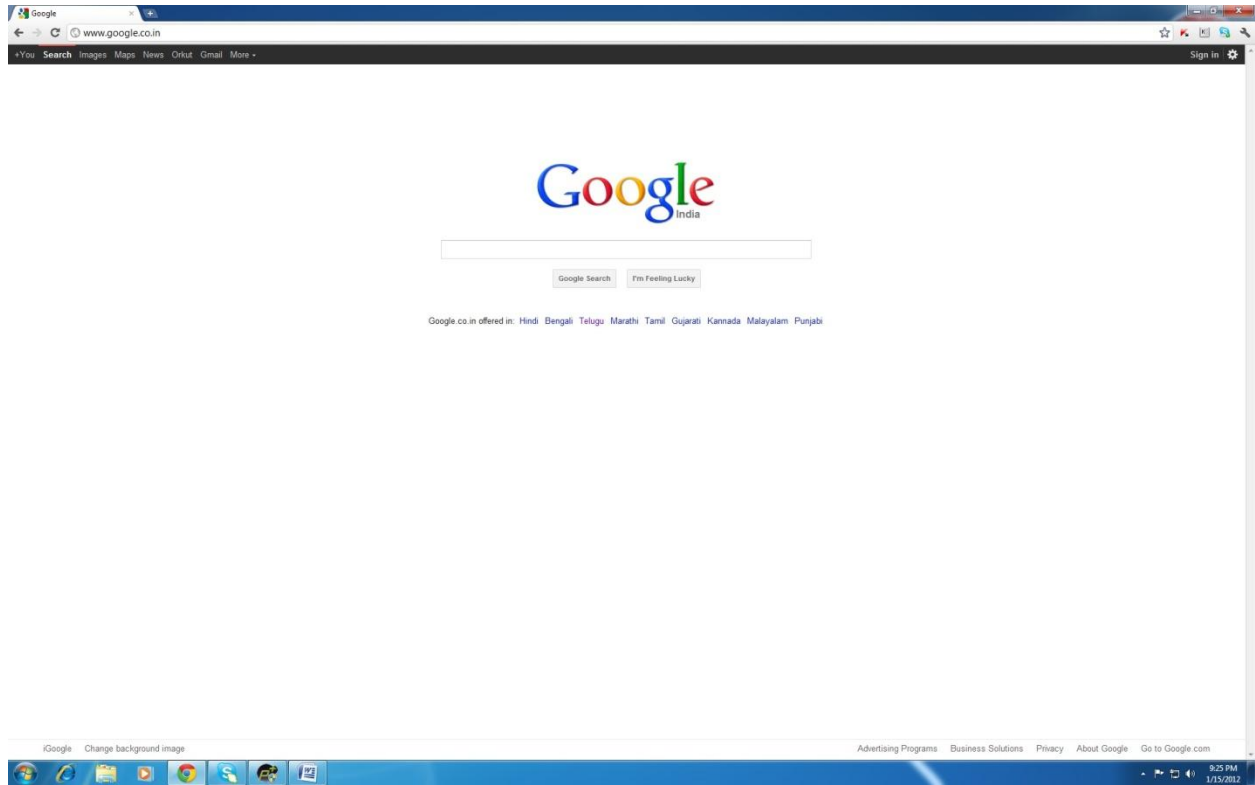


From the Above Diagram, Testers are understanding Project Requirements by

- [1] Talking to Customer Site People.
- [2] By seeing Prototype or Screen Shots.
- [3] By Operating Screens of SUT.

SCREEN 1

Google Search



Field Validations (followed by Developers)

Search String in **Google Search** is taking Alphanumeric including Special Characters from 1 to 2048 Characters as one or more words.

Prepare Test Scenario's for **Google Search** Functional Testing.

Test Scenario 1Check / Verify / Validate **Search** String Field**B.V.A. (Size)**

Minimum	1 Characters	Passed
Minimum - 1	0 Character / Blank Field	Failed
Minimum + 1	2 Characters	Passed
Maximum	2048 Characters	Passed
Maximum - 1	2047 Characters	Passed
Maximum + 1	2049 Characters	Failed

E.C.P. (Type)

Valid	Invalid
([^] + [\ s] ?) { 1 , }	Blank Field
or	
[A - Z a - z 0 - 9 _ - + \ . ? . . .] + [\ s] ? { 1 , }	

Test Scenario 2Check / Verify / Validate **Google Search** Operation by Clicking **Search** Button**D.T.**

Search String	Expected Output after Clicking Search
Valid String	Addresses of Websites related to Searched String (List of URL's)
Invalid String (non meaning ful)	List of URL's (Uniform Resource Locator) which are unrelated
Blank Field	Stay in Same Page

SCREEN 2

The image shows a standard Windows-style dialog box titled "Login". The title bar is dark with the text "Login" in white, and it includes standard window controls (minimize, maximize, close) on the right. The main area of the dialog is light gray. It contains two text input fields. The first is labeled "Agent Name" and the second is labeled "Password". Below the input fields are two buttons: "Ok" and "Cancel".

Field Validations

Agent Name is taking Alphanumeric and Special Characters as Single Word from 4 to 10 Characters long.

Password is taking Alphanumeric as Single Word from 4 to 8 Characters long.

Prepare Test Scenario's for **Login** Module **Functional Testing**.

Test Scenario 1Check / Verify / Validate **Agent Name** Field**B.V.A. (Size)**

Minimum	4 Positions	Passed
Minimum - 1	3 Positions	Failed
Minimum + 1	5 Positions	Passed
Maximum	10 Positions	Passed
Maximum - 1	9 Positions	Passed
Maximum + 1	11 Positions	Failed

E.C.P. (Type)

Valid	Invalid
[^ \ s] +	[\ s]
	Blank Field

Test Scenario 2Check / Verify / Validate **Password** Field**B.V.A. (Size)**

Minimum	4 Positions	Passed
Minimum - 1	3 Positions	Failed
Minimum + 1	5 Positions	Passed
Maximum	8 Positions	Passed
Maximum - 1	7 Positions	Passed
Maximum + 1	9 Positions	Failed

E.C.P. (Type)

Valid	Invalid
[A - Z a - z 0 - 9] +	Special Characters
	Blank Field

Test Scenario 3Check / Verify / Validate **Login** Operation by Clicking **Ok** Button**D.T.**

Fields	Expected Output after Clicking Ok
All are Valid	Next Window
Any One Invalid	Error Message
Any One Blank Field	Error Message

Test Scenario 4Check / Verify / Validate **Cancel** Operation by Clicking **Cancel** Button**D.T.**

Fields	Expected Output after Clicking Cancel
All are Filled	Window Closed
Some Fields are Filled	Window Closed
All Fields are Empty	Window Closed

SCREEN 3

Book Return		- <input type="checkbox"/> X
Reader id	<input type="text"/>	
<input type="button" value="Search"/>		
Book id	<input type="text"/>	Rent <input type="text"/>

One Module	One Input	Two Outputs		One Operation
Book Return	Reader id	Book id	Rent	Search

Field Validations

Reader id is in **mm - yy - x x x x x x**

Book id and **Rent** depends on issued Book to given Reader

Here, Corresponding **Book id** is in **BOOK - X X X X X** and **Rent** is **10** Rupees.

Prepare **Test Scenario's** for **Book Return** Module **Functional Testing**.

Test Scenario 1Check / Verify / Validate **Reader Id** Field**B.V.A. (Size)**

Minimum = Maximum	12 Positions	Passed
Minimum - 1	11 Positions	Failed
Minimum + 1	13 Positions	Failed

E.C.P. (Type)

Valid	Invalid
([0][1-9])/([1][0-2])[-][0-9]{2}[-][0-9]{6}	[a - z A - Z]
	Special Characters except -
	Blank Field

Test Scenario 2Check **Reader id** Validation by Clicking **Search** Button**D.T.**

Reader id Field	Expected Output after Clicking Search Button
Valid Reader id who already got one Book	Corresponding Book id and Rent Value
Valid Reader id, but did not get any Book in past	Book id and Rent as Blank Fields
Invalid Reader id	Error Message
Blank Field	Error Message

Test Scenario 3

Check / Verify / Validate **Book Id** Format which came for **Valid Reader id** who got already one Book in Past

B.V.A. (Size)

Minimum = Maximum	10 Positions	Passed
Minimum - 1	9 Positions	Failed
Minimum + 1	11 Positions	Failed

E.C.P. (Type)

Valid	Invalid
[B][O] { 2 } [K] [-] [0 - 9] { 5 }	[a - z]
Or	[^ B O K]
[B][O] { 2 } [K] [-] [0 - 9] +	Special Characters except -
	Blank Field

Test Scenario 4

Check **Rent** that came for **Valid Reader id** who already got one Book in past

B.V.A. (Range)

Minimum = Maximum	10	Passed
Minimum - 1	9	Failed
Minimum + 1	11	Failed

E.C.P. (Type)

Valid	Invalid
[1][0]	[a - z A - Z]
	[0 - 9] except 0 1
	Special Characters
	Blank Field

Test Scenario 5Check / Verify / Validate **Minimize** Icon Functionality**D.T.**

Fields	Expected Output after Clicking Minimize
All Filled	Window Minimized
Some Filled	Window Minimized
All Blank Fields	Window Minimized

Note : The window should be in **Active Mode** or **Maximized Mode** to Check whether **Minimize** Icon is working properly or not.

Test Scenario 6Check / Verify / Validate **Maximize** Icon Functionality**D.T.**

Fields	Expected Output after Clicking Maximize
All Filled	Window Maximized
Some Filled	Window Maximized
All Blank Fields	Window Maximized

Note : The window should be in **Inactive Mode** or **Minimized Mode** to Check whether **Maximize** Icon is working properly or not.

Test Scenario 7Check / Verify / Validate **Close** Icon Functionality**D.T.**

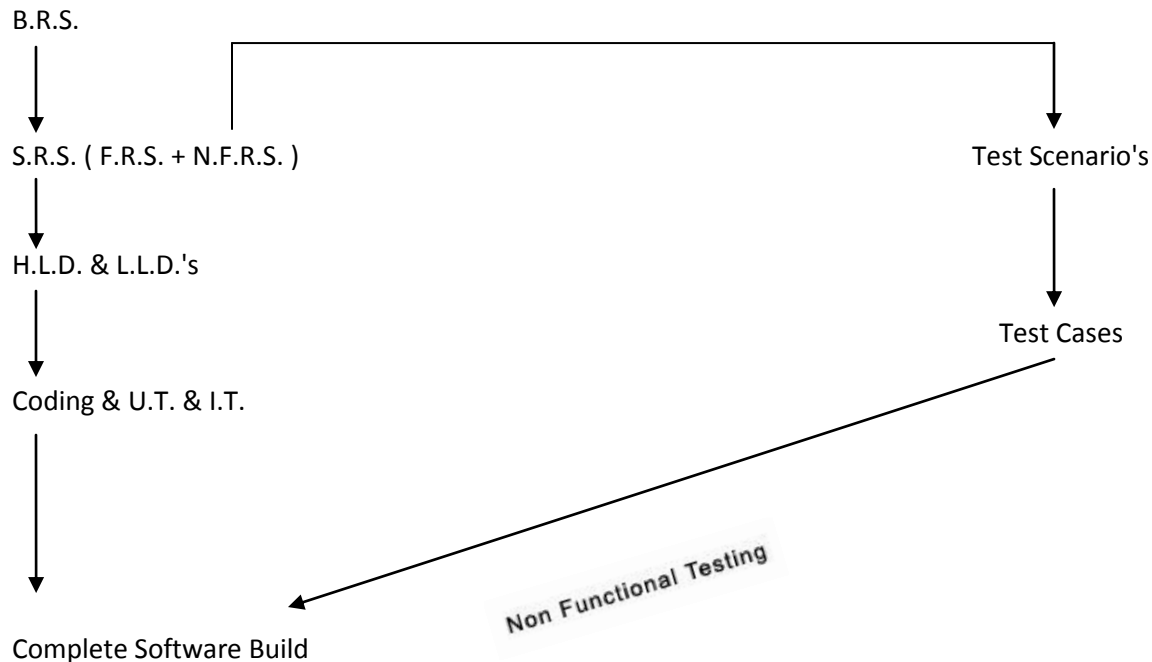
Fields	Expected Output after Clicking Close
All Filled	Window Closed
Some Filled	Window Closed
All Blank Fields	Window Closed

Note : The window should be in **Active Mode** or **Open Mode** to Check whether **Close** Icon is working properly or not.

4. Non Functional Specifications based Test Design

After Completion of writing Test Scenario's and Cases for responsible Module Functional Testing depending on Functional Specifications or Use Cases or Screens, Corresponding Testers can concentrate on Test Scenario's and Cases Writing for Complete Software Non Functional Testing.

Due to this reason, Non Functional Testing is called as System Testing.



without Documents, Functional Testing is Possible. But without Documentation, Non Functional Testing Design is impossible.

From the above Diagram,

* Non Functional Testing Design depends on N.F.R.S. in S.R.S.

* Non Functional Testing Execution is possible on a Complete Software.

Black Box Testing Techniques are not useful in Non Functional Testing Design, because, Non Functional Testing Topics are concentrating on characteristics of Software, but not on the functionality of the Software.

Non Functional Specification 1 (Usability Requirement)

From Customer Requirements and Expectations, Corresponding Library Management Software is **User Friendly** to Corresponding Library Employees to use.

Prepare Test Scenario's for Library Management Software Usability Testing.

Test Scenario 1

Check Spellings in all the Screens of Software.

Test Scenario 2

Check Meaning of Labels in all the Screens of Software.

Test Scenario 3

Check init cap (1st alphabet of the word as Capital) of all Labels in all the Screens of Software.

Test Scenario 4

Check Labels Font Size in all the Screens of Software.

Test Scenario 5

Check Font Style of Labels in all the Screens of Software.

Test Scenario 6

Check Labels Font Colors in all Screens of Software.

Test Scenario 7

Check Line Spacing in between Labels and Objects in all the Screens of Software.

Test Scenario 8

Check Line Spacing in between Objects in all the Screens of Software.

Uniformity (Yes / No)

Test Scenario 9

Check Alignment of Objects in all the Screens of Software.

(Left - Right - Top - Bottom)

Test Scenario 10

Check Functional Grouping of Related Objects in Screens.

(Framing)

Test Scenario 11

Check Borders of Functionally related Objects in all the Screens of Software.

Test Scenario 12

Check Icons in Screens with respect to providing Functionality.

Test Scenario 13

Check Tool Tips of Icons with respect to providing Functionality.

Test Scenario 14

Check Keyboard Access on all the Screens of Software.

Test Scenario 15

Check Date Formats in all the Screens of Software.

Test Scenario 16

Check Shortcuts for well known abbreviations in all the Screens of Software.

Test Scenario 17

Check System Menu existence in every Screen of Software.

(Minimize, Maximize, Restore, Close, . . .)

Test Scenario 18

Check for existence of Buttons like **OK** and **Cancel** in Every Screen to Continue or to Stop.

Test Scenario 19

Check Meaning of Error Messages in all the Screens of Software.

Test Scenario 20

Check HELP.

Note :

1. Above Mentioned Scenario's are applicable on any Software Usability Testing.
2. Test Scenario 20 is used for HELP Documents Testing or User Manuals Testing.

Non Functional Specification 2 (Compatibility Requirement)

From Customer Requirements and Expectations, Library Management Software will be run on the Platforms mentioned Below.

Windows 2003 Server	Server Side Operating System
Windows 2008 Server	
Windows XP	Client Side Operating System
Windows Vista	
Windows 7	

This Library Management Software consists of the following Functionalities / Modules :

1. Employee Registration
2. Employee Login
3. Reader Registration
4. Books Feeding
5. Books Issue
6. Book Return
7. Rent Payment
8. Employee Logout

Check Whether these 8 Modules are working Correctly on all the 5 Platforms or not.

Prepare Test Scenario's for Library Management Software Compatibility Testing.

Note

In the above details, Client does not mean Customer, but it is a word from Computer Hardware Terminology.

Test Scenario 1	Check Employee Registration Functionality in the Platforms specified below	
Availability Matrix / Compatibility Matrix		
Platform Component Type	Version	Yes / No
Server side Operating System	Windows 2003	Yes
	Windows 2008	Yes
	Others	Yes / No
Client Side Operating System	Windows XP	Yes
	Windows Vista	Yes
	Windows 7	Yes
	Others	Yes / No

Test Scenario 2	Check Employee Login Functionality in the Platforms specified below
------------------------	--

Test Scenario 3	Check Reader Registration Functionality in the Platforms specified below
------------------------	---

Test Scenario 4	Check Books Feeding Functionality in the Platforms specified below
------------------------	---

Test Scenario 5	Check Books Issue Functionality in the Platforms specified below
------------------------	---

Test Scenario 6	Check Book Return Functionality in the Platforms specified below
------------------------	---

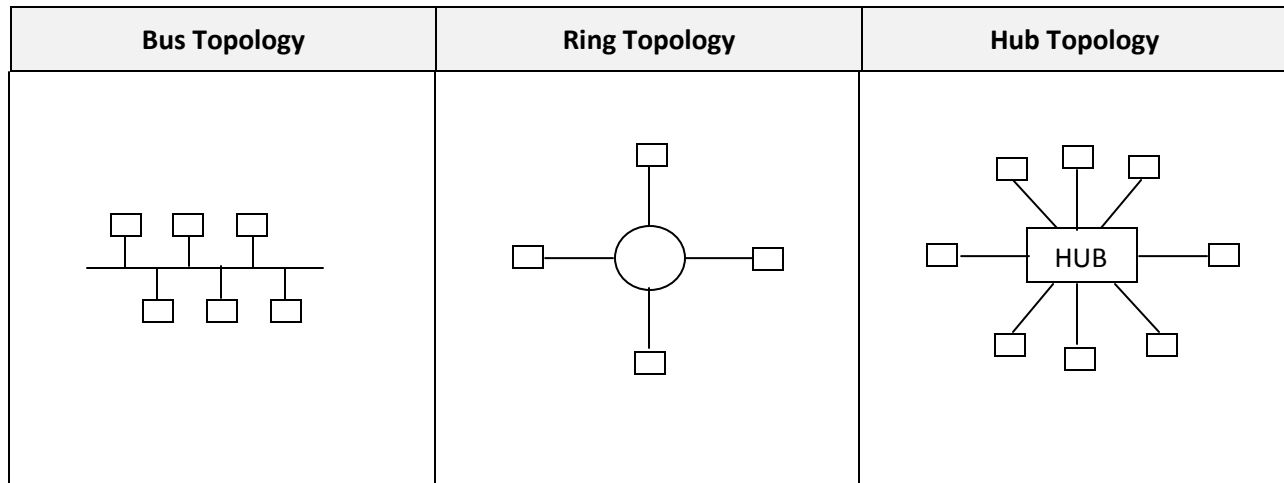
Test Scenario 7	Check Rent Payment Functionality in the Platforms specified below
------------------------	--

Test Scenario 8	Check Employee Logout Functionality in the Platforms specified below
------------------------	---

Test Scenario 2	Test Scenario 3	Test Scenario 4	Test Scenario 5	Test Scenario 6	Test Scenario 7	Test Scenario 8
For all the above Scenario's, Availability Matrix / Compatibility Matrix is same as Test Scenario 1						

Non Functional Specification 3 (Configuration Requirements) (Hardware Compatibility Testing)

From Customer Requirements and Expectations, Corresponding Library Management Software will run on different types of Networks like Bus, Ring & Hub Topologies.



In this Library Management Project, Rent Payment Module can provide a Receipt to Reader.

To make this Receipt, our Software can use different types of Printers like Inkjet, Dot Matrix and Laser.

Prepare **Test Scenario's** for **Library Management Software Configuration Testing**.

Test Scenario 1	Check Employee Registration Functionality in the Hardware Environment specified below.	
Hardware Configuration Matrix		
Hardware Component	Version / Type	Yes / No
Network	Bus	Yes
	Ring	Yes
	Hub	Yes
	Others	Yes / No

Test Scenario 2	Check Employee Login Functionality in the Hardware Environment, specified below.
-----------------	---

Test Scenario 3	Check Reader Registration Functionality in the Hardware Environment, specified below.
-----------------	--

Test Scenario 4	Check Books Feeding Functionality in the Hardware Environment, specified below.
------------------------	--

Test Scenario 5	Check Books Issue Functionality in the Hardware Environment, specified below.
------------------------	--

Test Scenario 6	Check Book Return Functionality in the Hardware Environment, specified below.
------------------------	--

Test Scenario 8	Check Employee Logout Functionality in the Hardware Environment, specified below.
------------------------	--

Test Scenario 2	Test Scenario 3	Test Scenario 4	Test Scenario 5	Test Scenario 6	Test Scenario 8
For all the above Scenario's, Hardware Configuration Matrix is same as Test Scenario 1					

Test Scenario 7	Check Rent Payment Functionality in the Hardware Environment, specified below.	
Hardware Configuration Matrix		
Hardware Component	Version / Type	Yes / No
Network	Bus	Yes
	Ring	Yes
	Hub	Yes
	Others	Yes / No
Printer	Inkjet	Yes
	Dot Matrix	Yes
	Laser	Yes
	Others	Yes / No

Non Functional Specification 4 (Inter Systems Testing / SOA Testing)

From Customer Requirements and Expectations, Rent Payment Module in Library Management Software can allow VISA, MASTER and AMERICAN EXPRESS Credit Cards.

Test Scenario 1

Check **Rent Payment** Functionality by connecting to External Servers, specified below

External Server Connectivity Name	Yes / No
VISA	Yes
MASTER	Yes
AMERICAN EXPRESS	Yes
Others	No

Non Functional Specification 5 (Performance Requirements)

From Customer Requirements and Expectations, corresponding Library Management Software will be used by 10 Employees of a Library at a time (Concurrently)

Prepare **Test Scenario's** for **Library Management Software Performance Testing**.

Test Scenario 1	Check Employee Registration Functionality under 10 Users Load	Load Testing
Test Scenario 2	Check Employee Registration Functionality under more than 10 Users Load by incrementing intervally.	Stress Testing
Test Scenario 3	Check Employee Registration Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 4	Check Employee Registration Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing
Test Scenario 5	Check Employee Login Functionality under 10 Users Load	Load Testing
Test Scenario 6	Check Employee Login Functionality under more than 10 Users Load by incrementing intervally.	Stress Testing
Test Scenario 7	Check Employee Login Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 8	Check Employee Login Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing
Test Scenario 9	Check Reader Registration Functionality under 10 Users Load	Load Testing
Test Scenario 10	Check Reader Registration Functionality under more than 10 Users Load by incrementing intervally.	Stress Testing
Test Scenario 11	Check Reader Registration Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 12	Check Reader Registration Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing

Test Scenario 13	Check Books Feeding Functionality under 10 Users Load	Load Testing
Test Scenario 14	Check Books Feeding Functionality under more than 10 Users Load by incrementing interval.	Stress Testing
Test Scenario 15	Check Books Feeding Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 16	Check Books Feeding Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing
Test Scenario 17	Check Book Issue Functionality under 10 Users Load	Load Testing
Test Scenario 18	Check Book Issue Functionality under more than 10 Users Load by incrementing interval.	Stress Testing
Test Scenario 19	Check Book Issue Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 20	Check Book Issue Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing
Test Scenario 21	Check Book Return Functionality under 10 Users Load	Load Testing
Test Scenario 22	Check Book Return Functionality under more than 10 Users Load by incrementing interval.	Stress Testing
Test Scenario 23	Check Book Return Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 24	Check Book Return Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing

P.T.O.

Test Scenario 25	Check Rent Payment Functionality under 10 Users Load	Load Testing
Test Scenario 26	Check Rent Payment Functionality under more than 10 Users Load by incrementing intervally.	Stress Testing
Test Scenario 27	Check Rent Payment Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 28	Check Rent Payment Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing
Test Scenario 29	Check Employee Logout Functionality under 10 Users Load	Load Testing
Test Scenario 30	Check Employee Logout Functionality under more than 10 Users Load by incrementing intervally.	Stress Testing
Test Scenario 31	Check Employee Logout Functionality under more than 10 Users Load as huge (at a time / Suddenly)	Spike Testing
Test Scenario 32	Check Employee Logout Functionality under 10 Users Load Continuously or iteratively (repeatedly)	Longevity Testing / Endurance Testing

Non Functional Specification 6 (Data Volume Requirements)

Data Capacity	Database Capacity
---------------	-------------------

From Customer Requirements and Expectations, Corresponding Library Management Software can allow Library Employees to store 25 Employees Information, 10000 Readers Information, 1 Lakh Books Information and 1 Crore Transactions on Book Issue, Book Return and Rent Payment.

prepare **Test Scenario's** for **Library Management** Software **Data Volume Testing**.

Test Scenario 1	Check Employee Registration Functionality to register 25 Employees as Maximum
Test Scenario 2	Check Reader's Registration Functionality to register 10,000 Readers as Maximum
Test Scenario 3	Check Book Feeding Functionality to feed 1 Lakh Books Information as Maximum
Test Scenario 4	Check Book Issue, Book Return and Rent Payment Functionalities to generate 1 Crore Transactions as Maximum

Extra Notes

Every Scenario has 2 Targets	
Which Module	Which Testing Topic
One Scenario has Many Cases	
Positive and Negative	Valid and Invalid

Non Functional Specification 7 (Installation Requirements)

From Customer Requirements and Expectations, Corresponding Library Management Software is Easy to Install and Uninstall in Customer Expected Configured Network.

Test Scenario 1	Check Setup program execution to Start Installation
------------------------	---

Test Scenario 2	Check Ease of Use in Screens during Installation
------------------------	---

Test Scenario 2	Check Occupied Disk Space after Installation
------------------------	---

Test Scenario 2	Check Un-installation Completely
------------------------	---

NOTE

1. **Multi Languity Testing** is needed to conduct, only when the **corresponding Software** is **supporting various people languages characters** as **inputs** and **outputs**.
2. **Security Testing** is **Complex to conduct** because, Testers need to have **knowledge of Hacking** to take up this task.
3. **Parallel Testing** means that the **comparison of Software Products**. So, **Parallel Testing** is of **no need** in **Projects**.

C) Low Level Test Design (L.L.T.D.)

After completion of Test Scenario's writing, corresponding Testers are implementing these Scenario's as Detailed Test Cases Documents in IEEE 829 Format.

Test Cases Document

In General, Testers are writing all Test Cases related to One Scenario in One Document, Called as Test Cases Document.

Test Cases Document in IEEE 829 Format

1. Test Cases Document id

A Unique Number for Future Reference

2. Test Cases Document Name

Corresponding Test Scenario

3. Feature / Module to be Tested

Corresponding Module Name in Software for which Module Testing, this Document was Prepared

Example :

Scenario	Check User id Field
Module is Login	Test is Functional Testing

4. Testing Topic

Corresponding Testing Topic Name in Software Testing for which this Document was prepared.

Examples :

Check Login Operation in Windows XP (Platform)	
Module is Login	Test is Compatibility Testing

Check Login Operation under 10 Users Load	
Module is Login	Test is Performance Testing

5. Test Environment

Required Hardware and Software for this Test Cases Document to run on SUT.
(Required Resources)

6. Test Suite id

Name of the Test Batch, in which, this Cases Document is Member

Example

Different Testing Topics related to One Module is called as One Batch

7. Priority

The Importance of Test Cases Document to run on SUT.

(This Topic is useful to Testers, when they are facing Lack of Time)

Example (General)

Priority		
High	Medium	Low
Functional Test Cases Documents	Non Functional Test Cases Documents except Usability Tests	Usability Tests

Example (Terminology followed in HCL Company)

Priority		
High	Medium	Low
P0	P1	P2

8. Test Effort

Expected Time to apply Corresponding Test Cases on SUT.

(Average time is 20 Minutes) (ISO 9001, Six Sigma Standards)

9. Test Setup

Necessary Tasks to do before starting Corresponding Test Cases Execution on SUT.

10. Test Procedure

A Step by Step procedure to list out all Positive and Negative Test Cases related to Corresponding Test Scenario.

Step No.	Step Description	Test Data	Expected Result	Actual Result / Output	Step Result	Defect id	Comments
Test Design				Test Execution			

11. Test Result

The Final Result of Test after Execution.

Final Result is Passeded, when all Steps Results are Passed.

Final Result is Failed, when any one Step Result was Failed.

Extra Notes

* In One Working Day, One Tester can prepare 30 Test Cases Documents on an Average.

* In Test Execution, One Tester can take 20 Minutes as Average time to Finish.

* So, One Tester can Execute 20 Test Cases Documents on SUT in One Working Day.

* One Test : All Cases Positive and Negative related to that Test.

* Internet : Network of Networks (All can Use)

* Intranet : Only for Specific Permitted Users

Functional Specification 1

In an Offline Banking Software, **Bank Employees** can do **Login** in corresponding Branch.

During this **Login**, **Bank Employee** can enter **Employee id** and **Password**.

Employee id is a **6** Digit Number.

Password is allowing **Alphanumeric** from **4** to **8** Characters Long.

After entering **Employee id** and **Password**, corresponding **Bank Employee** can Click **Ok** Button.

Some Times, they can use **Cancel** Button to **Close Login Screen**.

Prepare **Test Scenario's** and **Test Cases Documents** for **Login Module Functional Testing**.

Test Scenario's

TS1	Check Employee id Field.
TS2	Check Password Field.
TS3	Check Login Operation by Clicking Ok Button.
TS4	Check Login Closing by Clicking Cancel Button.

Test Cases Documents

Test Cases Document 1

1. Test Cases Doc id	TCD_ABC_10th Jan_1
----------------------	--------------------

2. Test Cases Doc Name	Check Employee id Field
------------------------	--------------------------------

3. Test Suite id	TS_Login_FT_PO_1
------------------	------------------

4. Test Setup	Login Screen Opened and have Valid and Invalid Employee id's in Hand
---------------	---

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Activate Login Window Fill Employee id	6 Digits	Accepted
		5 Digits	Rejected
		7 Digits	Rejected
		[0 - 9] +	Accepted
		[a - z A - Z]	Rejected
		Special Characters	Rejected
		Blank Field	Rejected

Total Number of Cases	7	Positive	2	Negative	5
-----------------------	---	----------	---	----------	---

Test Cases Document 2

1. Test Cases Doc id	TCD_ABC_10th Jan_2
-----------------------------	--------------------

2. Test Cases Doc Name	Check Password Field
-------------------------------	-----------------------------

3. Test Suite id	TS_Login_FT_PO_2
-------------------------	------------------

4. Test Setup	Login Screen Opened and have Valid and Invalid Passwords in Hand
----------------------	---

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Activate Login Window Fill Password	4 Positions	Accepted
		3 Positions	Rejected
		5 Positions	Accepted
		8 Positions	Accepted
		7 Positions	Accepted
		9 Positions	Rejected
		[a - z A - Z 0 - 9] +	Accepted
		Special Characters	Rejected
		Blank Field	Rejected

Total Number of Cases	9	Positive	5	Negative	4
------------------------------	---	-----------------	----------	----------	---

Test Cases Document 3

1. Test Cases Doc id	TCD_ABC_10th Jan_3
-----------------------------	--------------------

2. Test Cases Doc Name	Check Login Operation by Clicking Ok Button
-------------------------------	---

3. Test Suite id	TS_Login_FT_PO_3
-------------------------	------------------

4. Test Setup	Login Screen Opened and have Valid and Invalid Employee id's and Passwords in Hand
----------------------	--

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Activate Login Window Fill Employee id and Password Click Ok	All are Valid	Next Window
		Any One Invalid	Error Message
		Any One Blank Field	Error Message

Total Number of Cases	3	Positive	1	Negative	2
------------------------------	---	-----------------	---	----------	---

Test Cases Document 4

1. Test Cases Doc id	TCD_ABC_10th Jan_4
-----------------------------	--------------------

2. Test Cases Doc Name	Check Login Closing by Clicking Cancel Button
-------------------------------	---

3. Test Suite id	TS_Login_FT_PO_4
-------------------------	------------------

4. Test Setup	Login Screen Opened and have some Values in hand to fill Employee id and Password Fields
----------------------	--

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Activate Login Window Fill Employee id and Password Click Cancel	All Filled	Window Closed
		Some Filled	Window Closed
		All Empty	Window Closed

Total Number of Cases	3	Positive	3	Negative	0
------------------------------	---	-----------------	---	----------	---

After Successful **Login**, corresponding **Bank Employee** can get **Options Window**.

He can Click **Fixed Deposit Link** in **Options Window** to get **Fixed Deposit Window**.

Fixed Deposit window will be filled by Corresponding **Bank Employee** by following the Rules Mentioned Below :

1. Depositor Name

Alphabets in Upper Case a One or More Words.

2. Amount

Rupees 1,500/- to Rupees 1,00,000/-
(Rupees Fifteen Hundred to Rupees One Lakh Only)

3. Time to Fix

Up to 24 Months. (1 Month to 24 Months)

4. Rate of Interest

10 % on Amount, when Deposited Amount \geq Rupees 50,000/-
(When Deposited Amount is Greater than or Equal to Rupees Fifty Thousand Only)

8.5 % on Amount, when Deposited Amount $<$ Rupees 50,000/-
(When Deposited Amount is Less than Rupees Fifty Thousand Only)

After Filling above Fields, **Bank Employee** can Click **Deposit** Button to **finish Fixed Deposit** Operation Successfully for Corresponding **Depositor**.

Prepare **Test Scenario's** and **Test Cases Documents** for **Fixed Deposit** Module **Functional Testing**

Test Scenario's

TS5	Check Depositor Name Field.
TS6	Check Amount Field.
TS7	Check Time to Fix Field.
TS8	Check Rate of interest Field.
TS9	Check Fixed Deposit Operation by Clicking Deposit Button.

Test Cases Documents

Test Cases Document 5

1. Test Cases Doc id	TCD_ABC_10th Jan_5
2. Test Cases Doc Name	Check Depositor Name Field
3. Test Suite id	TS_FD_FT_PO_1
4. Test Setup	Valid Employee id and Password for Successful Login and have Valid and Invalid Depositor Names in Hand.
5. Test Procedure	

Step No.	Step Description	Test Data	Expected Output
1	Do Login	Valid Employee id and Password	Options Window
2	Click Fixed Deposit Link	None	Fixed Deposit Window Opened
3	Activate Fixed Deposit Window Fill Depositor Name Field	1 Character	Accepted
		0 Character / Blank Field	Rejected
		2 Characters	Accepted
		256 Character	Accepted
		255 Character	Accepted
		257 Character	Rejected
		([A - Z] + [\ s] ?) { 1 , }	Accepted
		[a - z 0 - 9]	Rejected
		Special Characters except [\ s]	Rejected
		Blank Field	Rejected

Test Cases Document 6

1. Test Cases Doc id	TCD_ABC_10th Jan_6
-----------------------------	--------------------

2. Test Cases Doc Name	Check Amount Field
-------------------------------	---------------------------

3. Test Suite id	TS_FD_FT_PO_2
-------------------------	---------------

4. Test Setup	Valid Employee id and Password for Successful Login and have Valid and Invalid Amounts in Hand.
----------------------	--

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Do Login	Valid Employee id and Password	Options Window
2	Click Fixed Deposit Link	None	Fixed Deposit Window Opened
3	Activate Fixed Deposit Window Fill Amount Field	1.500	Accepted
		1.499	Rejected
		1.501	Accepted
		1,00,000	Accepted
		99,999	Accepted
		1,00,001	Rejected
		[0 - 9] +	Accepted
		[a - z A - Z]	Rejected
		Special Characters	Rejected
		Blank Field	Rejected

Test Cases Document 7

1. Test Cases Doc id	TCD_ABC_10th Jan_7
-----------------------------	--------------------

2. Test Cases Doc Name	Check Time to Fix Field
-------------------------------	--------------------------------

3. Test Suite id	TS_FD_FT_PO_3
-------------------------	---------------

4. Test Setup	Valid Employee id and Password for Successful Login and have Valid and Invalid Time Values in Hand.
----------------------	--

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Do Login	Valid Employee id and Password	Options Window
2	Click Fixed Deposit Link	None	Fixed Deposit Window Opened
3	Activate Fixed Deposit Window Fill Time to Fix Field	1	Accepted
		0	Rejected
		2	Accepted
		24	Accepted
		23	Accepted
		25	Rejected
		[0 - 9] +	Accepted
		[a - z A - Z]	Rejected
		Special Characters	Rejected
		Blank Field	Rejected

Test Cases Document 8

1. Test Cases Doc id	TCD_ABC_10th Jan_8
----------------------	--------------------

2. Test Cases Doc Name	Check Rate of Interest Field
------------------------	-------------------------------------

3. Test Suite id	TS_FD_FT_PO_4
------------------	---------------

4. Test Setup	Valid Employee id and Password for Successful Login and Valid Amount and have Valid and Interest Rates in Hand.
---------------	---

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Do Login	Valid Employee id and Password	Options Window
2	Click Fixed Deposit Link	None	Fixed Deposit Window Opened
3	Activate Fixed Deposit Window Fill Amount and Rate of Interest Fields	Amount > = 50,000 and Interest = 10	Accepted
		Amount > = 50,000 and Interest != 10	Rejected
		Amount < 50,000 and Interest = 8.5	Accepted
		Amount < 50,000 and Interest != 8.5	Rejected

Test Cases Document 9

1. Test Cases Doc id TCD_ABC_10th Jan_9

2. Test Cases Doc Name Check **Fixed Deposit Operation** by clicking **Deposit** Button.

3. Test Suite id TS_FD_FT_PO_5

4. Test Setup **Valid Employee id** and **Password** for Successful **Login** and have **Valid** and **Invalid Values** for all Fields in Hand.

5. Test Procedure

Step No.	Step Description	Test Data	Expected Output
1	Do Login	Valid Employee id and Password	Options Window
2	Click Fixed Deposit Link	None	Fixed Deposit Window Opened
3	Activate Fixed Deposit Window, Fill all Fields and Click Deposit Button	All are Valid	Deposit Successful
		Any One Invalid	Error Message
		Any One Blank Field	Error Message

Test Cases Document

Details and Design Steps of Cases is called as Test Cases Document.

NOTE

1. Due to Lack of Time, Corresponding Test Engineers can follow Partial Format of IEEE 829 Test Cases Document.

2. In above Example Test Cases Documents, **First 4 Fields** i.e.

- * **Test Cases Document id**

- * **Test Cases Document Name**

- * **Test Suite id**

- * **Test Setup**

are called as **Details of Test Cases Documents**.

Fifth Field i.e. **Test Procedure** is Called as **Design Steps of Test Cases Document**.

3. Some Times, Testers are using **Test Environment** Field for Non Functional Test Cased Documents.

4. Some Times, Testers are writing Design Steps only except Details an Test Environment in Test Case Documents, when they themselves are responsible for Corresponding Cases Execution on SUT also i.e. when the Same Tester is working as Test Designer and Test Executor.

d) Collect Test Data for Corresponding Test Cases

After Completion of Customer Requirements Study, Test Engineers are preparing Test Scenario's for responsible Modules and responsible Testing Topics.

Then, Testers are implementing these Scenario's as Detailed Test Cases Documents and then, Collect Test Data for Corresponding Cases.

Test Data

Data used in Testing is called as Test Data.

Test Data is Classified in to Valid Data and Invalid Data.

Valid Data is used for Positive Testing and Invalid Data is used for Negative Testing.

Test Data	
Valid Data	Invalid Data
Positive Testing	Negative Testing

There are 2 ways to collect Test Data

* Real Data from Customer Site (Suitable for Projects)

* Model Data Generated by Tester (Suitable for Projects / Products)

Test Data	
Real Data	Model Data
Customer Site	Tester

Example

" Soap Name " is a Field in SUT Screen.

From Customer Requirements, Soap Name is in Alphabets.

Model Name in Alphabets	Real Name in Alphabets
ABCD	LIRIL

Note :

* It is always better / Suggestive to maintain Meaningful Data.

* When Test Data is Huge for a Test Cases Document, Tester can store that Huge Test Data in a Separate File and Linkup Test Cases Document and Corresponding Test Data File.

e) Review Test Design

After Completion of Test Data Collection, Corresponding Testers can Submit their Scenario's and Test Cases Documents including Test Data to Corresponding Test Lead.

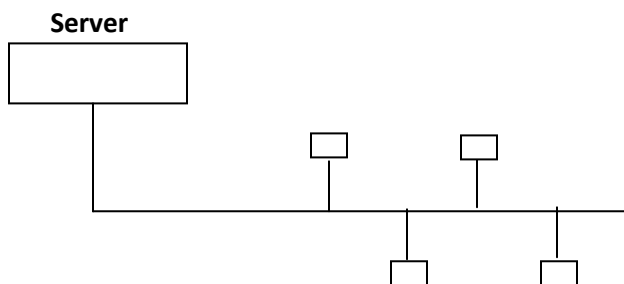
Test Lead can estimate Completeness and Correctness of those Documents and suggest Changes, if needed.

IV) Test Execution

After Completion of Test Design Stage, Corresponding Testing Team can concentrate on Test Execution.

a) Establish Test Environment

In this Step, Corresponding Testers can take the help of Hardware Team / Infrastructure Team to get all required Software and Hardware for Current Project / Product Testing Physically.



(Required Software and Hardware for SUT Testing)

Test Bed

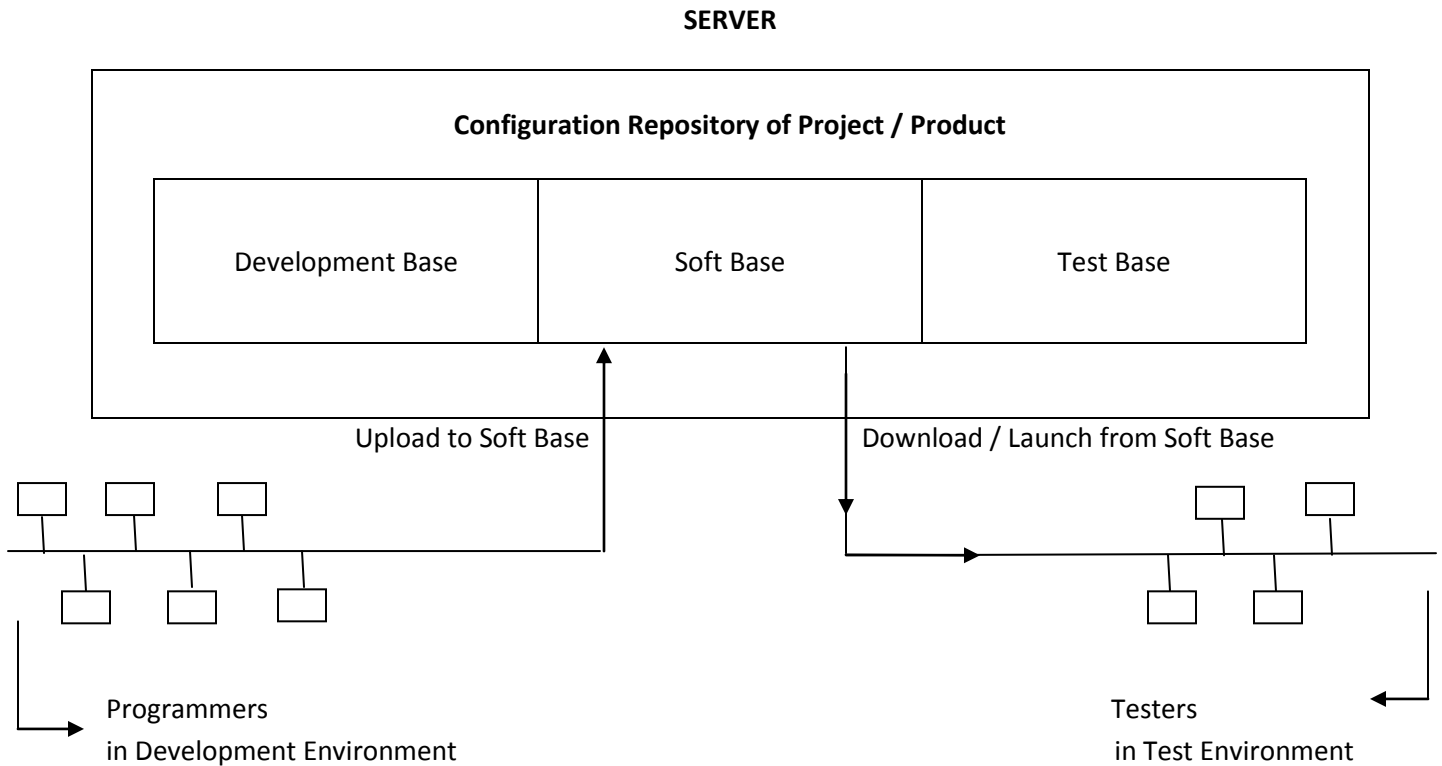
Test Bed = Test Environment + Test Cases Documents

b) Formal Meeting

After Completion of Integration Testing by Developers and After Completion of Test Design and Test Environment establishment by Testers, Corresponding PM can invite Developers and Testers to discuss Software Build Release Process, Defect Reporting Process and Software Build Version Control Process.

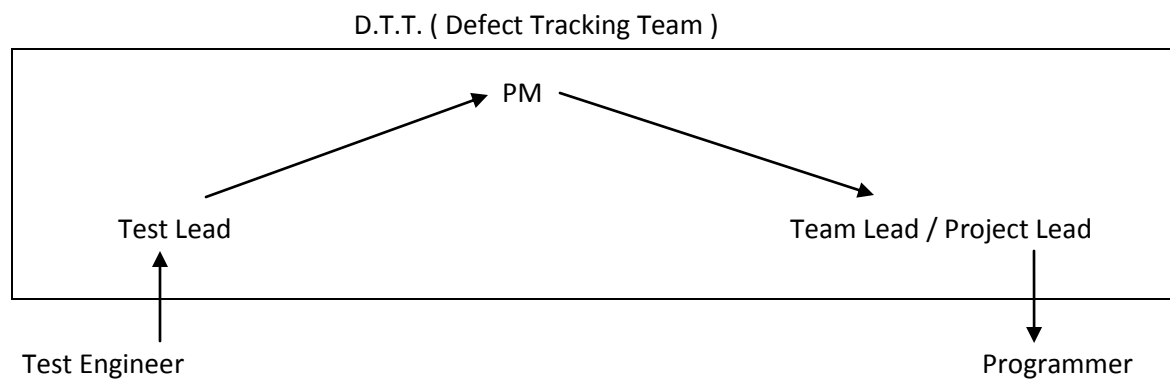
In General, Developers can place Software Build in Software of Server.

Testers are Launching / Downloading that Software Build into Test Environment.



During Tests Execution of that Downloaded SUT, Testers are finding Defects.

They are reporting these defects in the process shown below :



Software Build Version Control

After Fixing Defects, Developers are releasing Modified Software with a New Version Number.

This Version Numbering System is maintained in such a way that it is understandable to Corresponding Testers.

This type of Version Numbering System is called as Software Build Version Control.

Example

Software Build Version		Software Version
Developers	Testers	Customer Site
1.0	1.0	-
2.0	2.0	-
3.0	3.0	-
4.0	4.0	-
5.0	5.0	1.0

from the Above Example,

Developers are releasing Software Build Version 1.0 to Testers

Testers Identify and Report Defects.

Developers Modify and Release Software Build Version 2.0 to Testers

Testers Identify and Report Defects.

Developers making Required Changes and Release it to Testers as 3.0

Testers Identify and Report Defects.

Developers making Required Changes and Release it to Testers as 4.0

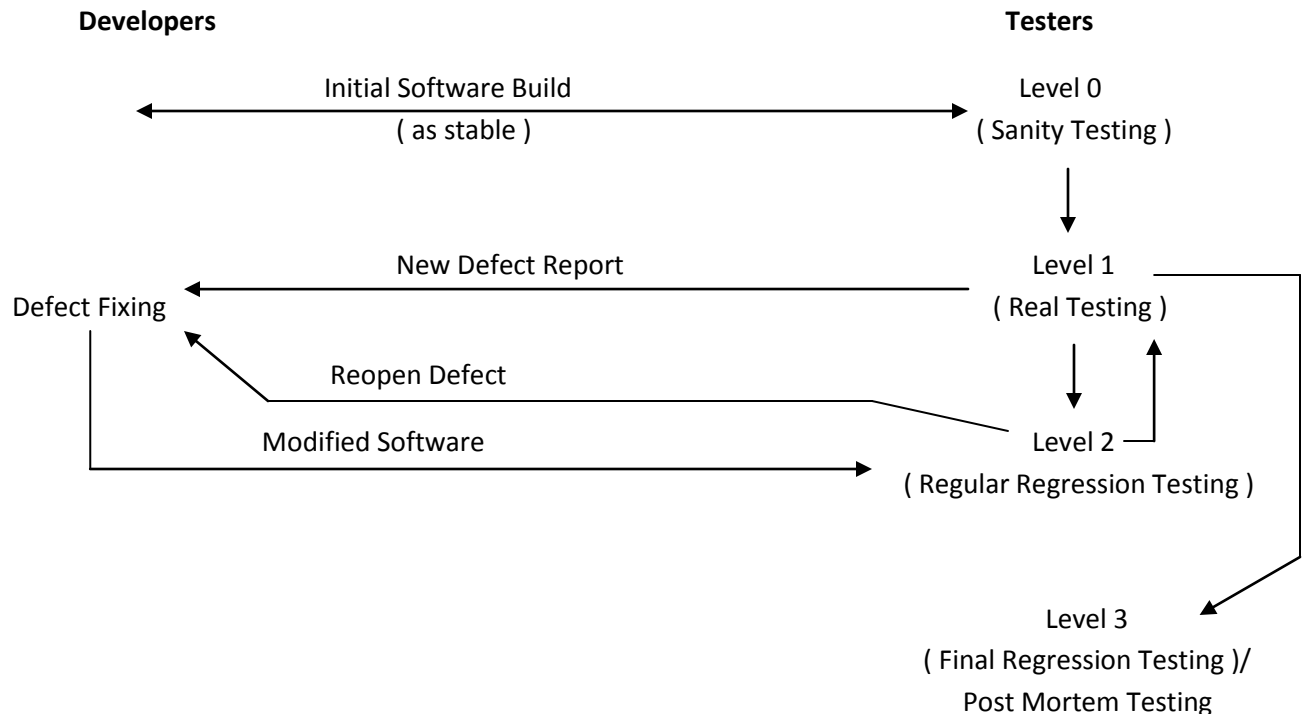
Testers Identify and Report Defects.

This Process goes on until Developers released Software Build is approved by Testers.

Only, The Final Approved Software Version is released to Customer as 1.0

c) Identify Tests Execution Level

After Completion of Formal Meeting with Developers, Corresponding Testers can concentrate on Tests Execution Levels Identification.



d) Levels of Test Execution vs Test Cases

After Completion of Levels of Test Execution Identification, corresponding Testers will select Test Cases for Each Level.

Level 0 (Sanity) (Team Level)

Select Test Cases related to **Basic Functionality** in SUT.

Level 1 (Real)

Select all Test Cases related to **all Functionalities** of SUT (Functional and Non-Functional)

Level 2 (Regression)

Select Test Cases related to **Modified Functionalities** of SUT.

Level 3 (Final Regression)

Select Test Cases related to **Doubtful Functionalities** of SUT.

e) Level 0 (Sanity Testing)

Testing Team is downloading Initial Software Build from Soft base of Server into One System of Test Environment and Operate that Initial Software Build Basic Functionalities with Valid Data to confirm that Software Build is Testable or Not.

Here Testability means Combination of the following Factors

Understandable

SUT Screens are Understandable to Testers.

Operatable

SUT Screens are Operatable by Testers.

Controllable

SUT Screens Operations are Controllable by Testers.
(to do and undo)

Consistency

SUT Screens are Consistent. (to Do and Re Do)

Observable

SUT Screens Operations Process Observable by Testers.

Maintainable

SUT is staying Long Time in Tester Computer without any Need for Re-Installation.

Simplicity

SUT Screens are not having Complexity.

Automatable

SUT is Automatable by using Available Testing Tools in Test Environment.

From the above factors, applied on SUT, Sanity Testing is also called as Octangle Testing / Testability Testing / Tester Acceptance Testing / Build Verification Testing (B.V.T.)

NOTE

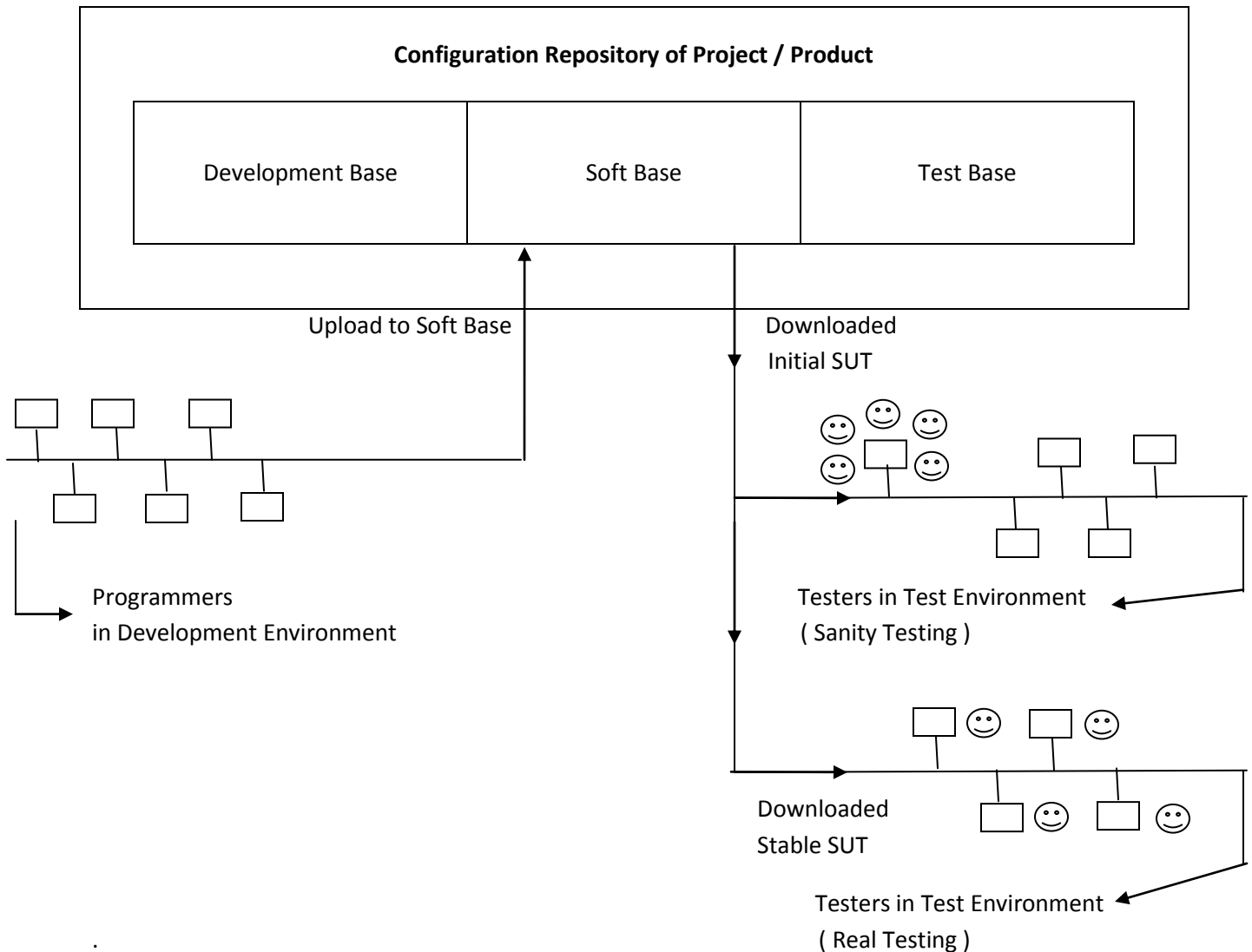
If Initial Software Build is not Testable with respect to above factors, then Testers can REJECT that Initial Software Build and wait for another Initial Software Build from Developers.

f) Level 1 (Real Testing)

(Comparison of Expected Values and Obtained Values)

After Completion of Sanity Testing, Testing Team is receiving a Stable Testable SUT and then Every Tester in the Testing Team can concentrate on All Functional and Non Functional Cases Execution in order to Detect Defects in responsible Module of SUT.

SERVER



P.T.O.

During Real Testing, Corresponding Test Engineer can concentrate on the following process :

Step 1

Collect Test Cases Documents related to responsible Modules and responsible Testing Topics.

Step 2

Arrange those Test Cases Documents as Batches with respect to those Cases Covering Modules or Functionalities.

Step 3

Take One Test Batch or Suite.

Step 4

Take One Test Cases Document in that Batch.

Step 5

Take One Case in that Document.

Step 6

Operate SUT with that Case and Compare Case Expected Value and SUT Actual Value.

Step 7

* If Expected Value is Equal to Actual Value, then go to Execution of Next Case in that Test Cases Document.

* If Expected Value is Not Equal to Actual Value, then go to Stop Execution and Report Corresponding Defect.

Step 8

* If All Cases are Passed in One Cases Document, then go to Next Cases Document.

* If All Cases Documents are Passed in One Batch, then go to Next Batch.

* If All Batches Cases are Passed, then go to Test Closure (Stop Testing)

NOTE

Sanity Testing is Team Level.

Real Testing is Individual Level.

Test Log

While following above processes in Real Testing, Every Test Engineer can prepare a Daily Report Called as Test Log in IEEE 829 Format.

Tester	X Y Z		
SUT Version	1.0	Date of Execution	12th Jan
Test Cases Doc id	Result	Defect id	Comments
TCD_XYZ_10th Jan_1	Passed	-	-
TCD_XYZ_10th Jan_2	Passed	-	-
TCD_XYZ_10th Jan_3	Failed	D1	Defect was Reported
TCD_XYZ_10th Jan_4	Blocked	-	Due to Dependency

Passed

Corresponding Document All Cases Expected Values are Equal to Actual Values of SUT.

Failed

Any One Case, Expected Value is not Equal to Actual Value of SUT.

Blocked

Current Case's Execution Postponed due to Dependency with Failed Cases Document.

g) Defect Reporting

When any Case Expected Value is not equal to Actual Value of SUT, then Tester can Stop Real Testing and concentrate on corresponding Defect Reporting.

Defect Report Format in IEEE 829

1. Defect id

Unique Number or Name for Future Reference

2. Defect Description

Description about Detected Defect.

3. Build Version

Version Number of Build, in which this Defect was Detected.

4. Feature or Module or Functionality

Name of Module in which this Defect was Detected.

5. Failed Test Cases Document id or Name

id or Name of Failed Test Cases Document, in which Cases Execution, this Defect was Detected.

6. Reproducible

Yes / No

Yes

Defect Appears in SUT Every Time, when Test Cases Executed Repeatedly.

No

Defect Appears Rarely, when Test Cases Executed Repeatedly.

Data Driven Testing

Doing Same Test with Various Values to know nature of Defect and Confirmation.

7. if Yes

Attach Test Cases Document.

8. if No

Attach Test Cases Document.

Attach Screen Shot.

9. Status

New	Reopen
Reporting First Time	Re-Reporting

10. Severity

The Seriousness of Defect with respect to SUT Functionality.
(Tester's Point of View)

High (Show Stopper)

Not able to continue further Testing until Defect Fixing.

Medium

Able to continue further Testing. But, Mandatory to Fix Defect.

Low

Able to continue further Testing. But, may or may not to Fix Defect.

Severity		
High	Medium	Low

or

Other Terminology to Describe the Nature of Severity		
Critical	Major	Minor

P.T.O.

11. Priority

The Importance of Defect Fixing with respect to Customer.
(Customer's Point of View)

Priority		
High	Medium	Low

or

Other Terminology to Describe the Nature of Priority		
Critical	Major	Minor

Few Examples of Defect Cases

Data was not inserting in to Database of SUT			
Severity	High	Priority	High

Data was inserted Wrongly in to Database of SUT			
Severity	Medium	Priority	High

Spelling Mistake in Screen of SUT			
Severity	Low	Priority	High

No Left Alignment to Screen Fields			
Severity	Low	Priority	Medium

No Right Alignment to Screen Fields			
Severity	Low	Priority	Low

Wrong Output in Screen, But, this Output is needed as Input for Next Functionalities			
Severity	High	Priority	High

Wrong Output in Screen			
Severity	Medium	Priority	High

Does not Allow Customer Expected Load by SUT			
Severity	High	Priority	High

12. Detected by

Name of Test Engineer.

13. Detected on

Date of Detection and Reporting.

14. Assigned to

D.T.T. (Defect Tracking Team)
(Test Lead + PM + Team Lead)

15. Suggested Fix (Optional)

Suggestions to Fix Defect, if Tester Knows.

h) Defect, Error, Bug

If a Programmer finds any Problem in Program, then that Programmer is calling that Problem as Error or Flaw or Mistake.

If a Testers finds any Problem in Software, then that Tester is calling that Problem as Defect or Issue or Incident.

If a Customer faces any Problem in Software, then that Customer is calling that Problem as Bug.

Tester's Terminology across Countries			
Defect	Defect	Issue	Incident
India	Australia & Newzeland Indian Origins belonging to I.T. Field	America	U.K. Europe

Developers Terminology across Countries	Error
Customers Terminology across Countries	Bug

EXTRA NOTES

* All are Synonyms - But Timing Different i.e. Defect to Tester = Bug to Customer = Error to Programmer.

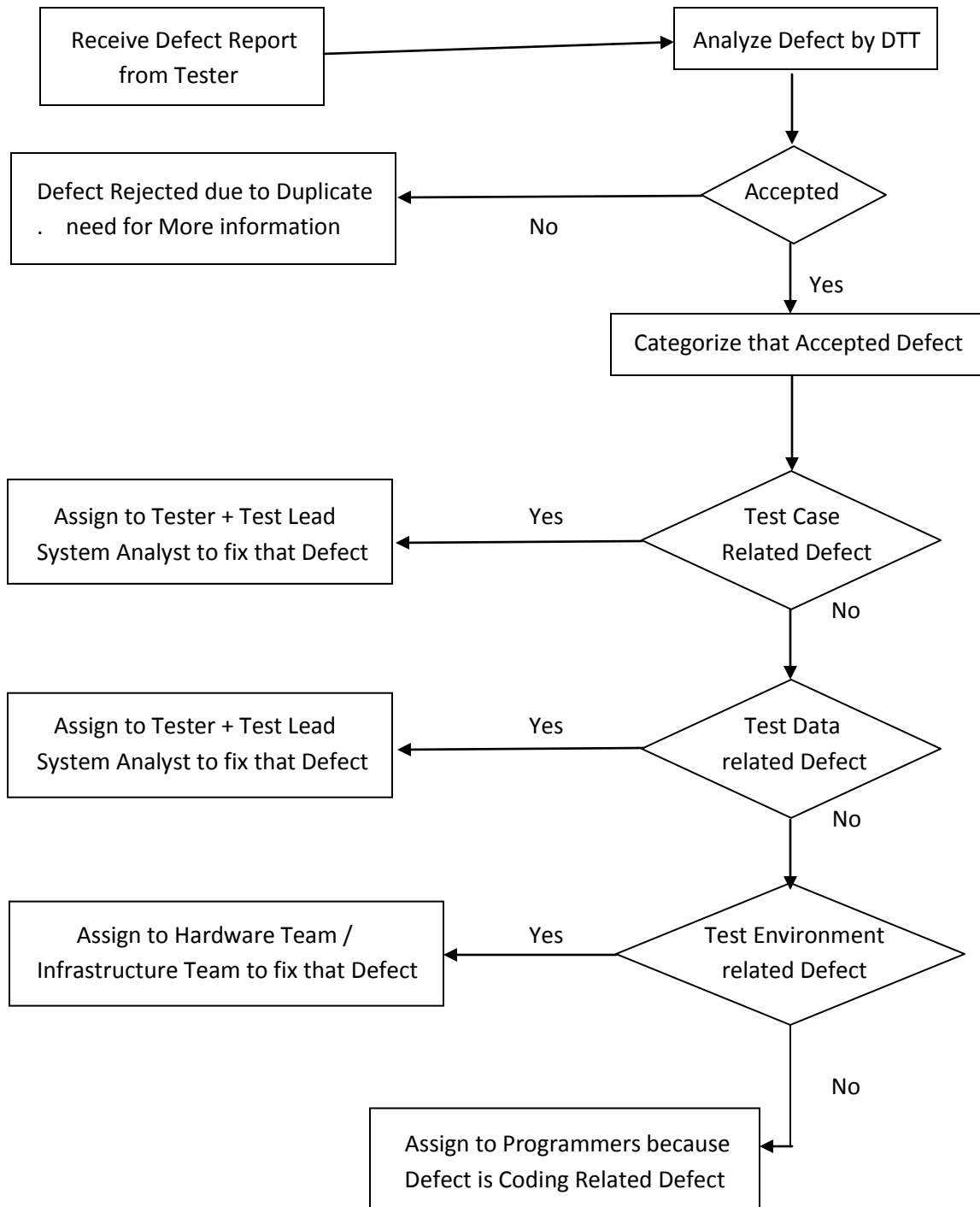
* All are Same.

* Mismatch in Expected and Obtained is called Defect.

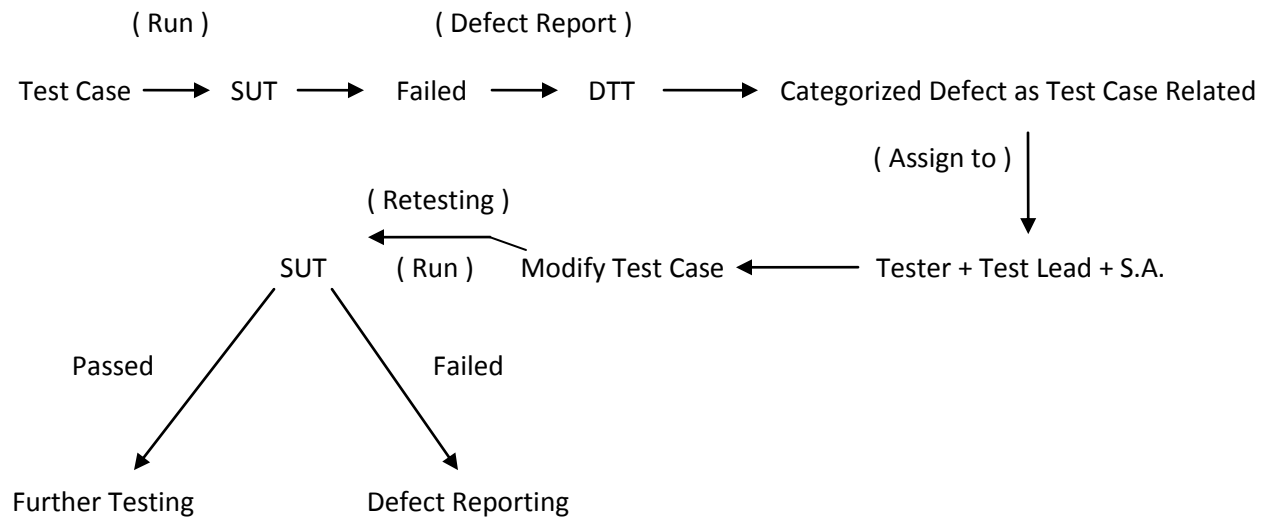
* Defect need not always be imposed on Developer. Sometimes, even Testing Team commits Mistakes like writing Wrong Scenario's or Writing Test Cases incorrectly and so on.

i) Defect Tracking / Defect Analysis

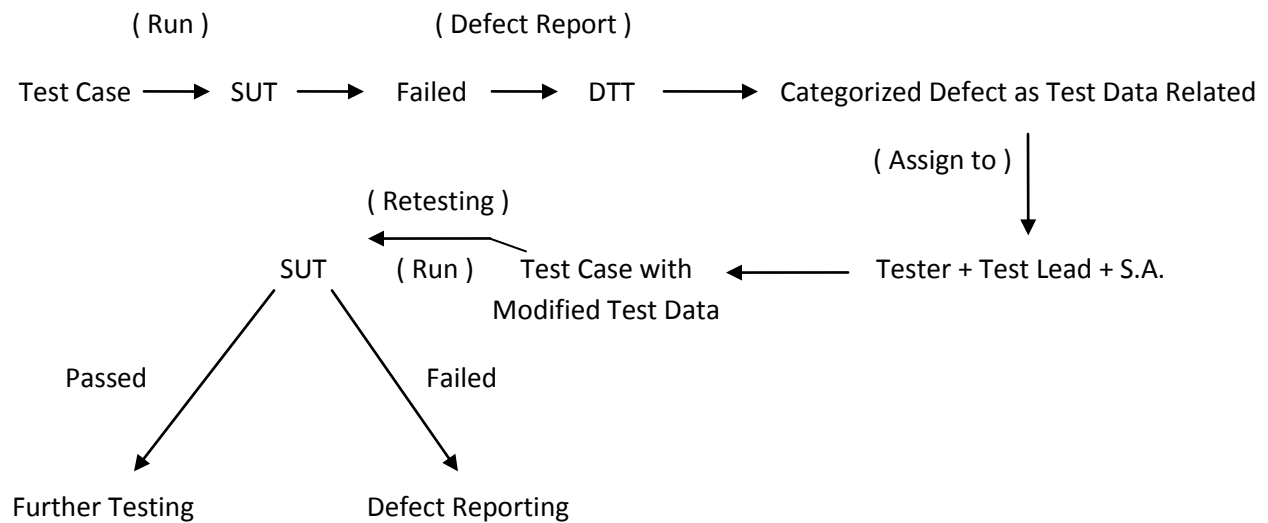
After Receiving Defect Report from Tester, Corresponding Tracking Team Members can analyze that Defect Report as Shown Below :



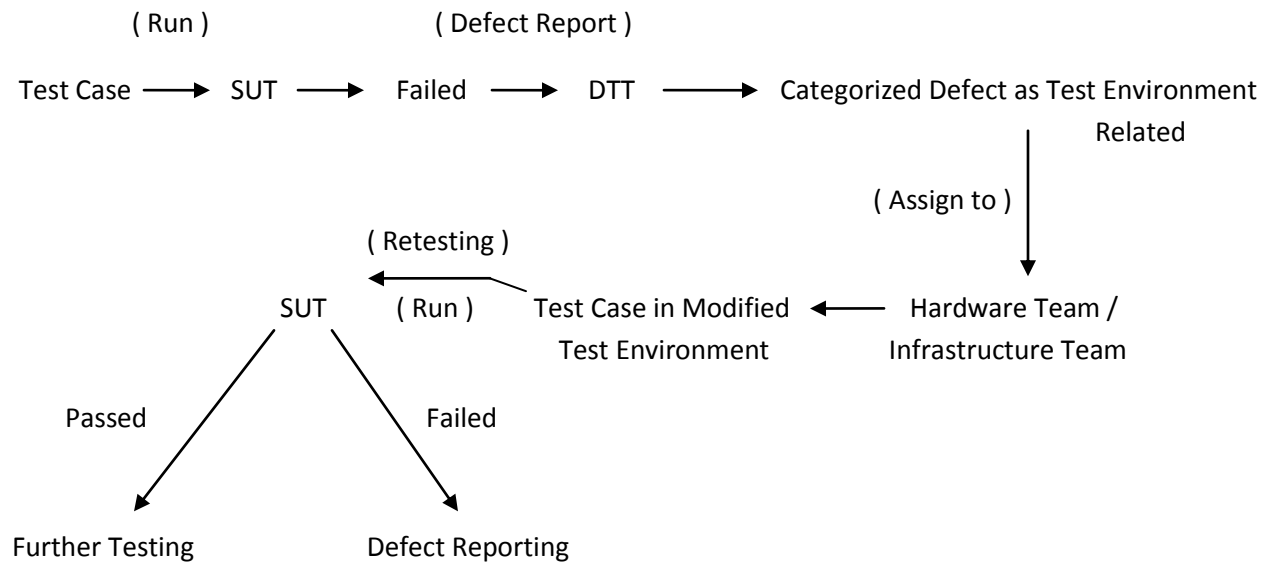
j) Test Case related Defect Fixing



k) Test Data related Defect Fixing



I) Test Environment related Defect Fixing



m) Coding related Defect Fixing

Tester Reported Defect to D.T.T.



D.T.T. categorized that Defect as Coding Related.



Assign that Defect to Development Team to Fix.



Team Lead can conduct Root Cause Analysis to identify Wrong Coding Areas.



Corresponding Programmer can perform changes in that Software Coding to Fix Defect.



Programmers can perform U.T. & I.T. related to Modified Coding Areas.



Place Modified Software Build in Soft base of Server with New Version Number



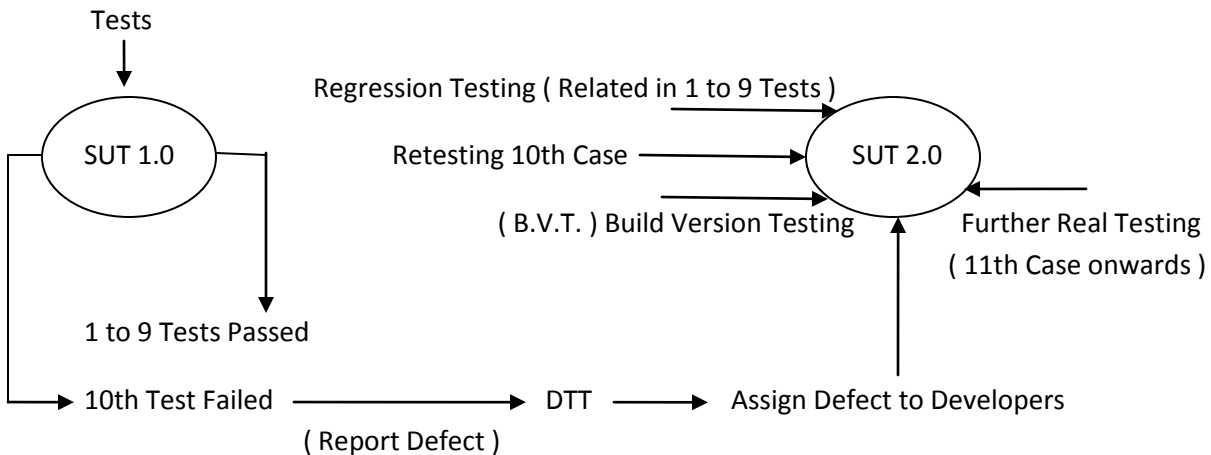
Programmers can conduct Smoke Test on Modified Software Build and send Software Release Note (S.R.N.) to Testing Team.

After Receiving Release Note from Development Team, Test Lead and Corresponding Testers can Study that Document to know regarding Modifications in corresponding Modified Software.

Smoke Test

Final Test done by Developers on Modified Software before sending Software Release Note to Testers.

Example



After Receiving SUT 2.0 Testers conduct all tests related to 10th test and also perform 10th test.

Retesting

Problem solved or not.

Regression Testing

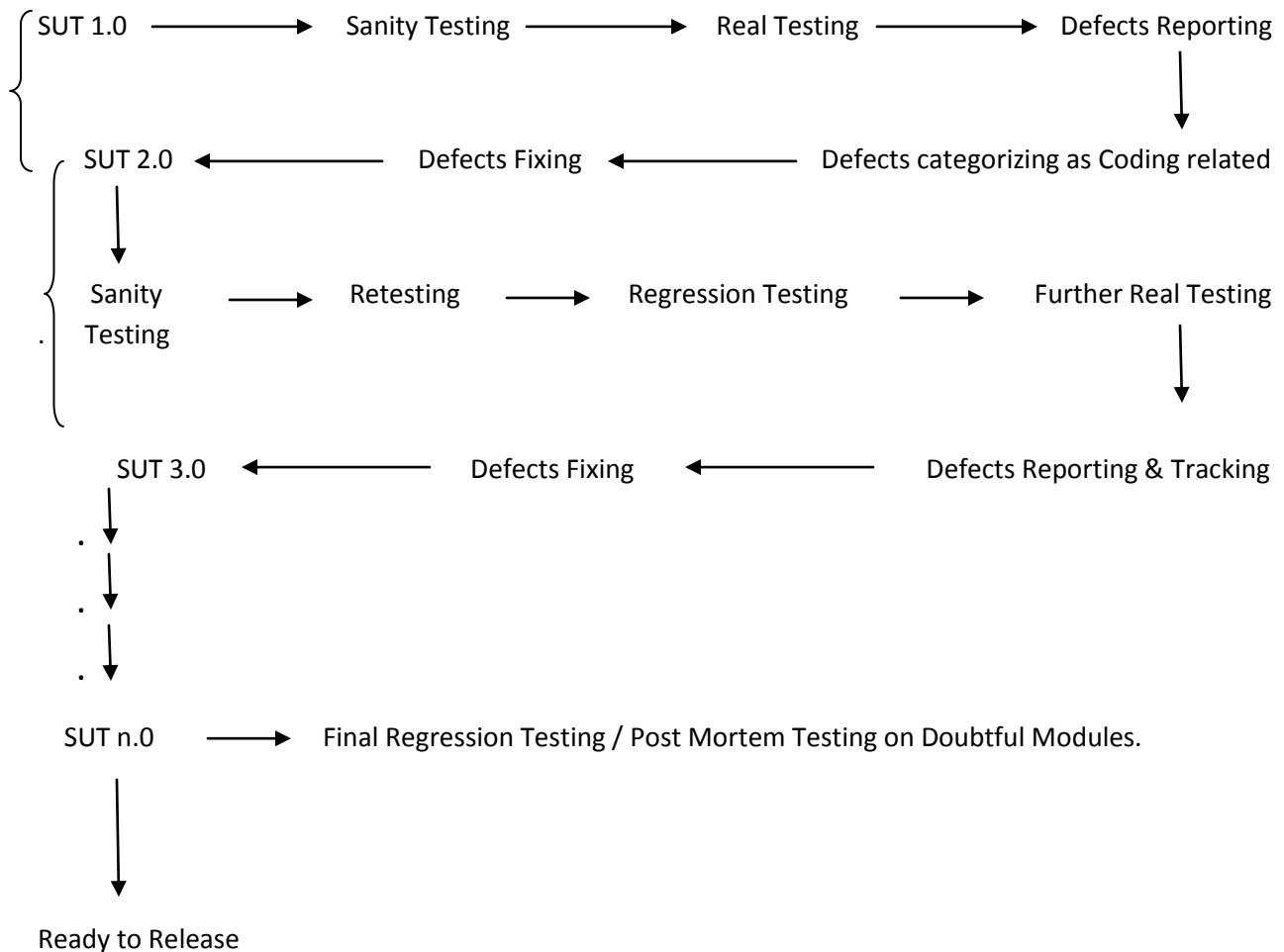
- * Whether Problem is solved or not without leaving any Side Effects i.e. without disturbing any other Modules of the Project.
- * From this Example, Testers are studying Software Release Note given by Developers to understand the Modifications done by them to Fix Defects.
- * Then, Testing Team can download / Launch that Modified Software in one system first (for Sanity Testing) to estimate Testability.
- * Then, Related Testers (may be all) can Launch / Download that Accepted Modified Software Build into their Cabin Systems.
- * Then, corresponding Tester can Re-Execute previously Failed Tests on Modified Software Build to confirm whether that Cases are Passed Now or Not.
- * If these Cases are failed this time also (Now also), then Tester can Re-Open the corresponding Defect.
- * If Previously Failed Tests are Passed this Time, then Tester can Close that Defect and go to Related Passed Test Cases Execution on Modified SUT to identify the Side Effects called as Regression Testing.
- * If Previously Related Passed test was failing now, then Tester can Report that as a New Defect.
- * If Previously Related Passed Tests are Passed now also, then go to Further Real Testing.

Test Conclusions

n) Test Cycles

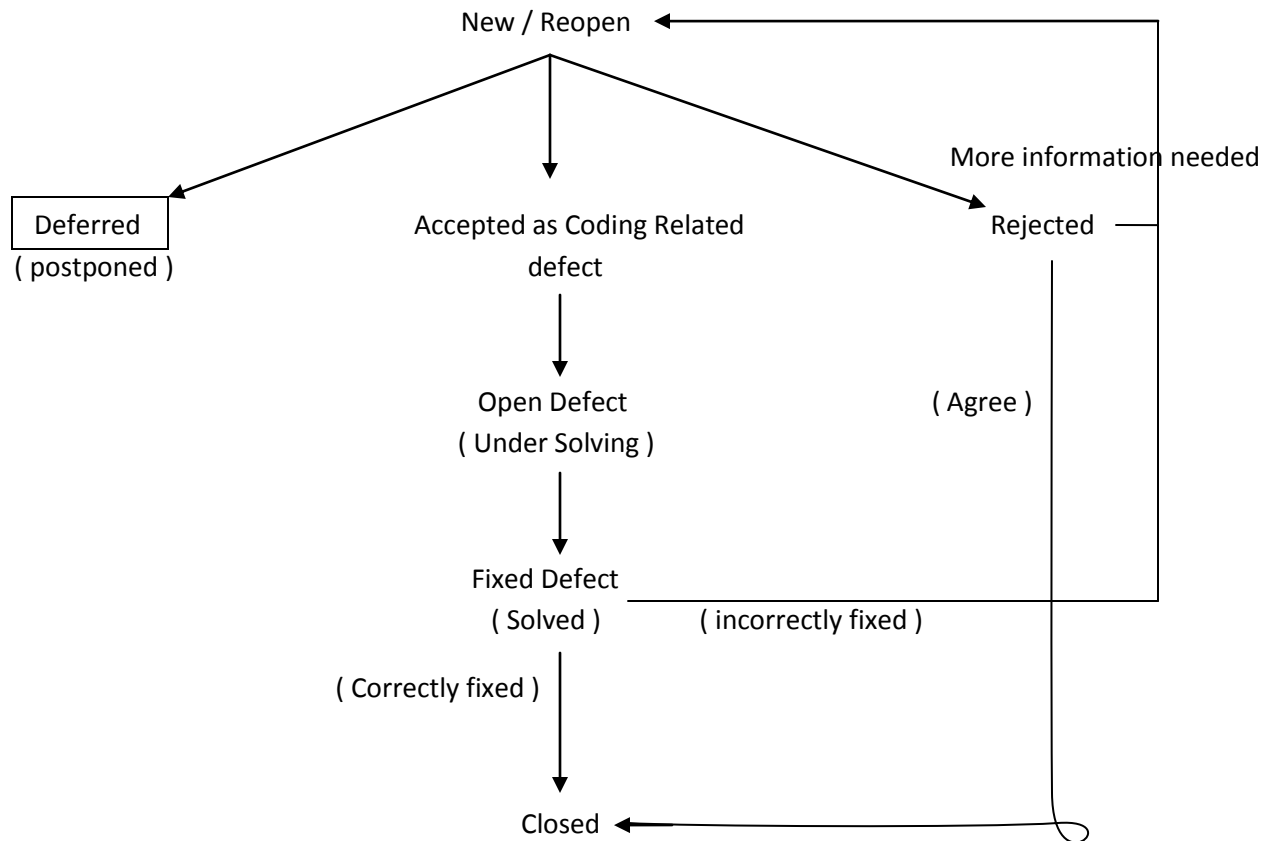
The Time Gap between 2 Consecutive Build Releases is called as Test Cycle.

The Average Time for One Cycle is One Week.



In the Above Diagram, The Time gap between SUT 1.0 and SUT 2.0 is referred as Test Cycle 1 and The Time gap between SUT 2.0 and SUT 3.0 is referred as Test Cycle 2.

o) Defect Life Cycle



Responsibilities of DTT, Developer and Tester from the above Diagram

DTT

Deferred / Accepted / Rejected.

Developer

Open Defect (Under Solving) / Fixed Defect (Solved)

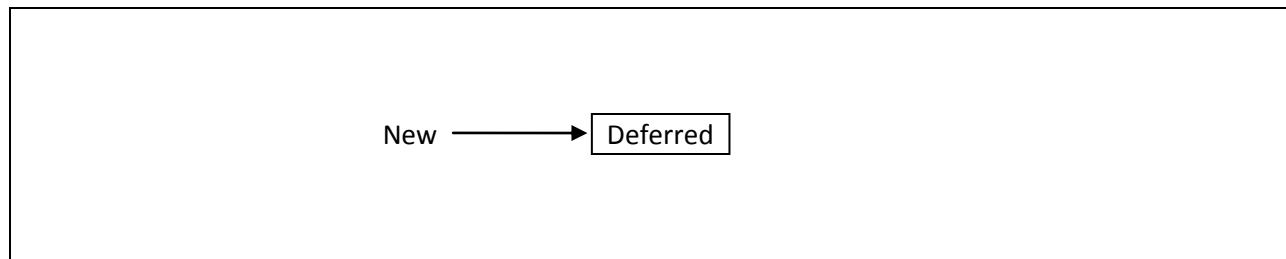
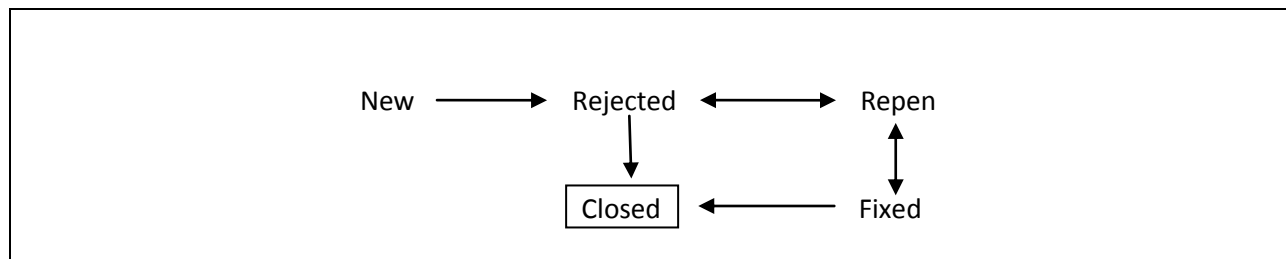
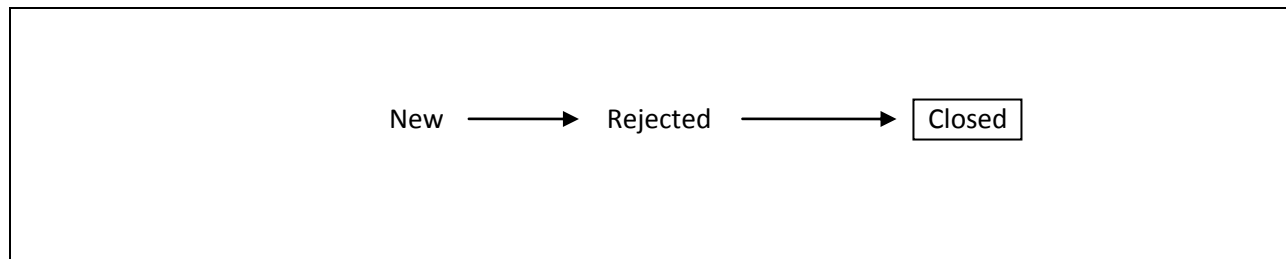
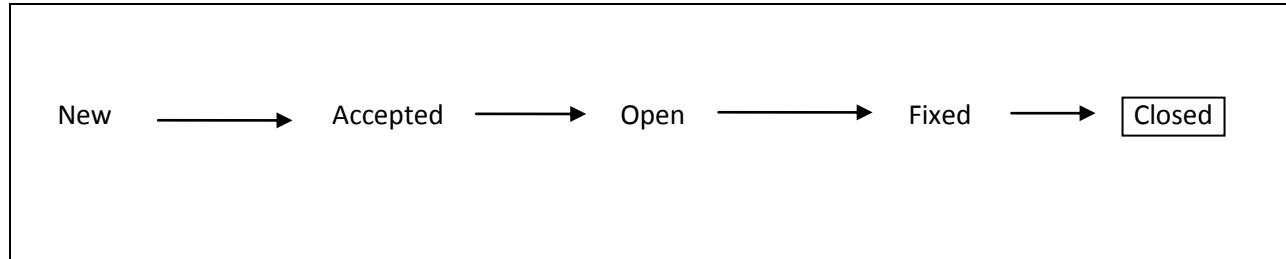
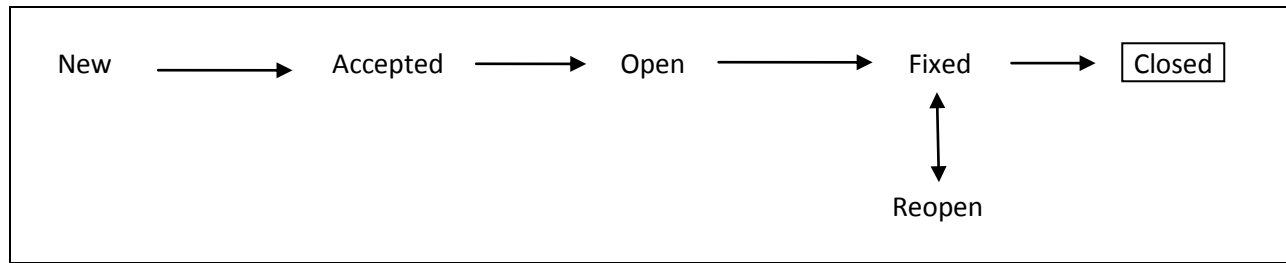
Tester

New / Reopen, Agreed, Closed,

NOTE

Final Status of a Defect should either be Closed or Deferred.

Few Examples



Extra Notes

* In General, Show Stopper occurs in Sanity Testing i.e. in Early Testing Stages.

* In General, Developers collect the Defects and Fix them on Weekends.

V) Test Closure

After Completion of all reasonable Test Cycles, Test Lead is conducting a Final Review Meeting to go to Stop Testing. In this Meeting, Test Lead concentrates on the factors mentioned below :

Coverage Analysis

- * Modules / Features / Functionalities / Requirements Coverage
- * Testing Topics / Testing Subjects Coverage (Functional + Non - Functional)

Defects Density Calculation

Example

Modules in SUT	% Defects
A	20 %
B	20 %
C	40 %
D	20 %

(Need for Final Regression / Post Mortem)

Analysis of Deferred Defects

Whether Deferred Defects are Postponable or not.
(Defects that are Postponed are Postponable or not)

After Completion of Test Closure Review Meeting, Testing Team can concentrate on Level 3 Post Mortem Testing / Final Regression Testing / Pre Acceptance Testing.

In this Stage, Testing Team can follow the Process as shown below

- * Identify High Defect Density Modules in SUT.
- * Repeat related Test Cases Execution on those Modules.
- * If any Golden Defect came, Developers can fix it as early as possible and Testers can close that Defect.

VI) Acceptance Testing

After Completion of Final Regression Testing Stage, the Project Management is concentrating on Feed Back Collection from Real Customers / Model Customers (Alpha Test & Beta Test)

Vli) Sign Off / Role Off

After Completion of Acceptance Testing and their Level Modifications in Software, Corresponding Test Lead can take the help of PM to Role Off some Testers from Current Project / Product and Select remaining Testers for ON-SITE (Release Team)

After Testers Role Off, Corresponding Test Lead can prepare a Final Test Summary Report called as R.T.M. (Requirements Traceability Matrix) in IEEE 829 Format.

Requirements / Modules	Test Cases Documents id's	Passed / Failed	Defect id	Closed / Deferred	Comments

The above Table is defining Mapping in between Requirements and Defects via Test Cases.

Example

Requirements / Modules	Test Cases Documents id's	Passed / Failed	Defect id	Closed / Deferred	Comments
Login	TCD_XYZ_10th Jan_1	Passed	-	-	-
	TCD_XYZ_10th Jan_2	Passed	-	-	-
	TCD_XYZ_10th Jan_3	Passed	-	-	-
	TCD_XYZ_10th Jan_4	Failed	D1	Closed	-
	TCD_XYZ_10th Jan_5	Passed	-	-	-

After Submitting this Table to PM, Test Lead can also Role Off from Current Project / Product.