BASIC FUNDAMENTALS OF TESTING

Software:

- A set of minimal data (or) set of computer programs that required to run a particular system in the real world.

Eg: banking software, telecom software, retail software embedded software etc…

Testing:

- It is process of identifying the defects, isolating the defects, subjecting for the rectification in order to ensure error free product and hence at the end customer satisfaction.

Objectives of Testing:

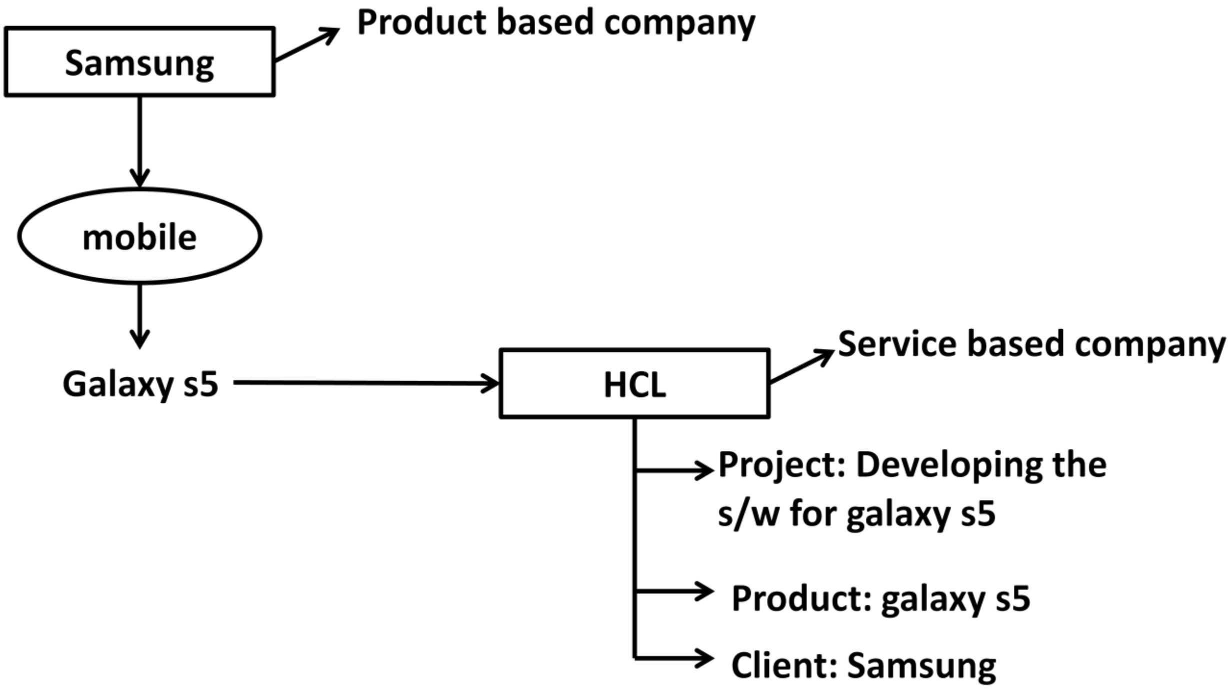
- To ensure error free product.
- To provide clear documentary proof.
- To avoid negative feedbacks from the customers.
- To provide the quality product.

Quality:

- Quality is defined as not only the justification of requirements but also it must be user friendly to the clients/customers.

Companies:

- Two types of companies are available in the market
    - Product based companies
    - Service based companies



Product:

- It is something which is developed by the point of view of multiple customers in the market.

Project:

- It is something which is developed by the point of view of specific customer requirement.

Service:

- Giving the service to the customer/client.

Error:

- Any incorrect human action that results to mistakes is called as error.

Bugs:

- Error in coding is called as bug.

Defect:

- It is defined as deviations between the expected behavior and actual behavior is called as defect.

Failure:

- Problems identified by the end users while using the product is called as failure.

NOTE: Errors results in the presence of defects where as the presence of defects results in the failure of the product.

Causes of Defects:

- Poor requirement.
- Miss communication.
- Improper designs.
- Poor coding.
- Lack of testing skills.
- Inadequate schedules.
- Improper planning.
- Work pressure.
- Lack of domain knowledge etc…

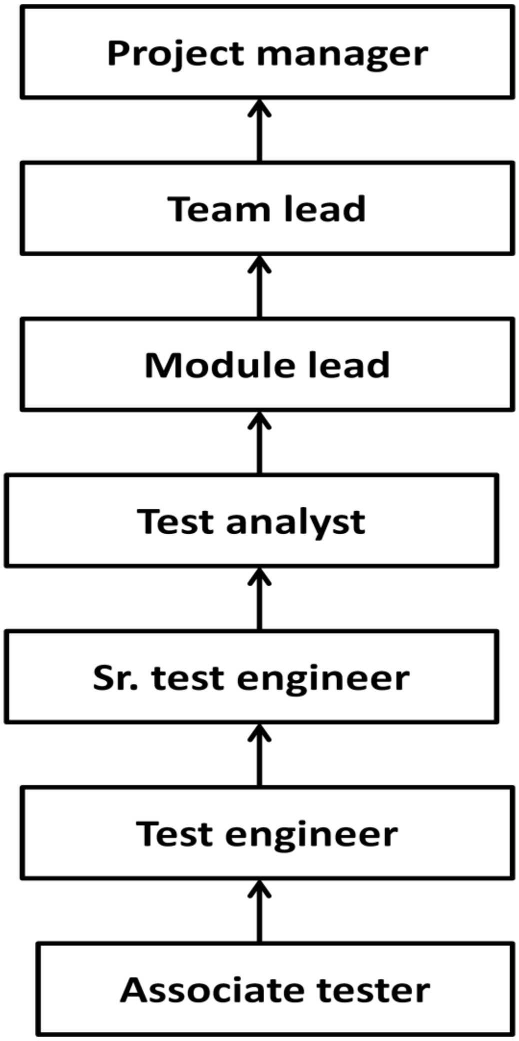
Domain Specifications:

- It is nothing but specialized field. Hence the various domains in the market are
  - BFS (Banking and Financial Services)
  - HMS (Health Management system)
  - ITIS (IT Infrastructure)
  - Retail
  - Application
  - Real estate
  - Travel and tourism
  - Airlines
  - Education etc….

Qualities of a Test Engineer:

- Good communication.
- Strong technical stuff.
- Good judgment skills.
- Diplomatic.
- Creativity.
- Quality oriented mind setup.
- No compromise in quality.
- Test to break attitude.
- Patientance.

Software Testing Team:



Project manager

Team lead

Module lead

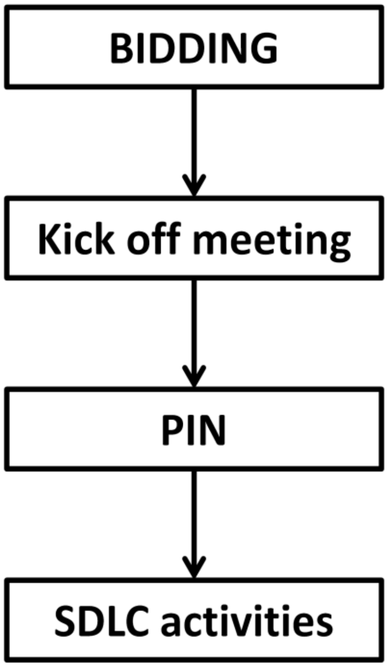
Test analyst

Sr. test engineer

Test engineer

Associate tester

PROJECT ALLOCATION



PIN=Project Initiation Note

<u>Bidding:</u>

- It is defined as request for proposal, estimations (time, budget, man power, resources etc…) and official agreement between two parties i.e sign in agreement.

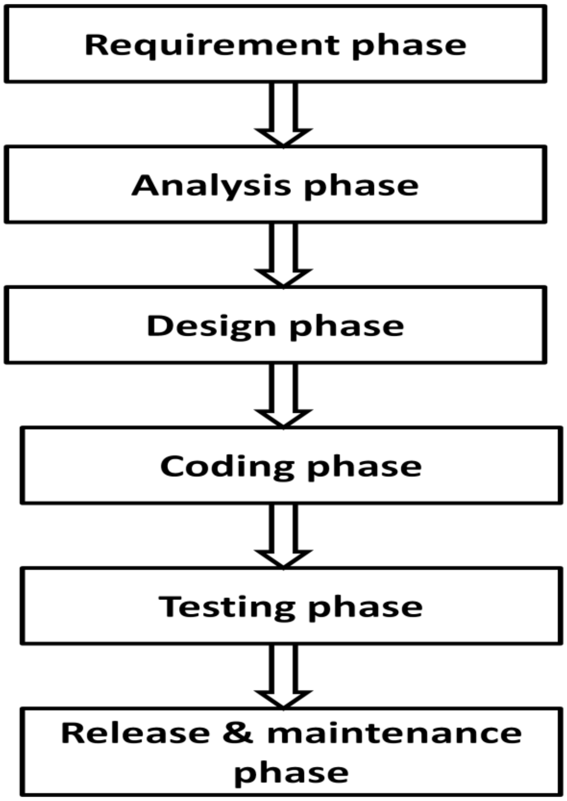
<u>Kick off Meeting:</u>

- These are the initial meetings which are conducted in the organization soon after the project is signed.
- These meetings are conducted among higher and middle level managements.
- The objective of these meetings is to discuss the overview of the project and also to select a suitable project manager.

PIN:

- Once a project manager is selected, project manager prepares a mail, hence this mail itself is called as project initiation note and this mail will be shared to CEO of the company for the approval.
- Once the mail is approved project manager prepares one more mail hence this mail is also called as project initiation note and shares this mail to all concerned team members in the organization stating that they are about to start a project.

SDLC PROCESS:



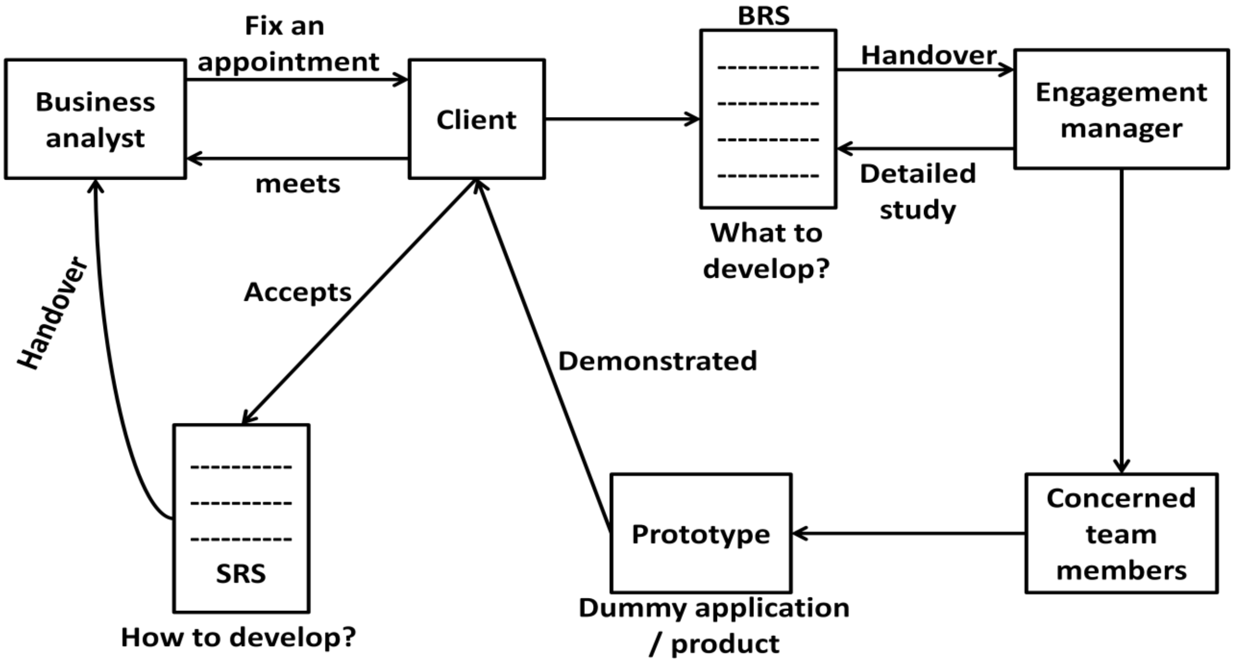
Requirement Phase

Roles:

- Business Analyst (BA)
- Engagement Manager (EM)

Task: Gathering the information from the clients.

Process:



- ➢ BA fixes an appointment with the client, meets the client by collecting the standard template from the organization and gathers the information, prepares a document called as BRS and handovers to EM.
- ➢ EM will do the detailed study to the BRS document and hence if any extra requirements are mentioned by the client EM deals with extra time and budget.
- ➢ If requirements are not clear EM will be asking concerned team members to develop the sample prototype.
- ➢ Hence this prototype will be demonstrated to the client for the approval.
- ➢ Based on the client approval information SRS document will be prepares and this will be handover to BA.

Proof:

- ➢ The document proof of requirement phase is BRS document.

NOTE: Different companies calls different names for the BRS document as follows

- Business Requirement Specification (BRS)

- Customer Requirement specification (CRS)
- User Requirement specification (URS)
- Business Design Document (BDD)
- Business Document (BD)

NOTE: Different companies calls different names for the SRS document as follows

- System Requirement Specification (SRS)
- Functional Requirement Specification (FRS)

Analysis Phase

Roles:

- Project Manager (PM)
- Technical Manager (TM)
- System Analyst (SA)

Process:

- ➢ Feasibility study – detailed study to FRS
- ➢ Tentative plans – temporary plans
- ➢ Time
- ➢ Budget
- ➢ Man power
- ➢ Resources
- ➢ Technology selection
- ➢ Environment

Design Phase

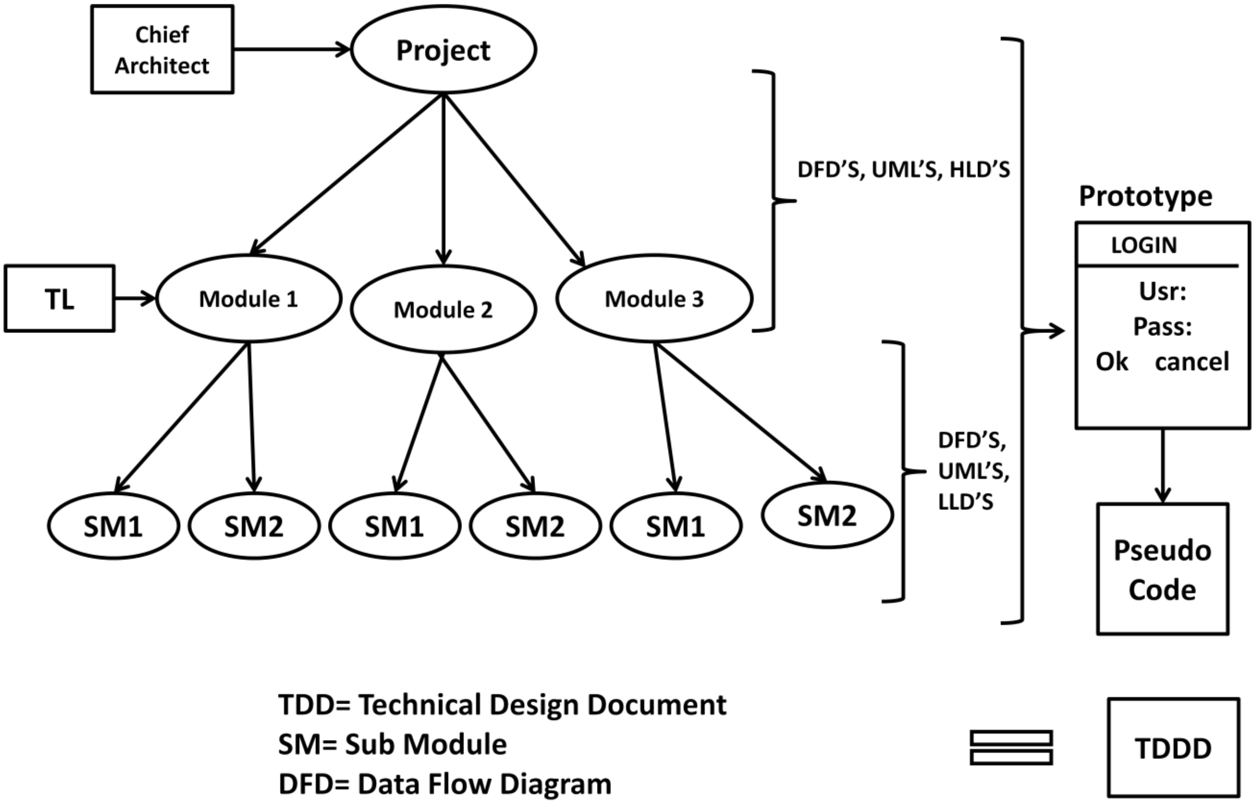
Roles:

- Chief Architects (CA)
- Technical leads (TL)

Task:

- ➢ Preparing the higher and lower level designs to the project.

Process:



TDD= Technical Design Document
SM= Sub Module
DFD= Data Flow Diagram
UML= Unified Modeling Language

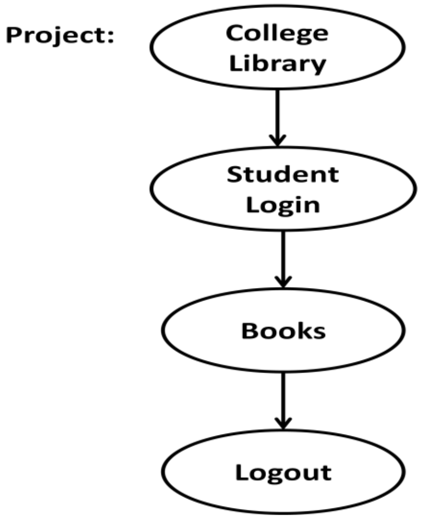
- The chief architect will divide the project into modules by drawing the data flow diagram's and by using UML prepares the higher level design.
- The tech lead and his team members will divide the modules further into sub modules by drawing the same data flow diagram's and by using UML prepares the lower level designs.
- Finally the design team prepares the prototype and also prepares the pseudo code to the application.

Proof:

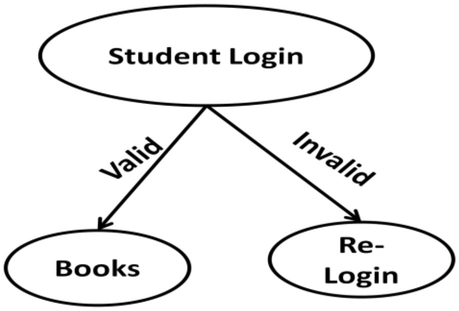
- The document proof of design phase is technical design document (TDD).

HLD= Higher Level Design
LLD= Lower Level design

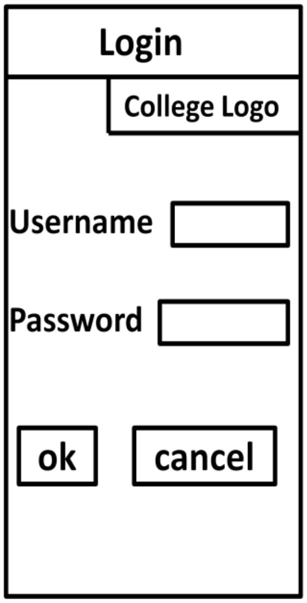
HLD



LLD

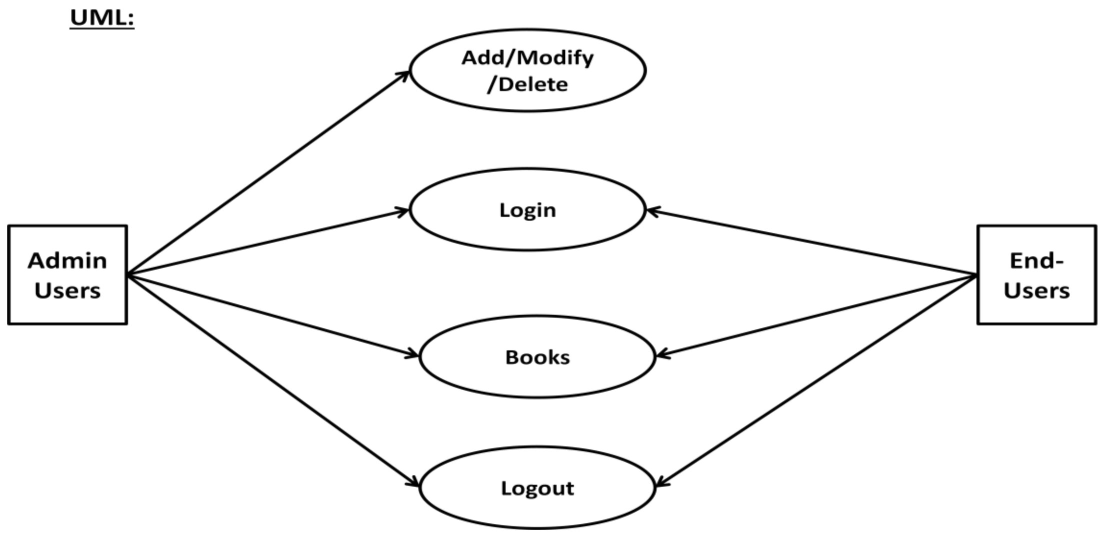


Prototype



Pseudo Code

Usr name-int
Password-varchar(10)
Submit-function



Pseudo code: Pseudo code document contains a set of English statements (or) step by step algorithm which makes developers more comfortable while developing the source code.

Coding Phase

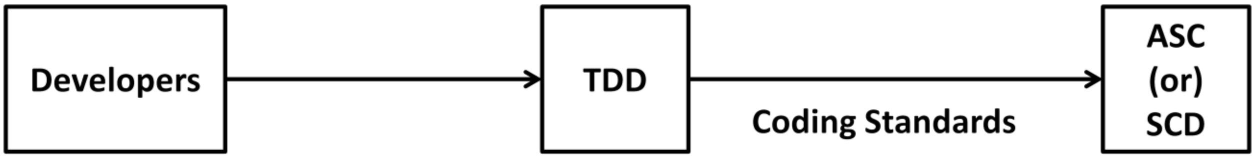
Roles:

- Programmers/Developers

Task:

- Developing the source code to the application.

Process:



ASC= Actual Source Code
SCD= Source Code Document

- In this process by referring the technical design document and by following the coding standards like proper indentation, backgrounds, colors, logo's etc developers will develop the actual source code.

Proof:

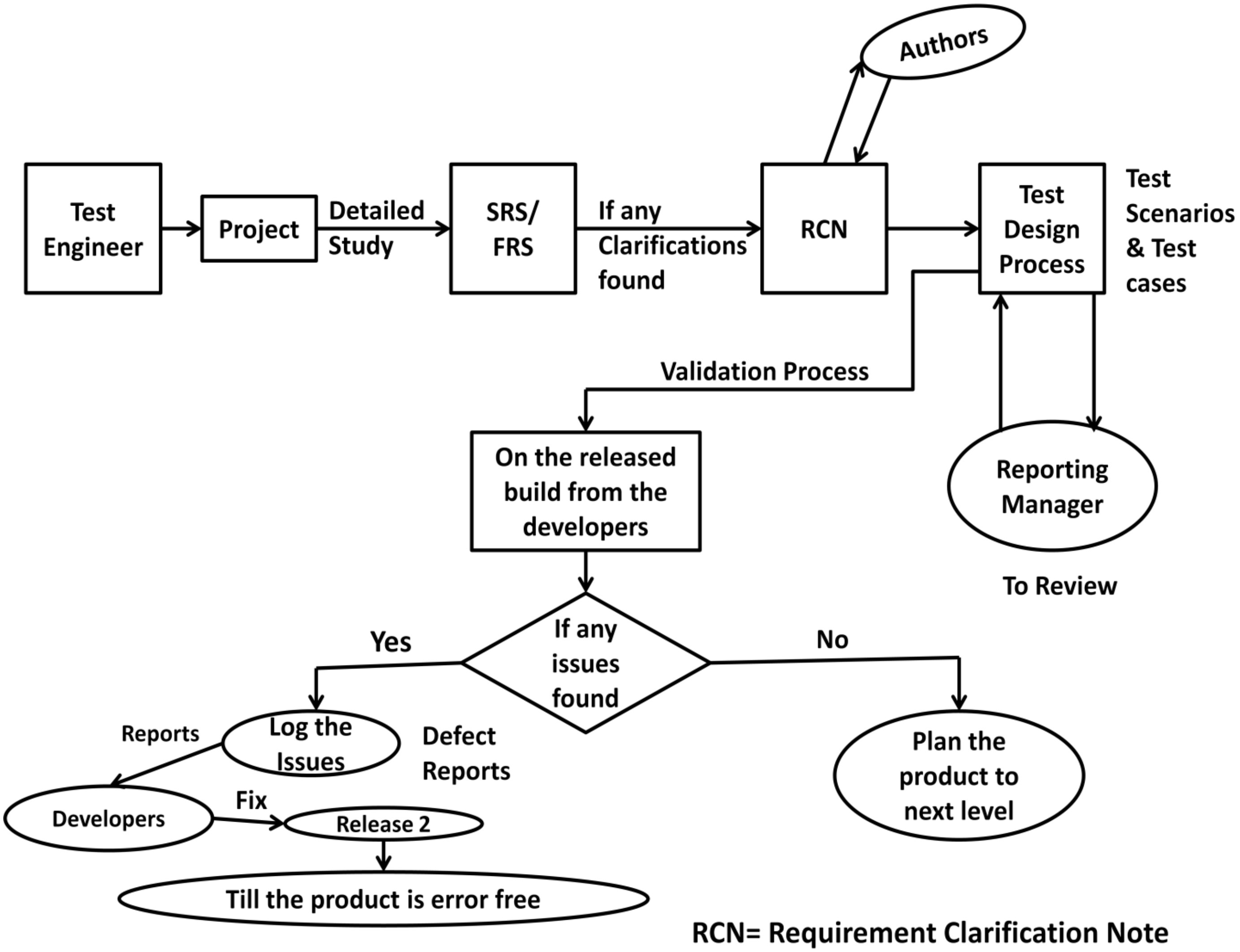
- The document proof of coding phase is SCD.

Testing phase

Roles: Test engineers.

Task: Testing the project.

Process:



RCN= Requirement Clarification Note

- Once a project is assigned to test engineer tester will do the detail study to the SRS/FRS document in order to analyze the client requirements.
- If any clarifications found in the technical document these clarifications will be specified in the RCN document and these documents will be shared to authors for the clarification.
- Once the clarifications are done testers will involve in the test design process i.e by writing the test scenarios and test cases.
- Once the test design process is complete these documents will be shared to reporting manager to review the documents. Hence this entire process is called as verification process.
- Once these documents are approved testers will involve in the test execution process on the released build from the development team.
- At the time of executing the test cases if any issues found these issues will be logged in the defect reports and these reports will be shared to developers to fix the issues.
- Developers will fix the issues and release the modified build to the testing department.
- On these modified build testers once again conduct re-testing (or) regression testing and hence this cyclic process will be carries out till the product is "**error free**".
- If there are no issues in the product, it will be planned to the next level.
- Hence the entire process is called as validation process.

Proof:

- The document proof of testing phase is error free product.

Release and Maintenance Phase

Roles:

- Installation engineer and
- Deployment engineer

Task:

- Releasing the product to the client and giving the service.

Process:

- The installation engineer will go to the client location and deploys the software in the client environment.
- Hence this process itself is called as release.
- Once the product is released clients start using the product and hence client will be expecting the "**maintenance**".
- Two types of maintenance are there
    - Normal maintenance
    - Continuous maintenance

Normal maintenance:

- It is a type of maintenance where the service can be given to the client for a specific time period.

Continuous maintenance:

- It is a type of maintenance where the service can be given to the client for a long period of time.
- Hence for this client has to pay the money.
- In future if any new enhancements/modifications expected by the client for this also client has to pay the money.

Proof:

- The document proof of release and maintenance phase is signoff agreement.

SORTS OF TESTING

- There are two sorts of testing's are there
  - Un-conventional testing
  - Conventional testing

Un-conventional testing:

- This testing is conducted by QA engineers.
- In this process QA engineers will involve in the project from the initial phase of SDLC to end phase of SDLC.
- It is also called as QA process.

Quality Assurance (QA): It is a process of monitoring and measuring the strength of the entire development process is called as QA.

Conventional Testing:

- It is conducted by test engineers.
- In this process testers will involve in the project from the development phase of SDLC to testing phase of SDLC.
- It is also called as QC process.

Quality Control (QC): The validation of deliverables (documents) of the development process is called as QC.

NOTE: Testing can also be defined as it is a process of QA and QC.

TESTING METHODOLOGIES

- Three types of testing methodologies are there
  - White box testing
  - Black box testing
  - Grey box testing

White box testing:

- It is conducted by the developers.
- Testing conducted by the developers on the source code to check whether the source code is working fine (or) not is called as white box testing.
- It is also called as structural testing (or) clear box testing.
- In white box testing developers will be conducting
  - Unit testing
  - Integration testing

Black box testing:

- It is conducted by test engineers.
- Testing conducted by the testers on the application to check whether the application is working as per the specifications (or) not is called as black box testing.
- In black box testing test engineers will conduct
  - System testing
  - User acceptance testing (UAT)

Grey box testing:

- It is a combination of white box and black box testing.
- The best example for grey box testing is "**Database Testing**".

TESTING PRINCIPLES

Early testing:

- It means that we have to start testing the project as early as possible i.e testing must be always started from the requirement phase itself.
- Hence the cost of the bug will be very high at the testing phase.

Testing is context dependence:

- It means that we have to select (or) opt appropriate testing approach based on the type of the application we are testing.

Testing shows the presence of defects:

- It means that we have to teat an application with an intention of showing the defects.
- Hence for this negative testing is best suitable approach.

Risk (or) priority based testing:

- It means that we have to identify the operations which most likely to cause failures and then testing these functionalities on priority basis are called as risk based testing.

Exhaustive testing is impossible:

- Testing everything is called as exhaustive testing.

  (Or)

  If you test the functionality with all possible valid and invalid inputs then it is called as exhaustive testing.
- Hence exhaustive testing is impossible.

NOTE: As exhaustive testing is impossible the best solution to ensure requirement coverage is "**Test Design Techniques**".

Pesticide paradox:

- It means that if prepared test cases are not helping to find the defects we can add, modify, delete our test cases for the better testing approach.

NOTE: Hence this process itself is can also be called as exploratory testing.

Defect clustering:

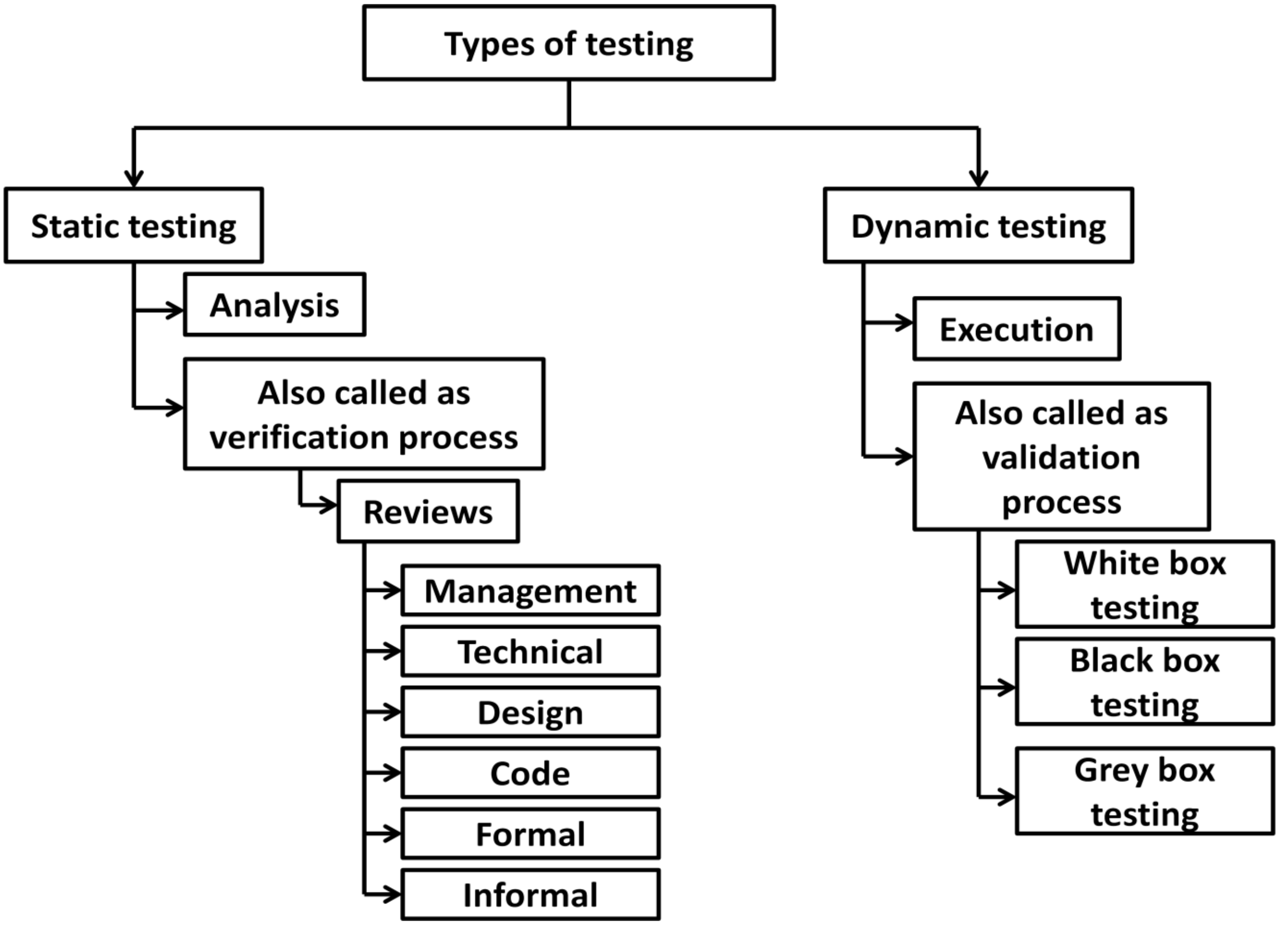
- Clustering is nothing but grouping.
- It means that the smaller modules (or) functionalities may contain more number of defects and hence concentrate on these modules also.

Absence of error falloncy:

- Fallоncy is nothing but false statement i.e 100% error free application is impossible.

TYPES OF TESTING

- Two types of testing's are there
    - Static testing
    - Dynamic testing



STATIC TESTING:

- It is a process of verifying "are we implementing the right product (or) not".
- Static testing can also be called as verification process. In static testing analysis will be carried out.
- Static testing will be carried out through the help of "**reviews and inspections**".

Review:

- Examining project related (or) platform related work is called as review.
- The various types of reviews that can be conducted in the project are as follows

Management reviews:

- These reviews are conducted among higher level and middle level managements.
- The objective of management reviews are to find out the "**slippage's**" in the project.
- **Slippage:** Deviations between the planned efforts to the actual effort.
- These reviews are conducted daily/weekly/monthly.

Technical reviews:

- These are the reviews conducted among the technical members such as design teams, developers, testers to decide the best approach of implementing a job. If there are any questions while implementing the technical work.

Design reviews:

- These reviews are conducted among the architects to discuss the higher and lower level designs of the project.

Code reviews:

- These are the reviews conducted among developers on the source code to confirm the coding standards.

Formal reviews:

- If a review is conducted by pre-plan, followed by set of predefined procedures, by proper documents then these reviews are called as formal reviews.
- The best examples for formal reviews are client reviews, inspections, audits.
    - **Client review:**
        - These reviews are conducted by client to check the progress of the project.
        - The mode of interaction of client can be telephonic, mail, video conference, client visits, on site members.
    - **Inspections:**
        - If a formal review is conducted while executing a task then it is called as inspection.

- **Audits:**
  - If a formal review is conducted after completion of a task to confirm does the task is accomplished as per the predefined checklist (or) not then it is called as audit.

Informal reviews:

- If a review is conducted without any proper preplan, not followed by set of predefined procedures, improper documents then these type of reviews are called as informal reviews.
- The best example for informal review is "**peer review**".
  - **Peer review:**
    - Discussions among the colleagues are called as peer review.

NOTE:

Authors: One who prepares the document.

Moderator/inspection leader: One who is mainly responsible for the review activity.

Reviewers/inspectors: Participants of review activity are called as reviewers.

Scribe/recorder: One who records the defect during the review activity.

Walkthroughs: Knowledge transfer sessions are called as walkthroughs i.e training sessions.

DYNAMIC TESTING:

- It is a process of validating whether the developed product is right (or) not.
- It is also called as validation process.
- In dynamic testing execution will be carried out.
- Dynamic testing can be conducted by
  - White box testing
  - Black box testing
  - Grey box testing

Objectives Of Reviews:

- Reviews are helpful to
  - determine the issues in the requirements
  - determine the issues in designs
  - determine the issues in coding
  - improve the testing process
  - deliver the quality product to the customer

ENVIRONMENTS

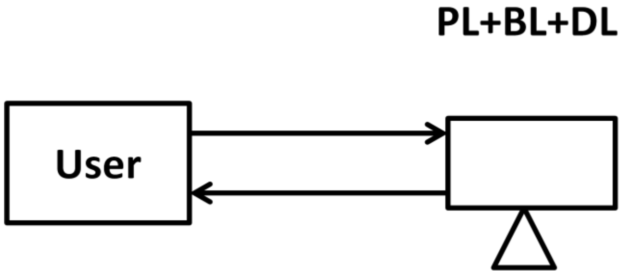
- ➢ Environment is a group of hardware's with some software's where we can install presentation logic, business logic and database logic.
  (Or)
  Environment is a combination of presentation layer, business layer and database layer which holds the presentation logic, business logic and database logic.

TYPES OF ENVIRONMENTS:

- ➢ Four types of environments are there
  - Stand alone environment (1-tier architecture)
  - Client/server environment (2-tier architecture)
  - Web environment (3-tier architecture)
  - Distributed environment (N-tier architecture)

Stand Alone Environment:

- ➢ In this environment single tier is present, in the same presentation logic, business logic and database logic are present.
- ➢ If at all if the application is being used by single user, single machine, single place then the suggested environment is 1-tier architecture.
- ➢ Examples are paint, MS-office etc.

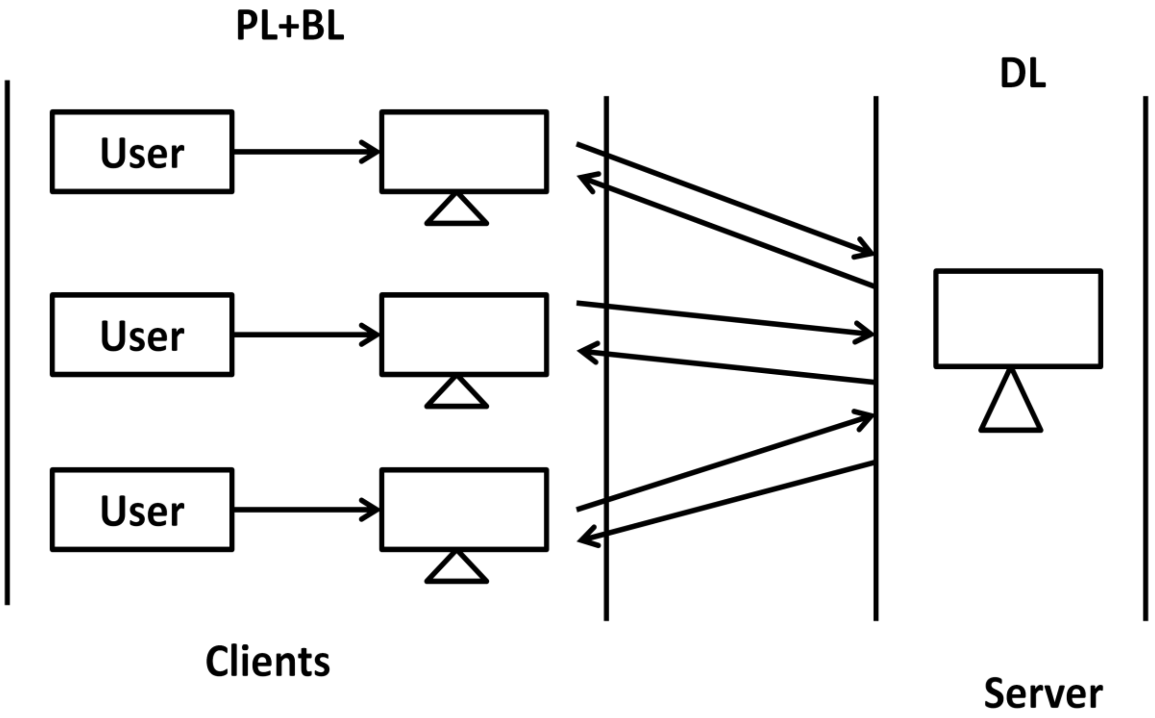


PL= Presentation Logic
BL= Business Logic
Dl= Database Logic

Client/Server Environment:

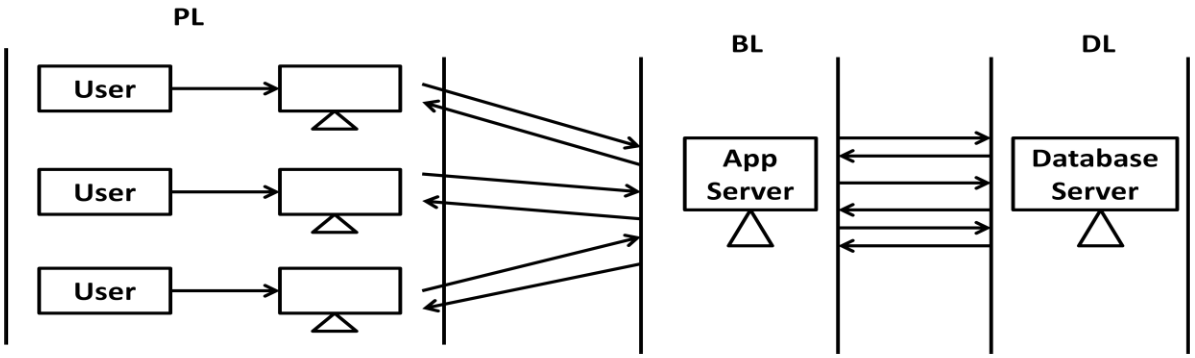
- ➢ In this environment two tiers are present one for clients and another for servers.
- ➢ Presentation and business logic are present in the clients and database logic is present in the database servers.

- If at all if the application is being used by the multiple users in a building (or) in a city and hence there is no problem with security then the recommended environment is 2-tier architecture.

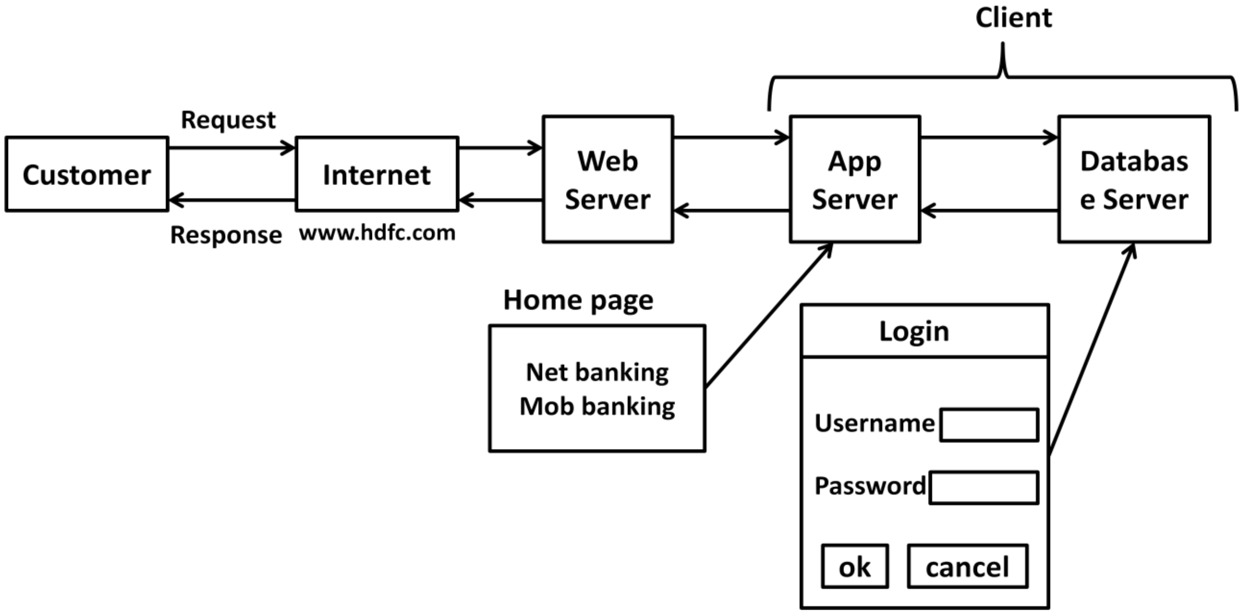


Web Environment:

- In this environment three tiers are present, one for clients, the middle one is for application server and the last one is for database server.
- Presentation logic is present in the clients, business logic is present in the application server and database logic is present in the database server.
- If at all if the application is being used by the limited number of customers all over the world then the recommended environment is 3-tier architecture.



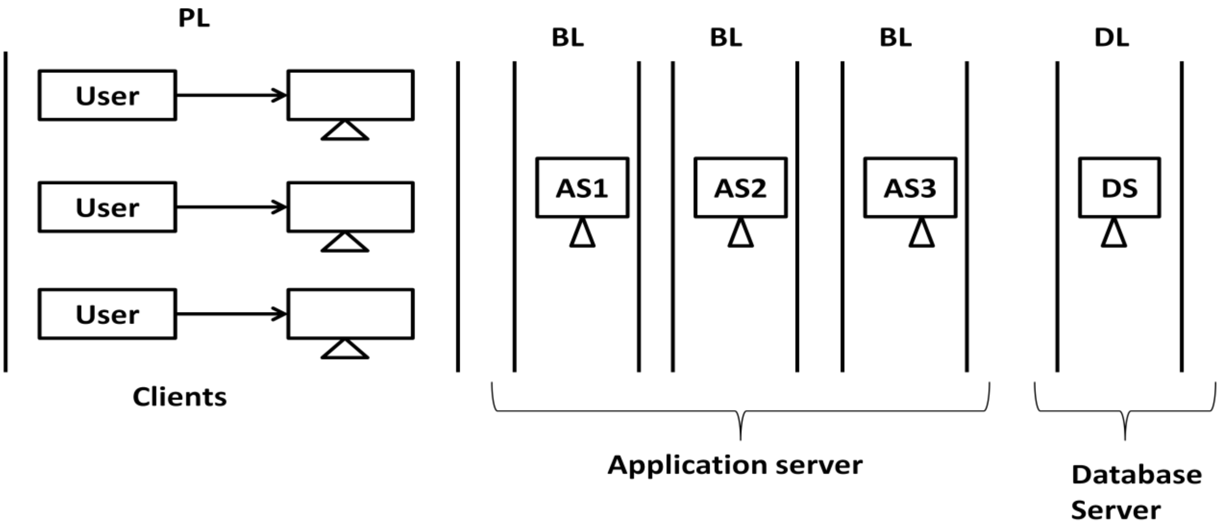
Eg: Travel and tourism application, banking application etc.



Distributes Environment:

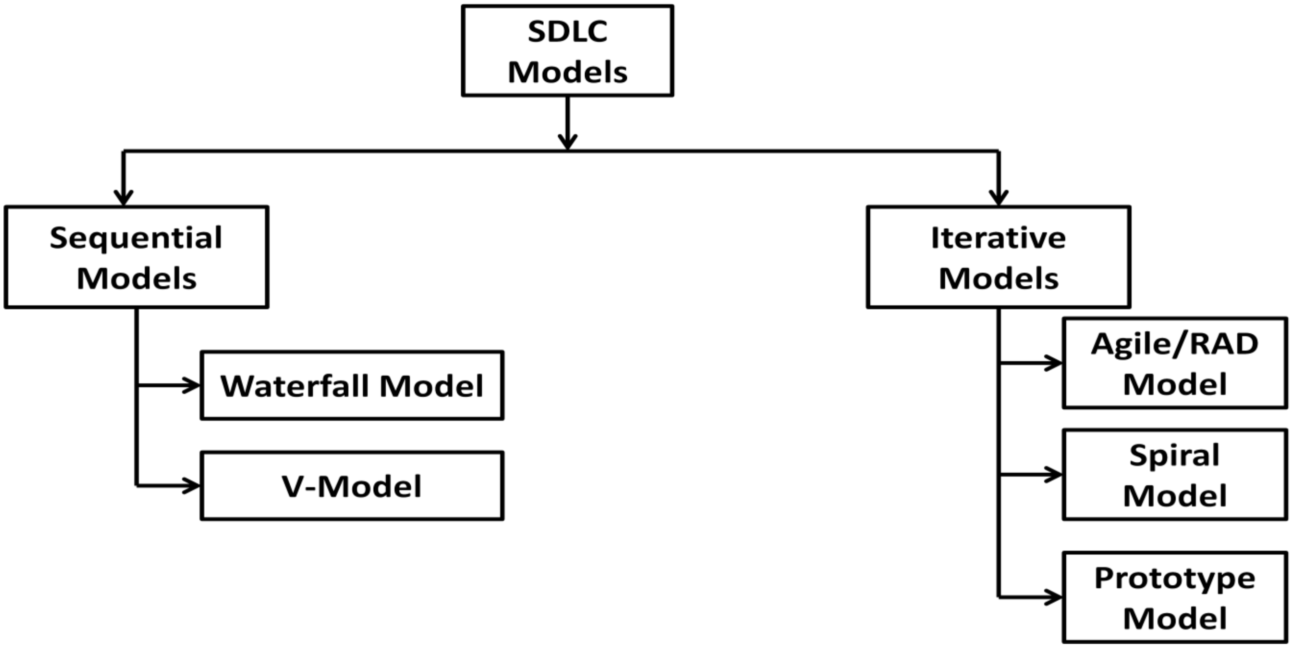
- This environment is also similar to web environment but the number of application servers has been introduced to improve the performance of the application.
- If at all if the application is being used by the unlimited number of customers all over the world then the recommended environment is N-tier architecture.

Eg: social networking application, live streaming of matches etc.

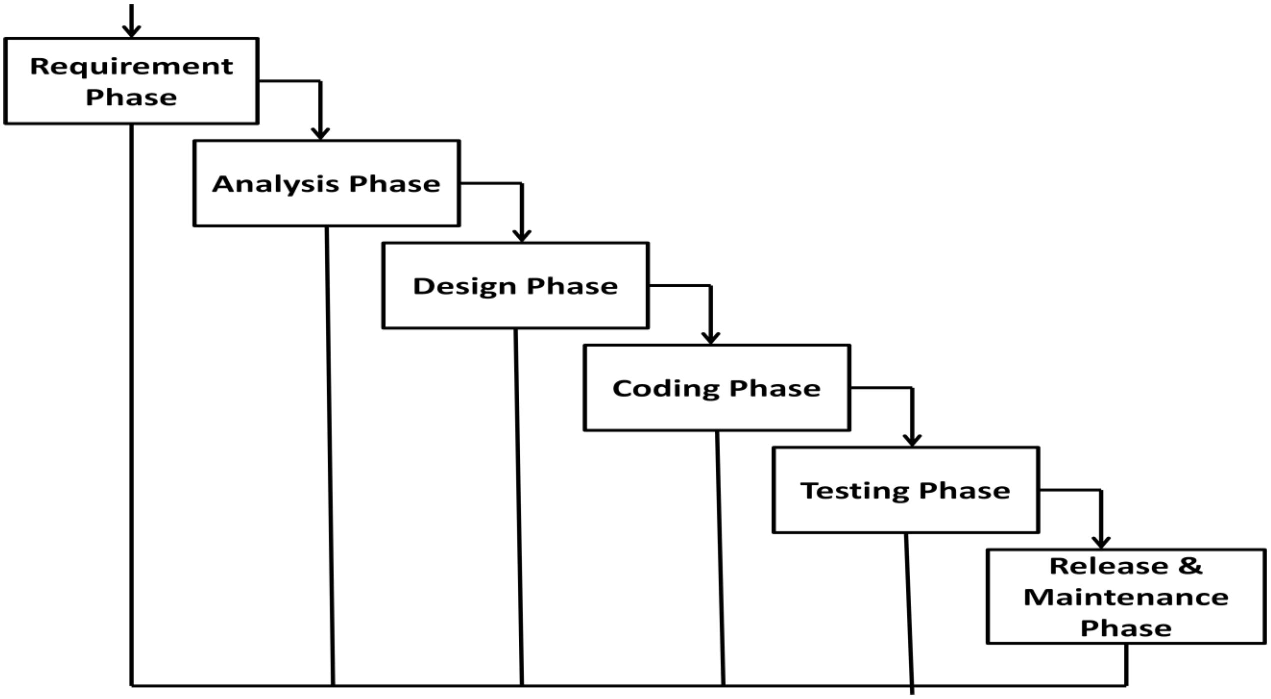


SDLC MODELS

- Following are the SDLC models
  - Sequential models
  - Iterative models



Waterfall Model:

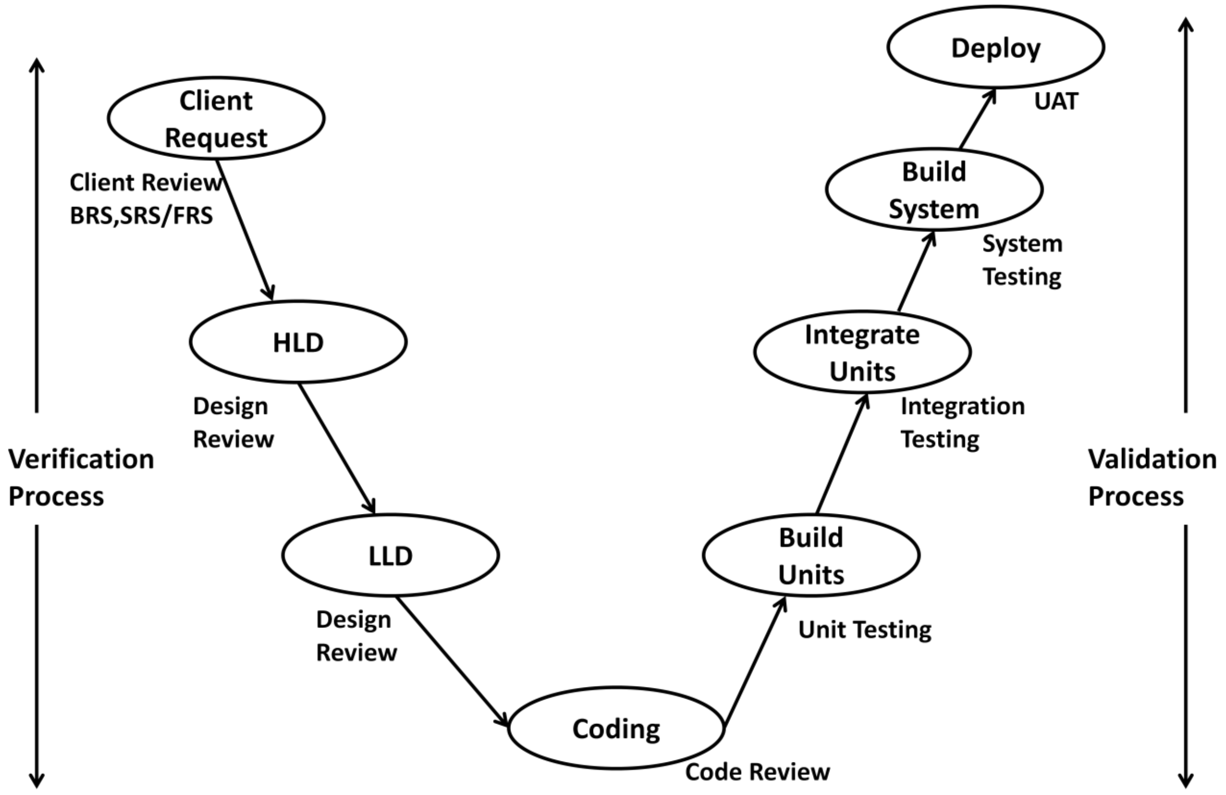


- This model is prepared for smaller applications.
- In this model all the requirements must be base lined (fixed).
- In this model all the SDLC activities will be carried out one after the other.
- As the requirements are very clear in this model only validation process will be carried.

Drawbacks:

- This model never accepts the new requirements i.e request changes in the middle of the project.
- This model is a time consuming model to implement the project.

V-Model:



- In v-model “v” stands for verification and validation.
- V-model has been introduced to eliminate the drawbacks of waterfall model.
- V-model is also applicable for smaller applications.
- In v-model both verification and validation process will be carried out.

- Hence the verification process will be carried out through the help of reviews and inspections, whereas validation process will be carried out through white box, black box and grey box testing.
- This model can also be called as parallel model as all the SDLC activities will be carried out parallely.

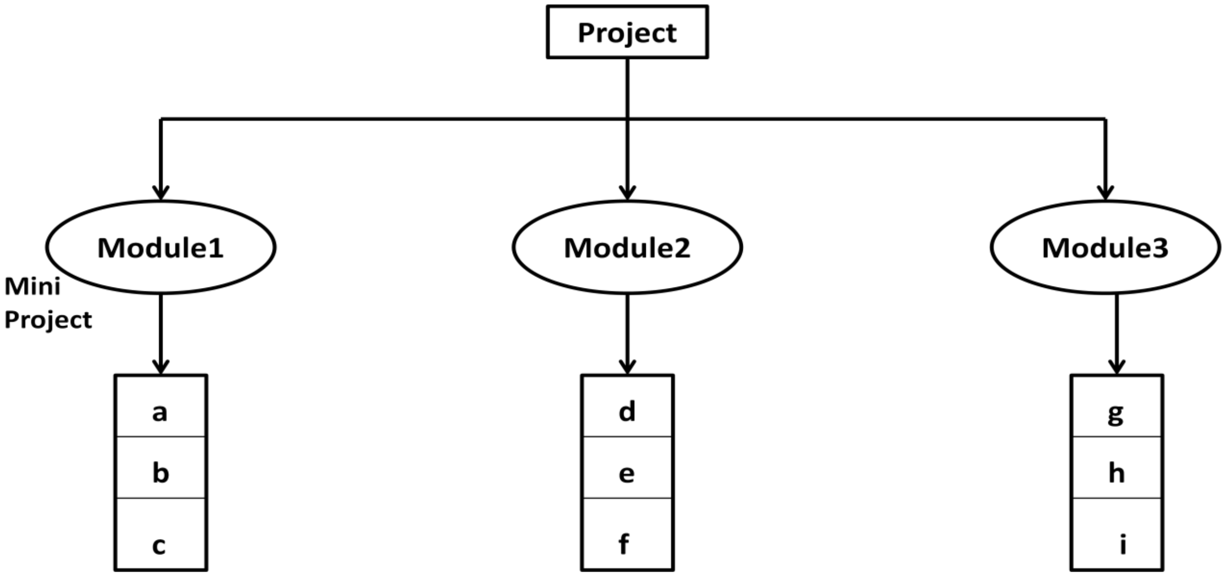
Advantages:

- This model accepts the requirement changes i.e request changes in the middle of the project.
- As the activities will be carried out parallely v-model consumes less time to complete the work i.e less time consuming model.

Q) Should I prefer waterfall model (or) v-model to test my project?

Answer: From the above two models we can conclude that based on the clarity of requirements (constant (or) dynamic) and also based on the timing constraints we can prefer either waterfall model (or) v-model to test a project.

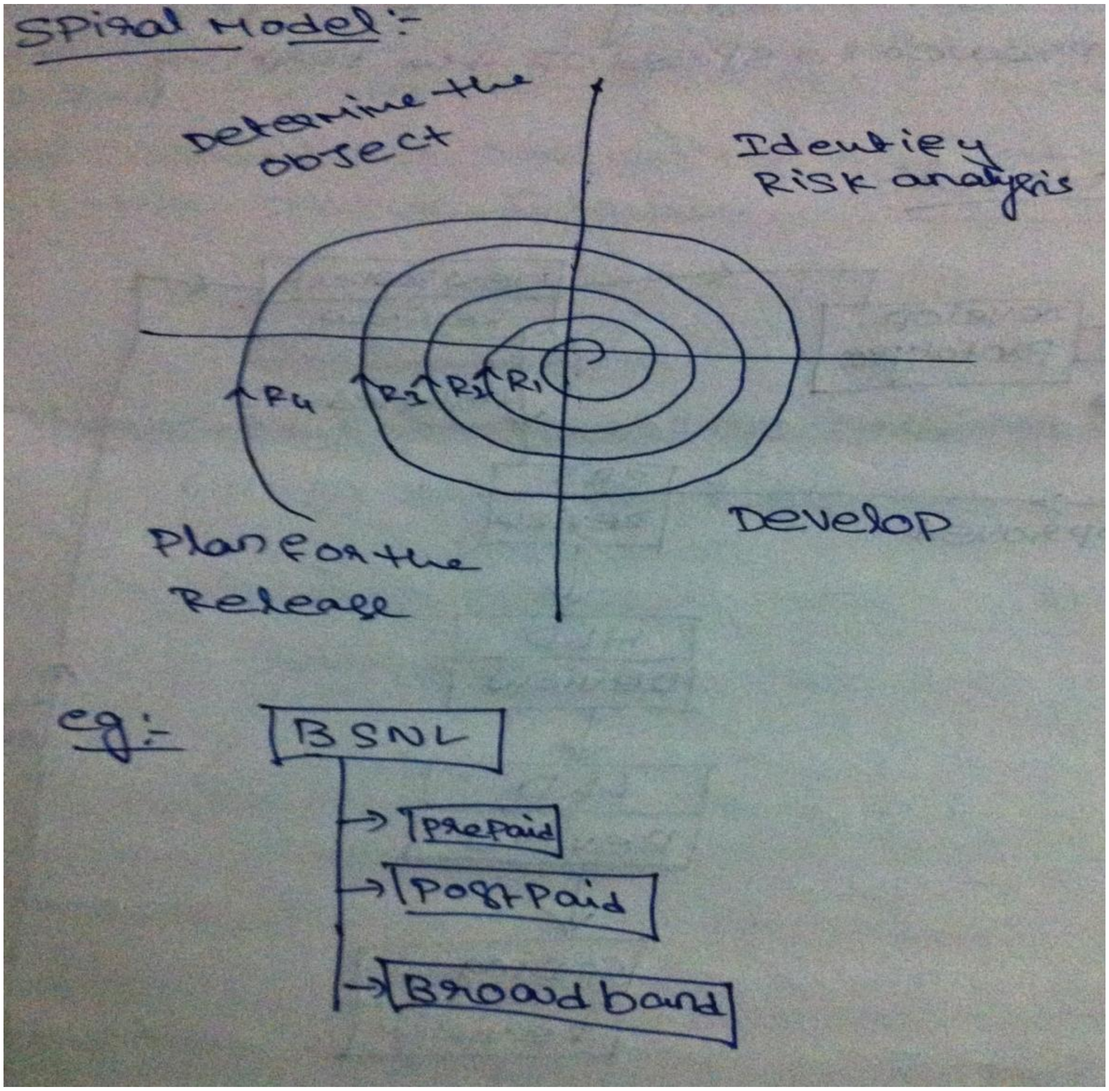
Agile/RAD Model:



- RAD= Rapid Application Development.
- This model is preferred if there is a short time to implement a project.

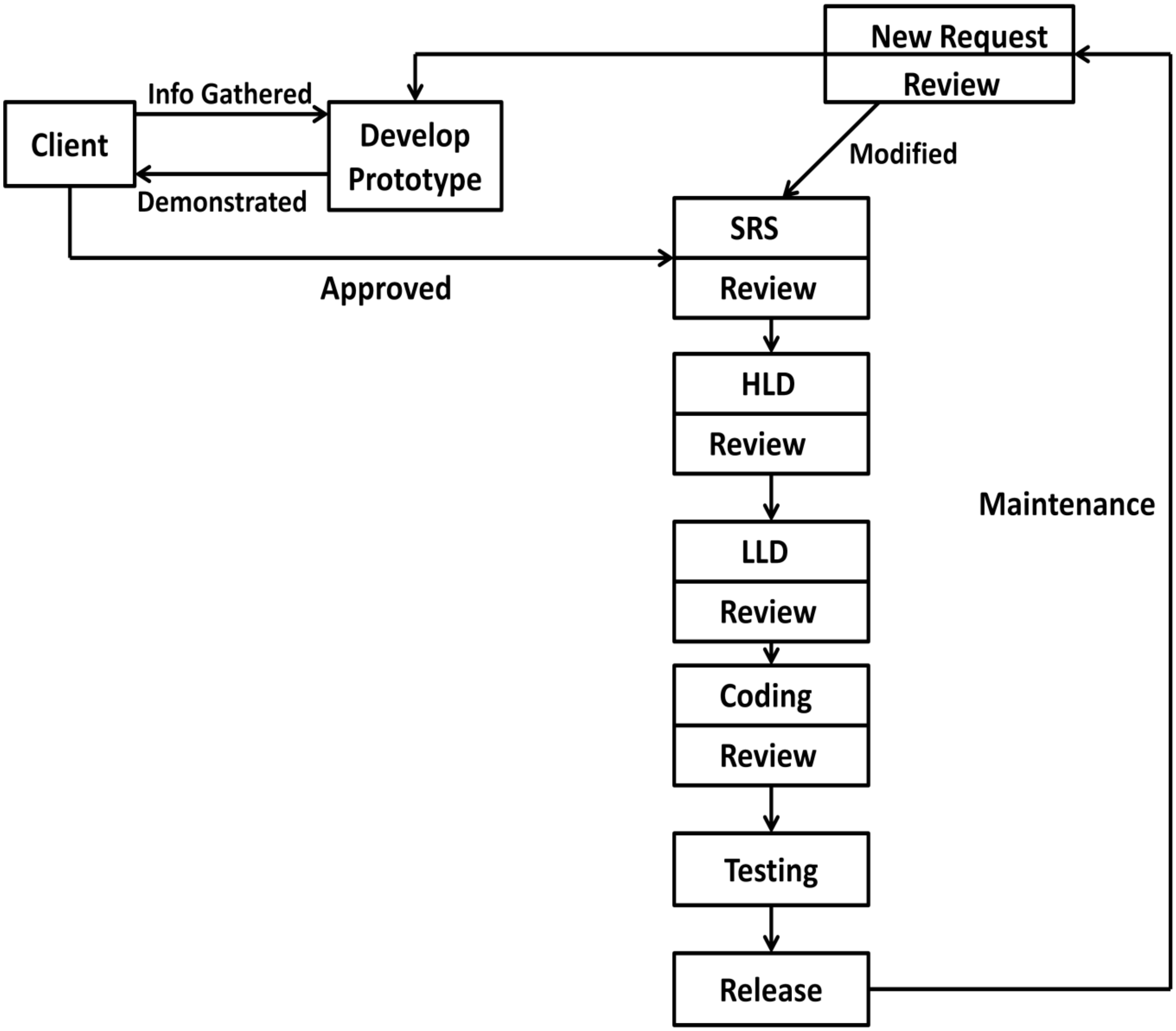
- In this model project will be divided into modules/sprints and hence each module will be considered as a mini project and then separate teams will be allocated to implement the modules. Hence these teams will take care to implement their modules simultaneously.
- While implementing their modules based on the clarity of the requirements these teams can prefer either waterfall model (or) v-model to implement their modules.
- After successful completion of all the modules in a single release application will be delivered to the customer.
- As the modules are getting implemented so quickly within the short period of time this model can also be called as RAD model.

Spiral Model:



- Spiral model is best suitable for maintenance of the project.
- In this model at first a basic system will be implemented and delivered to the customer.
- During project maintenance the new functionalities requested by customer will be implemented one by one.
- Whenever a new functionality is added (or) an existing functionality is modified all the core functionalities in the system needs to be retested.
- Hence automation testing is the best suitable approach to speed up the testing process.

Prototype Model:



- This model is preferred when the client requirements are not clear.
- In this model based on the information gathered from the client a dummy application will be developed and this will be demonstrated to the client for the approval.
- Based on the client approval information SRS document will be prepared and reviewed and hence the remaining all SDLC activities will be carried out one after the other as shown in the above diagram.

LEVELS OF TESTING

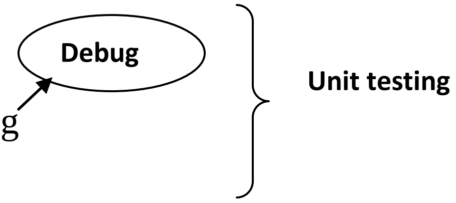
- There are four levels of testing's are there
  - Unit level testing
  - Integration testing
  - System testing
  - User Acceptance Testing (UAT)

UNIT LEVEL TESTING:

- Unit is nothing but the smallest part (or) component.
- Testing conducted by the developers on the units to check whether the units are working fine (or) not is called as unit testing.
- Unit testing can also be called as module (or) component testing.
- In simple terms unit testing is nothing but "**debugging process**".

Eg: c unit

```
// to add two numbers

# include <stdio.h>

Void main ()

{

Int a,b,c;

a=10;b=20;

c=a+b     -------- semicolon is missing

printf("%d",c);

}
```



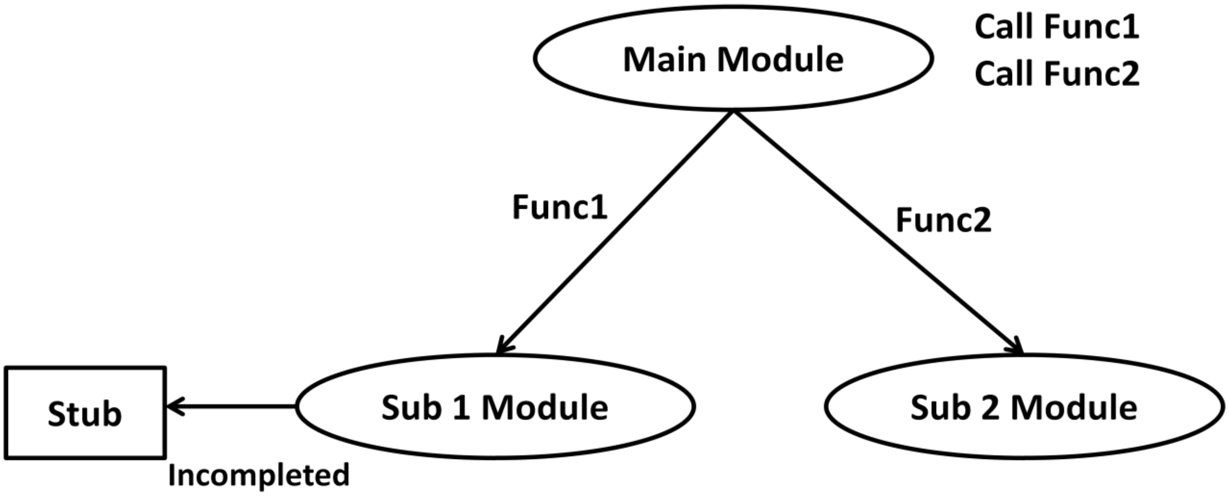
INTEGRATION TESTING:

- It is conducted by developers.
- Testing conducted by the developers, by combining the units and checking the interactions (end to end) is called as integration testing.

- While integrating the units developers can follow any one of the following approaches
  - Top down approach
  - Bottom up approach
  - Hybrid approach

Top Down Approach:

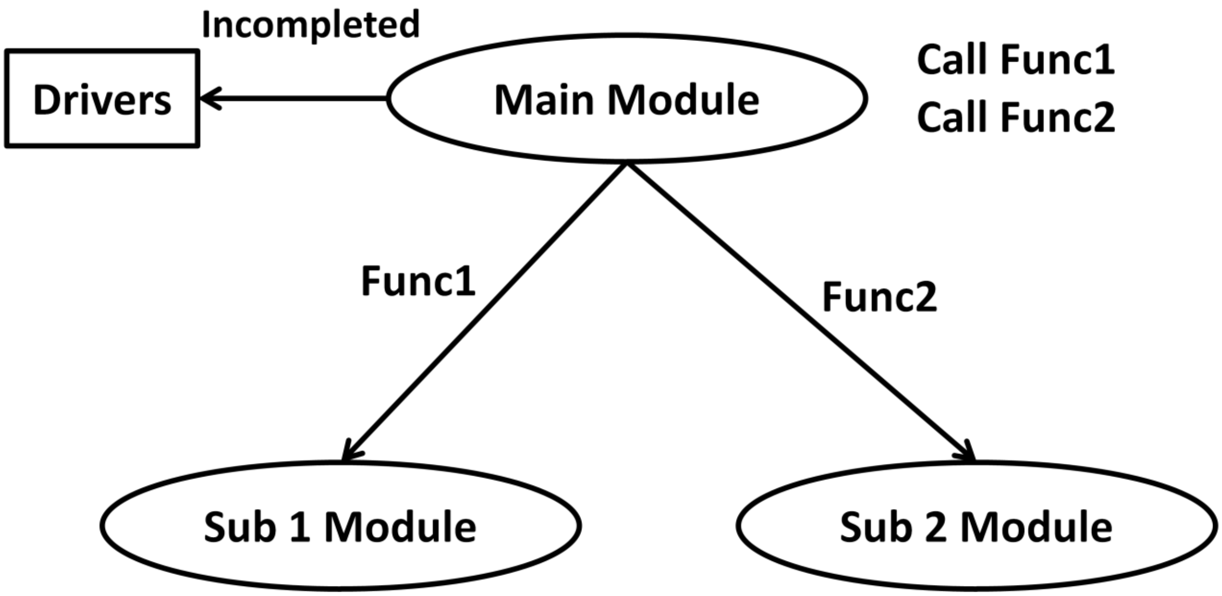
- This approach is preferred when there are in completed units at the bottom level.
- The in completed units at the bottom level will be replaced with "**stubs**".



Stub: A simulated (or) dummy program that replaces the in completed units at the bottom level is called as stub.

Bottom Up Approach:

- This approach is preferred when there are in completed units at the top level.
- The in completed units at the top level are replaced with "**drivers**".



Driver: A simulated (or) dummy program that replaces the in completed unit at the top level is called as driver.

Eg:

Call login (“apple”, ”mercury”) → **Driver**

Public function login (username, password)

Msgbox username

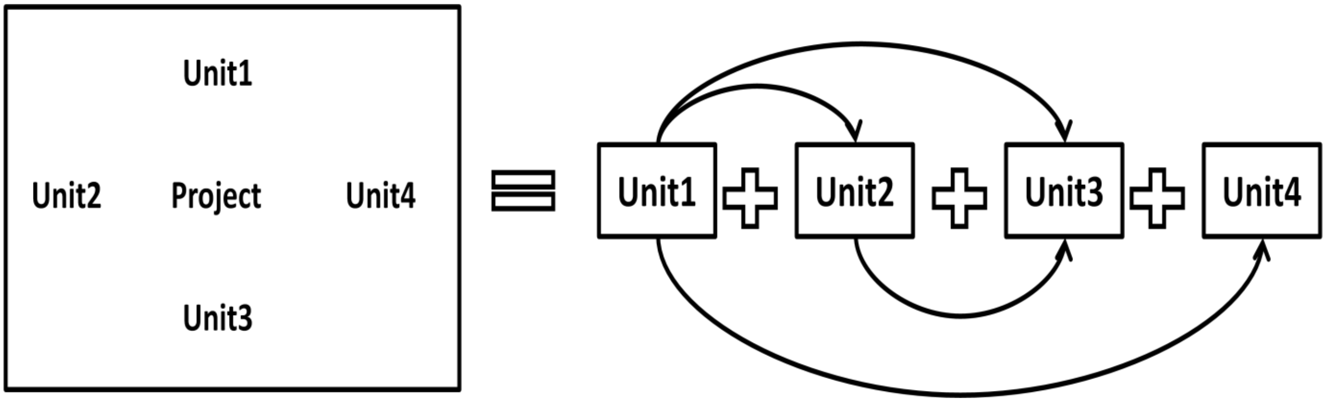
Msgbox password

End function

Stub

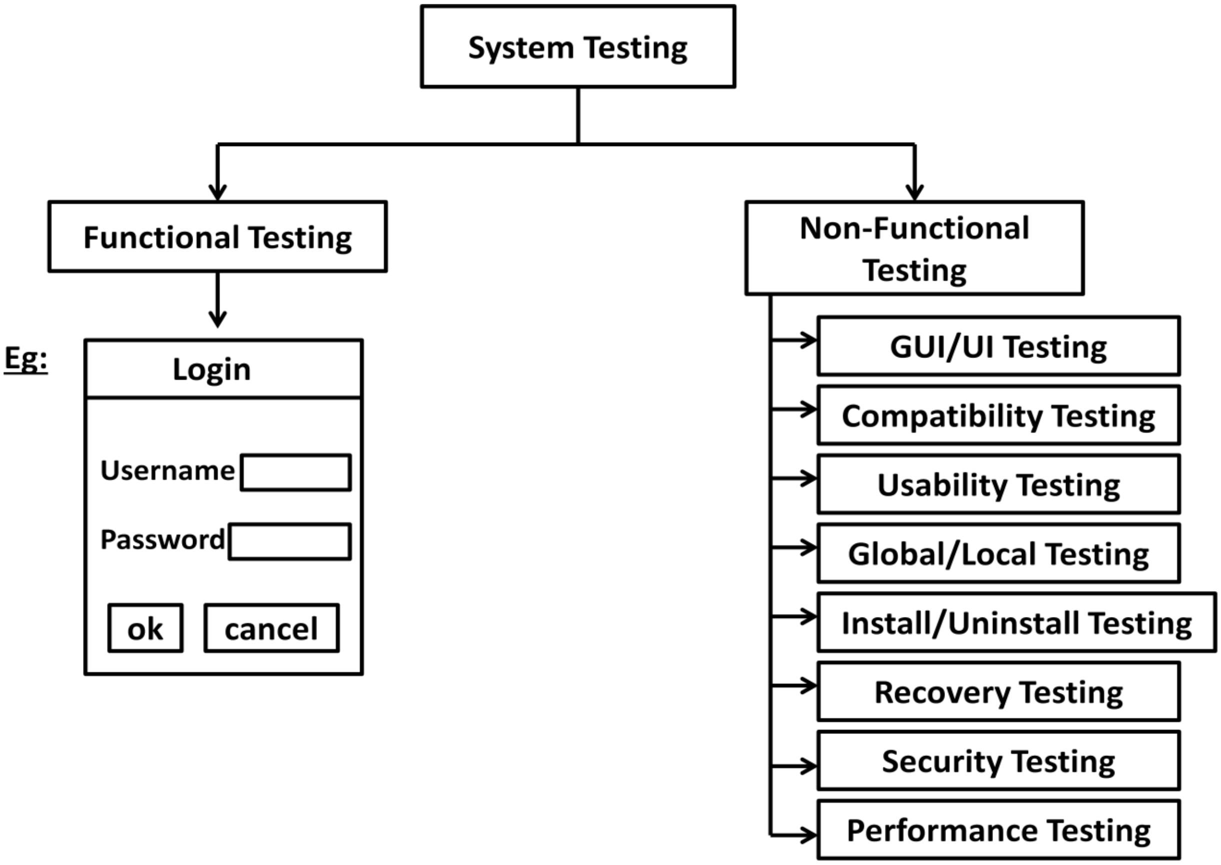
Hybrid Approach:

- This approach is preferred when all the units are available in the project i.e combining all the units and checking all the interactions (end to end) is called as hybrid approach.



SYSTEM TESTING:

- Testing conducted by the test engineers on the application to check whether the application is working as per the functional specifications (or) not is called as system testing.
- System testing further classified into two categories
  - Functional testing
  - Non-functional testing



Functional Testing:

- Checking whether all the functionalities in the system working fine (or) not as per the specifications is called as functional testing.
- Hence functional testing includes following types of testing
  - **Smoke/sanity test:**

    Smoke test:
    - It is conducted by developers.
    - The objective of smoke test is to check whether all the features are available (or) not in the system before releasing the build to the testing department.
    - Hence smoke test is a rough test which can be done in a span of 20 to 30 minutes.

    Sanity test:
    - It is the first level of testing which is conducted at the testing environment by the test engineers.
    - The objective of sanity test is to check whether all features (or) modules available (or) not for the further level of testing.
    - Hence sanity test is deep.

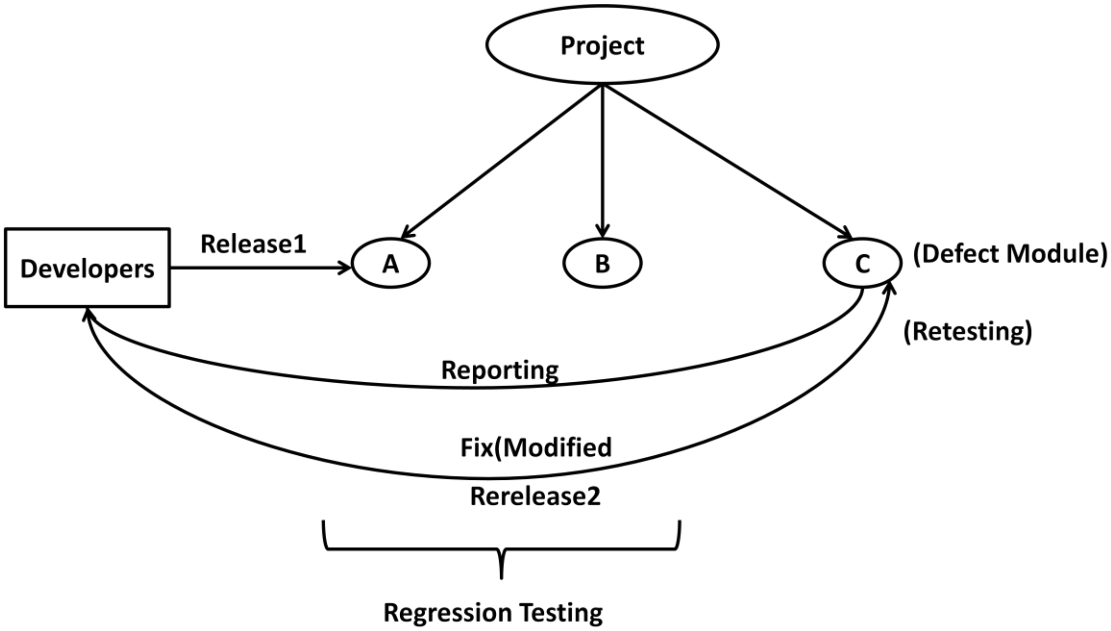
    NOTE: The objective of smoke and sanity test is almost same to check whether all the features are available (or) not in the system.
  - **Priority based testing:**
    - It is a type of testing i.e identifying all business scenarios to be validated pre-other to the execution i.e deciding what to be first and what to be tested last and executing in the same order.
    - Priority based testing is also called as risk based testing.
    - Hence the priority can be as follows

ORG1	ORG2
Very high	P0
High	P1
Medium	P2
Low	P3

- **<u>Re-testing:</u>**
    - ➢ Testing conducted by testers on the defect functionalities repeatedly i.e again and again till the defect functionality get passed is called as re-testing.
- **<u>Regression testing:</u>**
    - ➢ Regression testing can be conducted in two scenarios

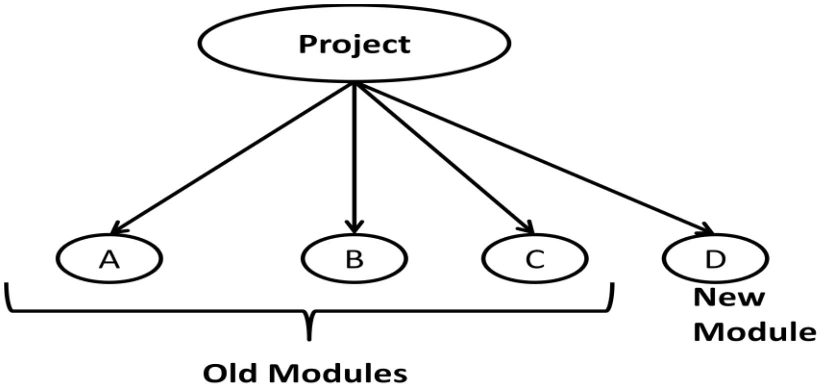
<u>Scenario1:</u> On the modified build

- ➢ Checking whether the modification has given any impact in the interconnected features (or) not is called as regression testing.



<u>Scenario2:</u> On the enhanced build

- ➢ Checking whether the new enhancements have introduced any side effects in the interconnected features (or) not is called as regression testing.

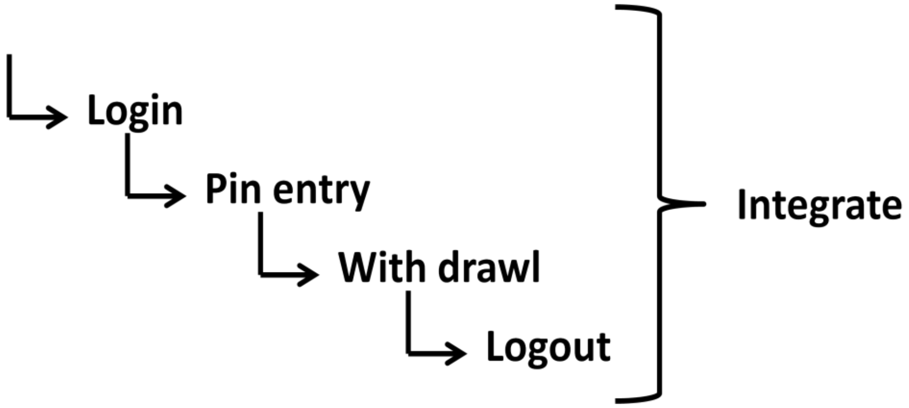


NOTE: The objective of regression testing is to check whether the bug fixers (developers), modifications, enhancements has introduced any side effects in the interconnected features (or) not.

NOTE: Automation testing is the nest suitable approach to speed up the testing process.

- **Formal testing:**
  - ➢ If we test an application by following all predefined procedures and proper documents then this type of testing is called as formal testing.
- **Informal testing:**
  - ➢ If we test an application as we wish i.e without following any procedures and documents then this type of testing is called as informal testing.
  - ➢ Informal testing can also be called as Adhoc/rattle/uneven/zigzag/random/monkey/gorilla testing.
  - ➢ Generally this testing is recommended by senior experience testers in the organization.
  - ➢ Due to lack of time also this testing can be preferred.
- **Exploratory testing:**
  - ➢ It is nothing but exploring the application i.e adding, modifying the existing test vases for the better testing is called as exploratory testing.
- **End-to-end testing:**
  - ➢ It is a process of identifying all core functionalities in the system and testing these functionalities right from one end to the other end including data integration is called as end-to-end testing.
  - ➢ Generally this testing is recommended by domain experts.
- **System integration testing:**
  - ➢ It is a type of testing combining the functionalities and checking the interactions among the functionalities right from one end to the other end is called as system integration testing.

Eg: Business Scenario: Cash with drawl



<u>Non-functional testing:</u>

- ➢ Checking the non-functional specifications in the application as per client requirements is called as non-functional testing.
- ➢ Hence non-functional testing includes following types of tests
  - **<u>GUI/UI testing:</u>**
    - ➢ Checking does all user interfaces has properly built in the system (or) not is called as GUI testing.

<u>Checklist:</u>

- ➢ Check whether all the objects (buttons, edit boxes, text boxes, radio buttons, checkboxes etc) are available (or) not.
- ➢ Check whether the mandatory field’s is highlighted with the star mark (or) not.
- ➢ Check whether the cursor is navigating to all field’s properly (or) not.
- ➢ Verify the alignments, borders, colors in the application.
- ➢ Verify the style uniformness, fonts, alignments in the application.
- ➢ Verify the spelling mistakes throughout the application.
- ➢ Check whether the page is properly aligned to the desktop (or) not.
- ➢ Verify minimize, maximize and close operations.
- ➢ Check whether all the links are properly displaying (or) not.
- ➢ Verify the logos and images in the application.
- ➢ Verify the various browser navigations like forward, backward, refresh etc.
- ➢ Verify the keyboard operations.

- Verify the mouse operations.
- Verify whether all the web elements are present (or) not.

- **Compatibility testing:**
  - Checking whether the product is compatible with different software and hardware configurations (or) not is called as compatibility testing.

Checklist:

- Check whether the application is compatible (or) not with different operating systems like windows family, Linux, Macintosh, Sun, Hp, Android, IOS, Symbian etc.
- Check the application with different hardware's like processors, rams, server's etc.
- Check whether the application is compatible with different browsers (or) not i.e opera, IE, Firefox, safari, chrome etc.

NOTE: Checking the application with multiple browsers compatibility is also called as "**cross browser testing**".

- **Usability testing:**
  - Checking the user friendliness of the application is called as usability testing.

Check list:

- Check whether the application is understandable by any kind of user's (or) not.
- Check whether the application is operatable by any kind of user (or) not.

- **Globalization testing:**
  - Checking does the system is having a provision of changing languages, currency, date, time if the application has been designed for global users.
- **Localization testing:**
  - Checking does the system is having a provision of changing languages, currency, date, time if the application has been designed for local users.

- **Installation testing:**
  - Checking does the software is successfully installing (or) not as per the guidelines document i.e readme file document (or) not is called as installation testing.
- **Un-installation testing:**
  - Checking does the software is successfully uninstalling (or) not from the system is called as un-installation testing.
- **Recovery testing:**
  - Checking how the system is handling the unexpected events like power failures, crashes etc is called as recovery testing.

Checklist:

  - Check does the system is having a provision of backup (or) not.
  - Check the lifespan of the backup.
  - Check when the application crashes by restarting the system whether the data is recovering (or) not.
  - Check whether the data losses are there (or) not.

- **Security testing:**
  - Checking does all the security conditions has properly built in the system (or) not is called as security testing.

Checklist:

  - Check for authorization.

    Authorization: Checking does the system is having a provision of creating login accounts, passwords, changing settings etc is called as authorization.
  - Check for authentication.

    Authentication: Checking does the system is able to recognize the valid users (or) not and displaying the right information to right user (or) not is called as authentication.
  - Check for firewall leakage.

    Firewall leakage: It is type of testing where users will enter into the application as one level of user and try to access the application beyond the limits. Hence checking when the user is trying to do the activities beyond the limits whether the firewall is blocking (or) not is called as firewall leakage.

- Check whether the URL’s are secured (or) not.
- Check for encryption.
  Encryption: Checking whether the entered password (or) text is getting converted into the coding format (or) not.
- Check for decryption
  Decryption: Checking whether the coding format is getting converted into the text format (or) not.
- Check the session expiry.
- Check the cookies i.e whether the cookies are automatically deleting (or) not.
- Check whether the passwords are provided with virtual keyboards (or) not.
- Check whether the alerts are displaying properly (or) not (mail, phone etc).
- Check whether the accounts are blocking (or) not with the wrong entries.
- Check whether the hint messages are displaying (or) not.
- Check whether the security codes are displaying (or) not.
- Verify the browser navigations with forward and backward i.e pages are secured (or) not.
- Check whether the images are provided (or) not.
- Check whether the system is protected with audio and videos (or) not.

- **Performance testing:**
  - It is a process of analyzing various efficiency factors such as throughput, response time, tuning, bench marking, capacity planning, load, stress, volume, server’s, product evaluation etc. is called as performance testing.

    (Or)

    The main objective of performance testing is to pinpoint the bottlenecks in the application.

    (Or)

    It is a process of determining the effectiveness (or) speed of the system (or) product is called as performance testing.

USER ACCEPTANCE TESTING (UAT):

- Testing conducted on the completed application by the domain experts (or) by end-users is called as UAT.
- UAT can be conducted in two stages
  - Alpha testing
  - Beta testing

Alpha testing: It is a first level of UAT which is conducted at the office premises by the domain experts to confirm whether the application is ready for live production (or) not.

Beta testing: It is the last level of UAT which is conducted at the client premises by the end-users i.e product will be given to the customer and these will test the product.

TEST DESIGN TECHNIQUES

- Two types of test design techniques are there
  - White box test design techniques
  - Black box test design techniques

WHITE BOX TEST DESIGN TECHNIQUES:

- From the testing principles we understood that exhaustive testing is impossible.
- Hence in order to ensure 100% **code coverage** developers are opting for white box test design techniques.

Q) What is code?

Answer: A code is a set of statements, branches, paths etc is called as code.

Code coverage: The percentage of code tested during white box testing is called as code as code coverage.

- Following are the techniques that can be used to calculate the code coverage

$$\textbf{Statement coverage} = \frac{\textbf{No. of statements executed}}{\textbf{Total no. of statements}} \times \textbf{100\%}$$

$$\textbf{Branch/Decision coverage} = \frac{\textbf{No. of branches/decisions executed}}{\textbf{Total no. of branches/decisions}} \times \textbf{100\%}$$

$$\textbf{Path coverage} = \frac{\textbf{No. of paths executed}}{\textbf{Total no. of paths}} \times \textbf{100\%}$$

NOTE: 100% LCSAJ (Linear Code Sequence and Jump) will automatically ensure 100% branch/decision coverage.

NOTE: 100% branch/decision coverage will automatically ensure 100% statement coverage.

NOTE: 100% path coverage will automatically ensure 100% branch/decision coverage.

NOTE: 100% path coverage will automatically ensure 100% statement coverage.

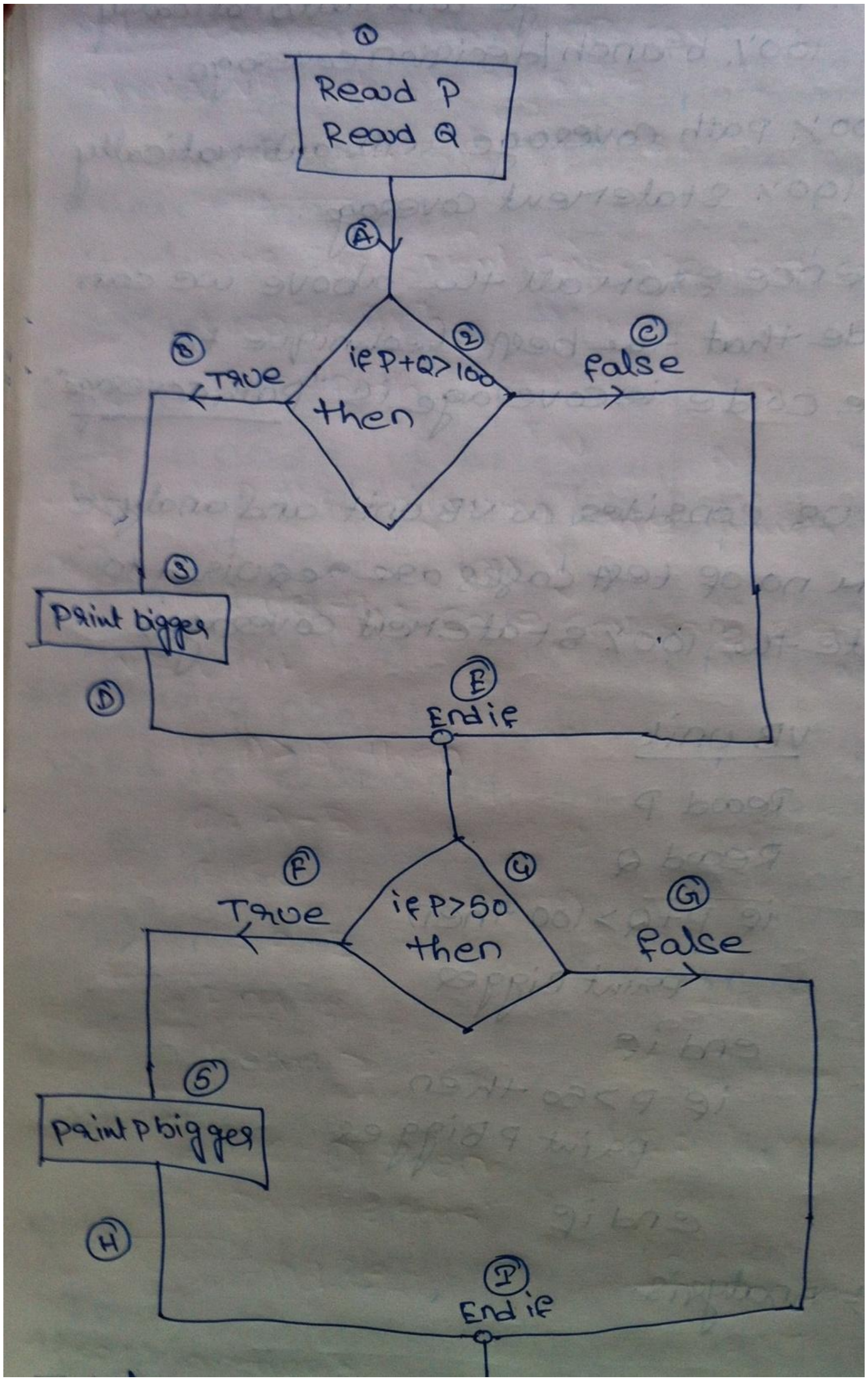
NOTE: Hence from all the above we can conclude that the best technique to calculate code coverage is "**Path Coverage**".

Eg: Let us consider a VB unit and analyze minimum number of test cases required to calculate the 100% statement coverage?

VB unit

```
Read P
Read Q
If P+Q > 100 then
    Print bigger
End if
If P > 50 then
    Print P bigger
End if
```

Step1: Analysis



- Total number of statements: 8
- Total number of branch/decisions: 2

B1→1A, 2B, 3D, E, 4F, 5H, I

B2→1A, 2C, E, 4G, I

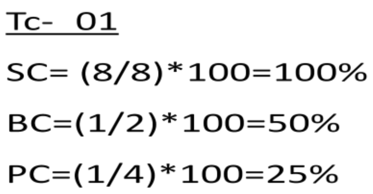
- Number of paths:

  P1→1A, 2B, 3D, E, 4F, 5H, I
  P2→1A, 2B, 3D, E, 4G, I
  P3→1A, 2C, E, 4G, I
  P4→1A, 2C, E, 4F, 5H, I

Step2: Design and execution

		TEST CONDITION			
		P=80,Q=30	**P=30,Q=80**	**P=10,Q=20**	**P=60,Q=20**
No. of statements	**VB unit**	**Tc_01**	**Tc_02**	**Tc_03**	**Tc_04**
1	Read P	Y	Y	Y	Y
2	Read Q	Y	Y	Y	Y
3	If P+Q>100 then	Y	Y	Y	Y
4	Print bigger	Y	Y	N	N
5	End if	Y	Y	Y	Y
6	If P>50 then	Y	Y	Y	Y
7	Print P bigger	Y	N	N	Y
8	End if	Y	Y	Y	Y

Step3: Report generation

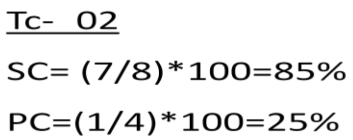


Tc- 01

SC= (8/8)*100=100%

BC=(1/2)*100=50%

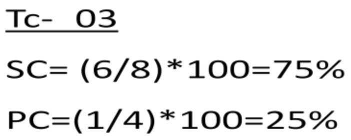
PC=(1/4)*100=25%



Tc- 02

SC= (7/8)*100=85%

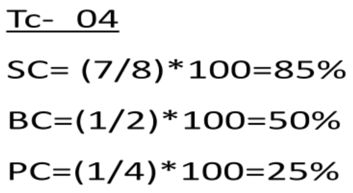
PC=(1/4)*100=25%



Tc- 03

SC= (6/8)*100=75%

PC=(1/4)*100=25%



Tc- 04

SC= (7/8)*100=85%

BC=(1/2)*100=50%

PC=(1/4)*100=25%

<u>NOTE:</u> To ensure 100% code coverage in the above VB unit we require

One statement coverage test case

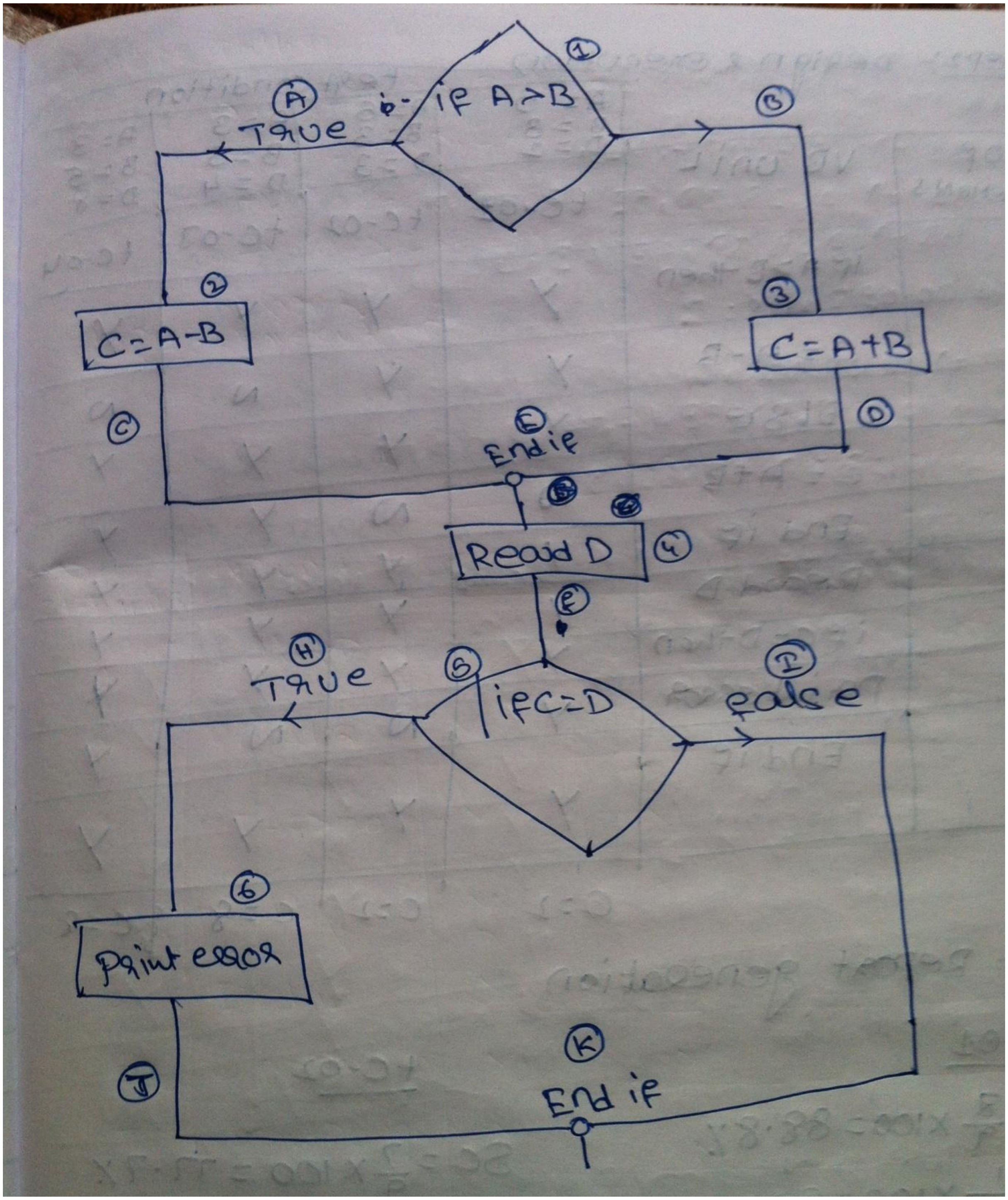
Two branch coverage test cases

Four path coverage test cases

Eg: Calculate the code coverage for the following VB unit

```
If A>B then
C=A-B
Else
C=A+B
End if
Read D
If C=D then
Print “error”
End if
```

Step1: Analysis



- Total number of statements: 9
- Total number of branches: 2

B1→1A, 2C, E, 4F, 5H, 6J, K

B2→1B, 3D, E, 4F, 5I, K

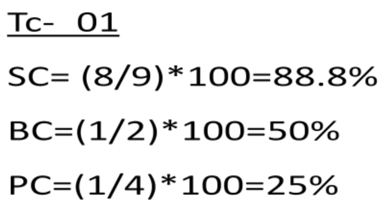
- Total number of paths:

  P1→1A, 2C, E, 4F, 5H, 6J, K
  P2→1A, 2C, E, 4F, 5I, K
  P3→1B, 3D, E, 4F, 5I, K
  P4→1B, 3D, E, 4F, 5H, 6J, K

Step2: Design and execution

		TEST CONDITION			
		A=5 **B=3** **D=2**	**A=5** **B=3** **D=3**	**A=3** **B=5** **D=4**	**A=3** **B=5** **D=8**
No. of statements	**VB unit**	**Tc_01**	**Tc_02**	**Tc_03**	**Tc_04**
1	If A>b then	Y	Y	Y	Y
2	C=A-B	Y	Y	N	N
3	Else	Y	Y	Y	Y
4	C=A+B	N	N	Y	Y
5	End if	Y	Y	Y	Y
6	Read D	Y	Y	Y	Y
7	If C=D then	Y	Y	Y	Y
8	Print error	Y	N	N	Y
9	End if	Y	Y	Y	Y

Step3: Report generation

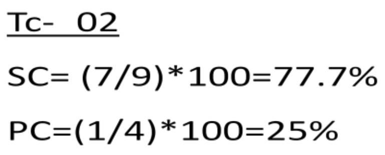


Tc- 01

SC= (8/9)*100=88.8%

BC=(1/2)*100=50%

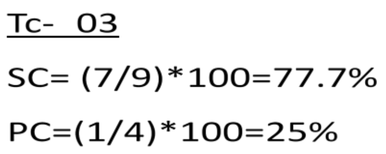
PC=(1/4)*100=25%



Tc- 02

SC= (7/9)*100=77.7%

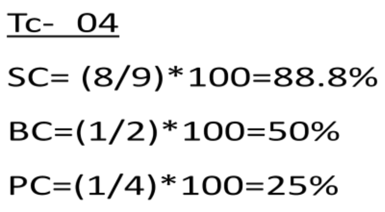
PC=(1/4)*100=25%



Tc- 03

SC= (7/9)*100=77.7%

PC=(1/4)*100=25%



Tc- 04

SC= (8/9)*100=88.8%

BC=(1/2)*100=50%

PC=(1/4)*100=25%

NOTE: To ensure 100% code coverage in the above VB unit we require

Two statement coverage test case

Two branch coverage test cases

Four path coverage test cases

Eg: Consider the following

Pick up and read the newspaper.

Look at what is on television.

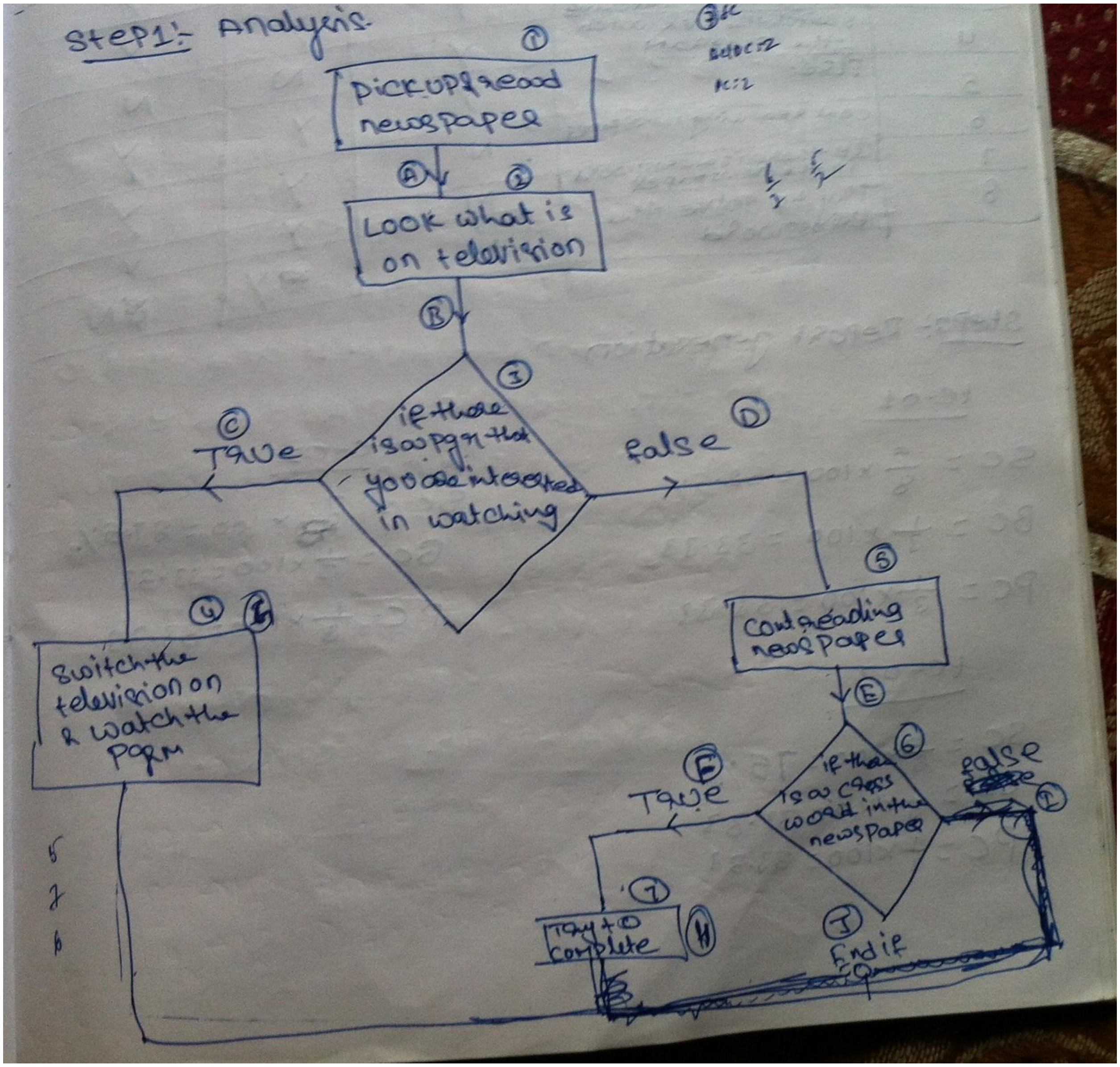
If there is a program that you are interested in watching then switch the television on and watch program.

Otherwise

Continue reading the newspaper.

If there is a crossword in the newspaper then try and complete the crossword.

Step1: Analysis


Step1:- Analysis
1
Pickup & read newspaper
A
2
Look what is on television
B
3
if there is a pgm that you are interested in watching
C
True
false
D
4
G
switch the television on & watch the pgm
5
Cont reading newspaper
E
6
if there is a cross word in the newspaper
F
True
false
I
7
Try to complete
H
J
End if

- Total number of statements: 8
- Total number of branches: 3

  B1→1A, 2B, 3C, 4G

  B2→1A, 2B, 3D, 5E, 6F, 7H, J

  B3→1A, 2B, 3D, 5E, 6I, J
- Total number of paths:

  P1→1A, 2B, 3C, 4G

  P2→1A, 2B, 3D, 5E, 6F, 7H, J

  P3→1A, 2B, 3D, 5E, 6I, J

Step2: Design and execution

No. of statements	VB statements	Tc_01	Tc_02	Tc_03
1	Pickup and read newspaper	Y	Y	Y
2	Look at what is on the TV	Y	Y	Y
3	If there is a program that you are interested in watching	Y	Y	Y
4	Switch the TV & watch the program	Y	N	N
5	Else	Y	Y	Y
6	Continue reading newspaper	N	Y	Y
7	If there is a crossword in the newspaper	N	Y	Y
8	Try to solve & complete the crossword	N	Y	N

Step3: Report generation

Tc- 01

SC= (5/8)*100=62.5%

BC=(1/3)*100=33.3%

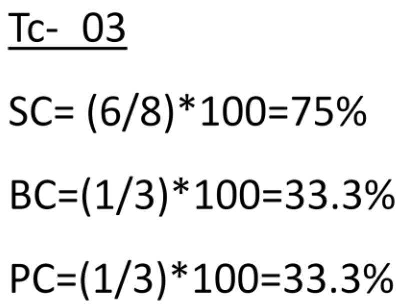
PC=(1/3)*100=33.3%

Tc- 02

SC= (7/8)*100=87.5%

BC=(1/3)*100=33.3%

PC=(1/3)*100=33.3%



Tc- 03

SC= (6/8)*100=75%

BC=(1/3)*100=33.3%

PC=(1/3)*100=33.3%

NOTE: To ensure 100% code coverage in the above VB unit we require

Three statement coverage test case

Three branch coverage test cases

Three path coverage test cases

BLACK BOX TEST DESIGN TECHNIQUES:

- From the testing principle we understood that exhaustive testing is impossible. Hence in order to ensure 100% requirement coverage testers are opting for black box test design techniques.
- Following are the black box test design techniques
    - **Boundary Value Analysis (BVA):**
    - While testing the application is a application is having boundary ranges then the suggested technique is BVA.
    - BVA says that if a scenario is having boundary ranges then concentrate only on the boundary ranges and come to conclusion that whether the application is working (or) not.
    - Hence BVA can be calculated as follows

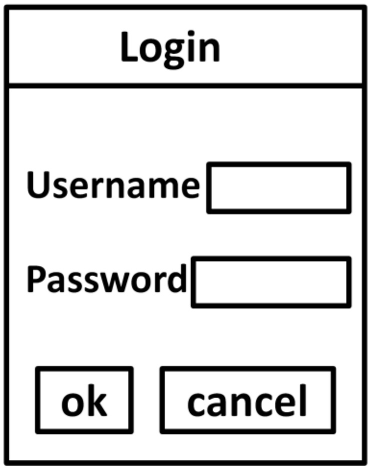
      BVA=LB-1, LB, LB+1
      UB-1, UB, UB+1

      LB= Lower Bound
      UB= upper bound

      Eg: A person 'X' has given a data on a person's age, which should be between 1 to 99. Using BVA which is the appropriate boundary range.

      BVA=0, 1, 2
      98, 99, 100

      Eg: As per the client requirement in a login form the username field must contain minimum four and maximum ten alpha numericals and password field must contain minimum three and maximum six alpha numericals and 'ok' button must be enabled only after username and password is valid and hence cancel button must be enabled at any instant.



<u>Task:</u> verify the username field in a login form

		Expected	
Tc_id	**BVA**	**Valid**	**invalid**
1	LB-1		3 chars/num's/blank spaces/special char
2	LB	4 char/num's	blank spaces/special char
3	LB+1	5 char/num's	blank spaces/special char
4	UB-1	9 char/num's	blank spaces/special char
5	UB	10 char/num's	blank spaces/special char
6	UB+1		11 chars/num's/blank spaces/special char

- **<u>Decision table testing:</u>**

➢ This testing is helpful to identify the defects which may occur because of the error committed while specifying the **logical conditions.**

➢ According to decision table testing when functionality depends on multiple inputs identifying all possible conditions and corresponding expected behavior and covering all possibilities to ensure requirement coverage.

➢ Task: Verify the 'ok' button in a login form

Tc	**Username**	**Password**	**Expected**
Tc_01	Valid	Valid	'ok' button enabled
Tc_02	Valid	Invalid	'ok' button disabled
Tc_03	Invalid	Valid	'ok' button disabled
Tc_04	Invalid	Invalid	'ok' button disabled

- **Equivalence Class Partition (ECP):**

- While testing the application if a application is having bigger boundary ranges then the suggested technique is ECP.
- ECP says that while testing the application if a application is having bigger boundary ranges divide the boundary range into **equal parts** and then concentrate on the **boundary ranges** and come to conclusion that whether the application is working fine (or) not.
- Hence the ECP calculated as follows

LB= Lower Bound
MV= Middle Value
UB= Upper Bound
ECP= LB-1, LB, LB+1
MV
UB-1, UB, UB+1

Eg: In an examination a candidate has to score minimum of 24 marks in order to clear the exam. The maximum that he can score is 40 marks. Hence identify the valid equivalence value if the student clears the exam.

ECP= 23, 24, 25
35
39, 40, 41

Eg: In ATM software application customer can withdraw minimum Rs. 100 and maximum Rs. 40,000 and also the entered amount must be multiples of 100.

Tc_id	BVA	Test condition		Expected
		Valid	invalid	
Tc_01	Lb-1		Rs. 99	Enter amount in multiples of 100
Tc_02	LB	Rs. 100		Cash withdrawn successfully
Tc_03	LB+1		Rs. 101	Enter amount in multiples of 100
Tc_04	MV	Rs. 20,000		Cash withdrawn successfully
Tc_05	UB-1		Rs. 39,999	Enter amount in multiples of 100

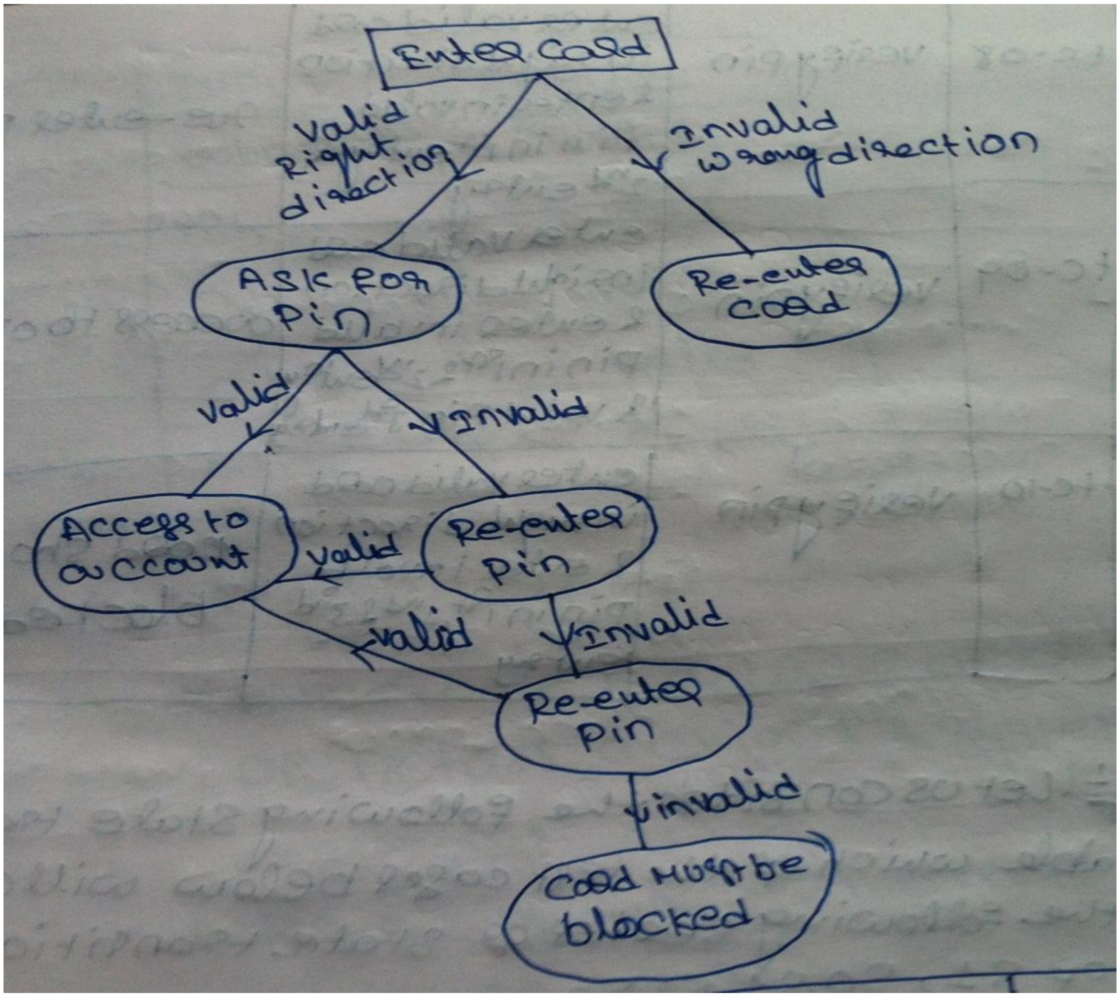
Tc_06	UB	Rs. 40,000		Cash withdrawn successfully
Tc_07	UB+1		Rs. 40,001	Enter amount in multiples of 100

- **State transition testing:**

➢ Every software application will have various states.

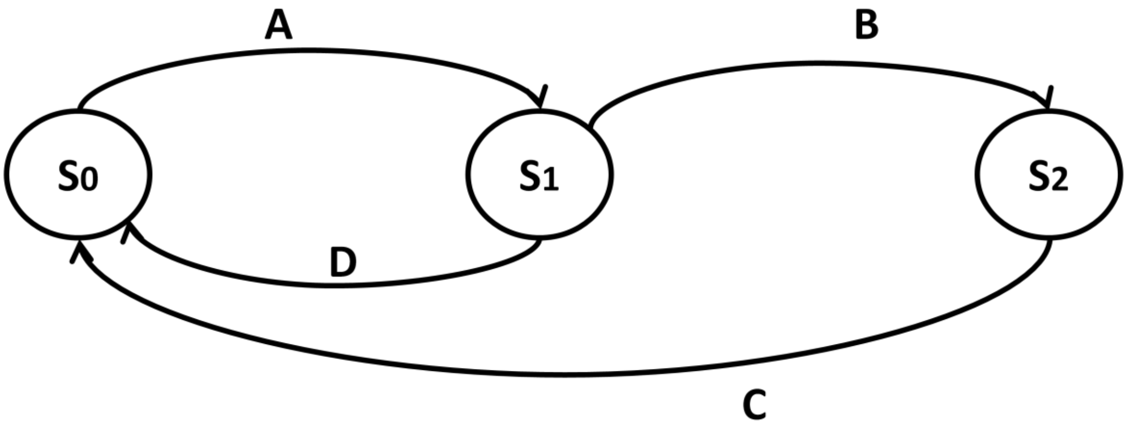
➢ A state of an application will changes from one state to another state based on the operations we perform and based on the input data we supply. Hence in order to check does the application navigating properly (or) not for various possible operations carried out by end-user's, state transition testing has been introduced.

Eg: Let us consider a state transition diagram of customer account access in an ATM software and check whether the application is working as per the state transition (or) not.



Tc-id	Tc-scenario	Tc-action/steps	expected
Tc-01	Verify card insertion	Enter valid card in right direction	Ask for pin
Tc-02	Verify card insertion	Enter valid card in wrong direction	Re-enter card
Tc-03	Verify card insertion	Enter invalid card in right direction	Re-enter card
Tc-04	Verify card insertion	Enter invalid card in wrong direction	Re-enter card
Tc-05	Verify pin	Enter valid card in right direction & enter invalid pin	Re-enter pin
Tc-06	Verify pin	Enter valid card in right direction & enter valid pin	Access to operations
Tc-07	Verify pin	Enter valid card in right direction & enter invalid pin in 1st entry & valid pin in 2nd entry	Access to operations
Tc-08	Verify pin	Enter valid card in right direction & enter invalid pin in 1st & 2nd entry	Re-enter pin
Tc-09	Verify pin	Enter valid card in right direction & enter invalid pin in 1st 2nd entry & valid pin in 3rd entry	Access to operations
Tc-10	Verify pin	Enter valid card in right direction & enter invalid pin in 1st, 2nd, 3rd entry	Card should be blocked

Eg: Let us consider the following state transition table which of the test cases below will cover the following series of state transitions of S0, S1, S2, S0



PATH= DABC

- **Error guessing technique:**

➢ Guessing an error with specific input data and testing the application with that input data is called as error guessing technique.

Eg: Guess the error for the username field in a login form.

Input	expected
abc@	Invalid
ab_	Invalid
abcd	Valid
ab12	Valid
ab#	invalid

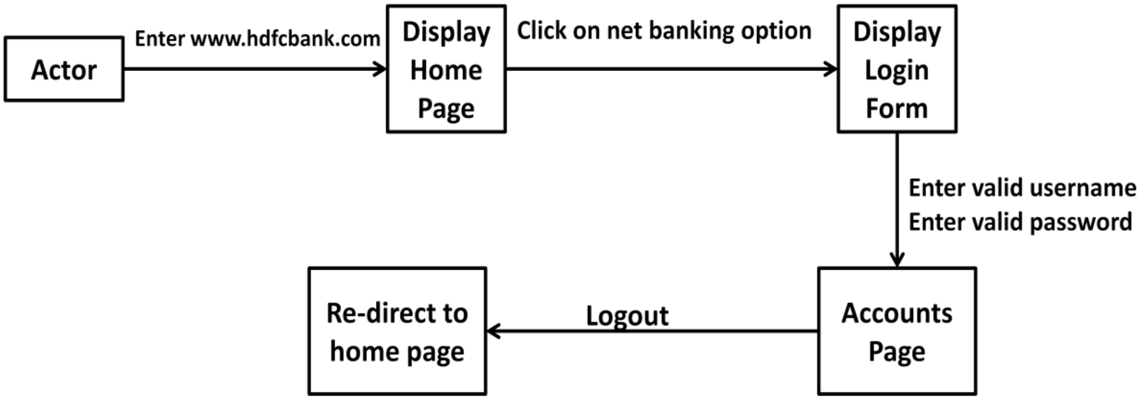
- **Use-case testing:**

➢ A use-case is a brief description of actor's actions and system responses.

➢ For every software application we have two types of actor's one is admin user and the other is normal user.

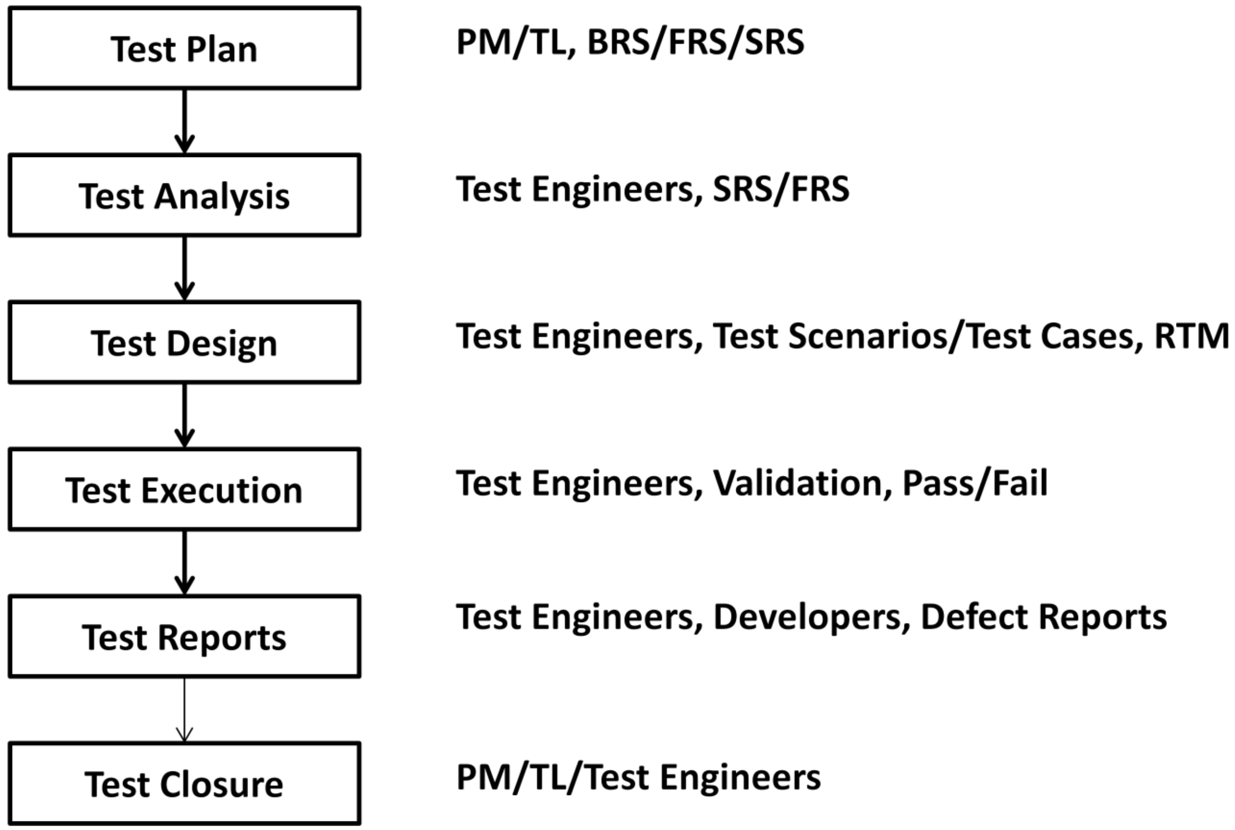
➢ If we prepare the test cases based on the use-cases then it is called as use-case testing.

Eg: Let us consider a use-case diagram of HDFC E-banking system and check whether it is working as per the use-cases (or) not.



Tc-id	Tc-scenario	Actors actions	System responses
tc-01	To verify the URL	1.enter as www.hdfc.com	Displays the home page
Tc-02	To verify the login form	Click net banking	Display the login page
Tc-03	To verify successful customer login	1. Enter valid CID 2. Enter valid customer name 3. Enter valid password 4. Click submit	Access to accounts page
Tc-04	Logout	Click logout	Display the home page

SOFTWARE TESTING LIFE CYCLE



Test Plan

- ➢ Test plan is a detailed strategic document that contains information about a project i.e how to test a project in an effective, efficient and optimized way.
- ➢ This test plan document is prepared by test leads in the organization.
- ➢ In order to prepare the test plan document following reference documents are used i.e BRS, SRS/FRS.
- ➢ Hence the test plan document contains following information.

1.0 INTRODUCTION

1.1 Objective

- It is a strategic document which contains some information that describes how to perform testing on XYZ application in an effective, efficient and optimized way.

- Test plan is a guideline document for testing an application which helps to determine the following areas.

1.2 Reference documents

- The list of all documents that are referred to prepare the test plan will be listed out here. Eg: SRS.

2.0 SCOPE

2.1 Features to be tested

- The list of all the features that are in the scope and plan for testing will be listed out here.

2.2 Features not to be tested

- The list of all the features that are not planned for testing will be mentioned here
    - Features that are out of scope.
    - Low risk features.
    - Features that are planned to incorporated in future.
    - Features that are skipped based on timing constraint.

3.0 TEST STARTEGY

3.1 Levels of testing

- The list of all levels of testing that are maintained by the company will be mentioned here.
  Eg: unit, integration, system, UAT testing's.

3.2 Types of testing's

- The list of all types of testing's that are maintained by the company will be mentioned here.
  Eg: static and dynamic testing

3.3 Test design techniques

- The list of all the test design techniques that are maintained in the project will be mentioned here.
  Eg: black box test design techniques

3.4 Terminology

- The list of all the terms and the corresponding meanings used in the company will be mentioned here.

3.5 Automation plan

- The list of all the areas that are planned for automation testing will be mentioned here.
  Eg: functional areas, regression areas.

3.6 List of automation tools

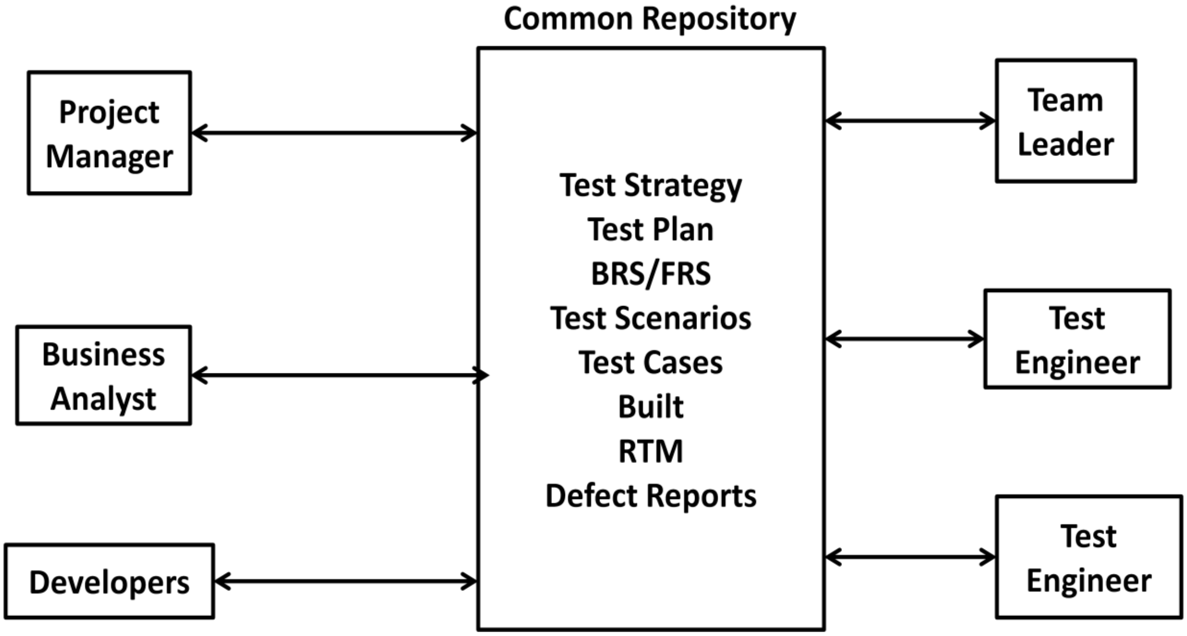
- The list of all the automated tools that are used in the company will be listed out here.
  Eg: functional & regression tools: QTP, RFT, Selenium
  Defect tracking tools- QC, Jira, Bugzilla
  Performance tools- LR, Test Complete etc

3.7 Test metrics

- The list of all the things that are planned to be measured during the testing process in the company will be mentioned here.
  Eg: test case review, defect case doc review, traceability review etc

3.8 Configuration management

- A centralized computer system where we store and manage all project resources is called as common repository.



4.0 BASE CRITERIA

4.1 Entry criteria

- A set of precondition to start an activity is called as entry criteria.
    - 100% unit & integration testing should be successful.
    - All the customer requirements should be baselines.
    - All test cases should be reviewed & approved.
    - Application should be successfully installed in the testing environment.
    - All required software & hardware resources must be available.
    - Testers should get an access rights to access the documents and applications.

4.2 Suspension criteria

- When the pending defects are more.
- Malpractices.
- Environment failures etc.

4.3 Exit criteria

- A set of post conditions to stop an activity is called as exit criteria.
    - All test cases should be executed successfully.
    - All major defects should be fixed and closed.
    - Project has to map to the deadlines.

5.0 ENVIRONMENTS

- ➢ The complete details of environment that is about to be used for testing purpose will be clearly described here.
  Eg: client-server, web environments.

6.0 SDLC MODELS

- ➢ The complete details of SDLC models that is about to be used for testing purpose will be clearly described here.
  Eg: agile model, V-model etc.

7.0 RESOURCE PLANNING

- ➢ The list of all hardware & software resources that are planned for the testing will be mentioned here.

8.0 STAFFING & TRAINING

- ➢ To accomplish the project successfully if a staff needed to be recruited (or) training needed to be provided these details will be mentioned here.

9.0 SCHEDULES

- The start and end dates of each task will be clearly planned and specified here.

  Eg:

S.No	**Task Item**	**Responsible**	**Planned Hours**
1	Test design for module-1	Sekhar	60Hrs
2	Test design for module-2	Muni	50Hrs

10.0 TEST DELIVERABLES

- The list of all the testing related documents that are maintained in the project will be mentioned here.
  - Test plan document
  - RCN documents
  - Test scenario documents
  - Test case documents
  - Traceability documents
  - Defect reports
  - Notes of communication with developers
  - Test summary reports

11.0 RISK & CONTINGENCIES

- The list of all the potential risks and the corresponding solution plans will be specified here.

  Eg:

  Risks:

  - Employees may leave the organization in the middle of the project.
  - Unable to deliver the project within the deadlines.
  - Unable to test all the features within the given time.
  - Customer may bring forward the deadlines.

  Contingencies (or) mitigation plan:

  - Employees need to be maintained on the bench.
  - Proper plan ensurance.
  - Concentrate on the priority based features.
  - What to be skipped need to be checked in case of deadlines.

12.0 ASSUMPTIONS

- The list of all the assumptions that are to be assumed by the testing people will be listed out here.

13.0 APPROVAL INFORMATION

- Who has approved and when it is approved will be clearly mentioned here.

Test Analysis

- Analysis is nothing but understanding the technical documents like SRS/FRS etc.

Roles: Test leads, test engineers, subject matter experts (SME).

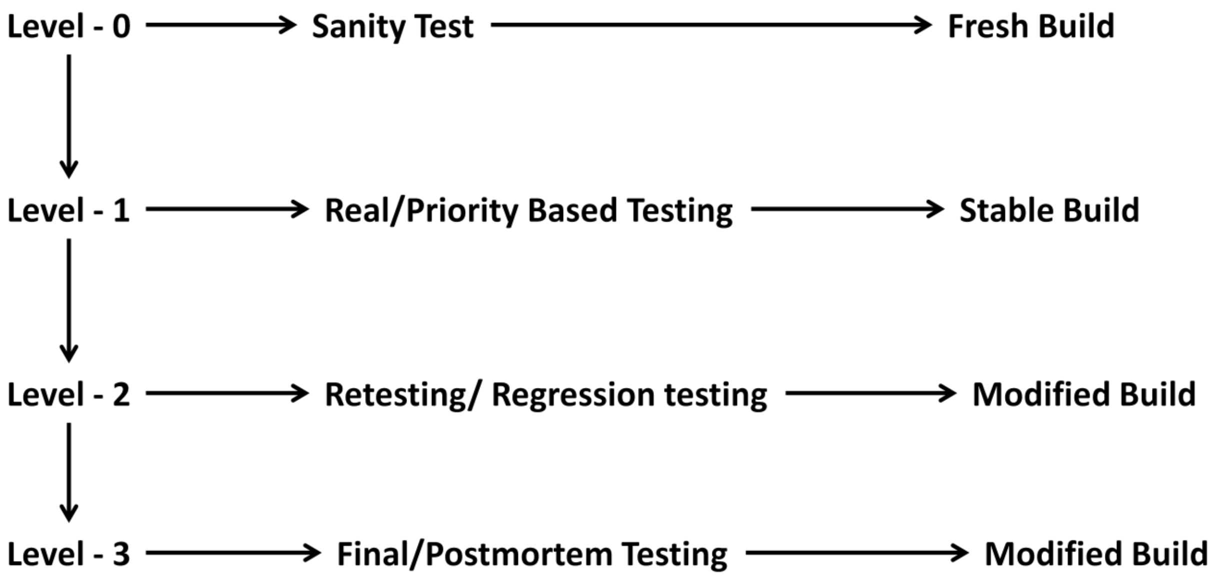
NOTE: Once the document analysis is done, if any clarifications found in the technical documents then these clarifications will be mentioned in the RCN documents.

NOTE: RCN is also called as Root Cause Analysis (or) Gap Analysis Documents.

Test Design

- Test design covers test scenarios and test cases.
  - **Test scenario:**
    - A test scenario is an item (or) functionality to be tested is called as test scenario.
    - In simple terms test scenario is nothing but "**What to be tested**".
  - **Test case:**
    - A test case is a set of pre-conditions, steps to be followed, input data, expected results to validate the functionality in the system.
    - In simple terms test case is nothing but "**What to be tested**" and "**How to be tested**".

Test Execution



NOTE: Once the test design process is done, i.e once the test cases are reviewed and approved successfully, on the released build tester's will execute test cases as follows.

Level – 0:

- In order to execute the test cases first of all tester's conduct the "**Sanity Test**" on the released build.

Level – 1:

- Once the sanity test is passed testers will execute the test cases based on the priority basis i.e high, medium, low. Hence this process itself is called as real/priority based testing.

Level – 2:

- At the time of executing the test cases, if any issues found these issues will be logged in the defect reports and hence these reports will be shared to developers to fix the issues. Developers will fix the issues and release the modified build to the testing department. On the modified build tester's will conduct retesting/ regression testing.

Level – 3:

- After successful completion of level – 0, level – 1, level – 2 testing team lead will identify the highest defects density module and hence testing once again conducted on these modules is called as final/postmortem testing.

Test Report

- Test report covers the results i.e whether the test cases are pass/fail.

Error:

- Any incorrect human action that results to mistakes is called as error.
- Humans mean BA, Architect, Developer, Tester, Deployment engineer, Customer.

Bugs:

- Error in coding is called as bug.
- Developers are responsible to fix the bugs.

Defect:

- It is defined as deviations between the expected behavior and actual behavior is called as defect.
- Testers are responsible to find the defects.

Failure:

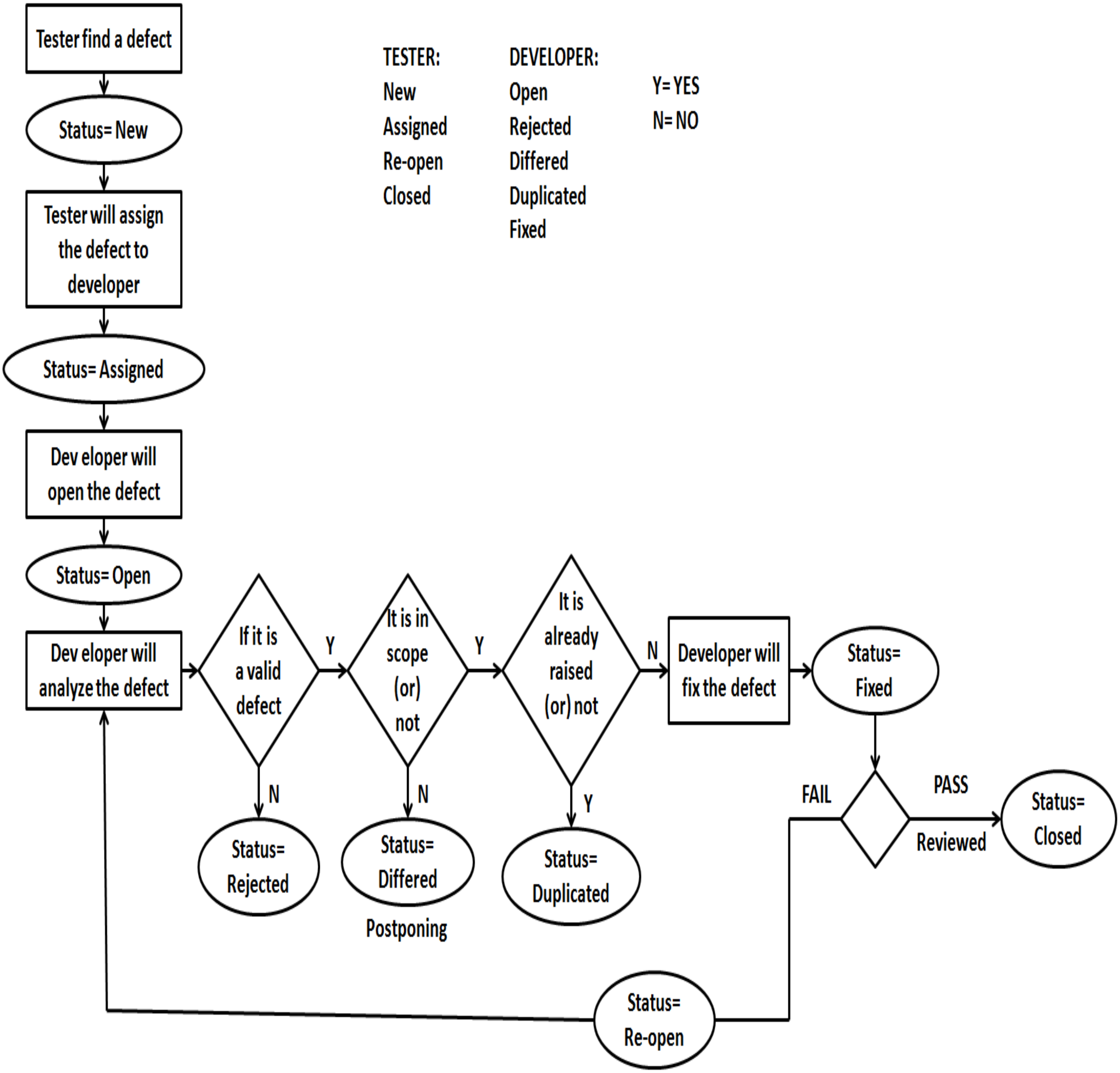
- Problems identified by the end users while using the product is called as failure.
- Customers are responsible to decide product is success/failure.

- Most common defects that can be found in any software application are as follows
  - Functionality not working

- Poor performance
- Security related issues
- Compatibility related issues
- Environment related issues
- GUI related issues
- Code related issues
- Design related issues
- Exception related issues
- Wrong calculations
- Usability related issues etc

NOTE: All these defects are monitored in the defect reports.

Defect/Bug Life Cycle:



Defect Seveority:

- The impact of the defect on the system is called as Defect Seveority.

  (Or)

  How seriously the defect is affecting to the client business is called as Defect Seveority.
- Testers are responsible to assign the Defect Seveority.
- Hence the Defect Seveority can be as follows

Org 1	Org 2	Org 3	Description
Very high	S0	Fatal	Functional Related issues Security Related issues Performance Related issues Compatibility Related issues
High	S1	Major	
Medium	S2	Medium	
Low	S3	Minor	GUI Related issues Usability Related issues

Defect Priority:

- The order in which the defect has to be resolved is called as defect priority.
- Testers are responsible to assign the defect priority.

Org 1	Org 2
Very high	P0
High	P1
Medium	P2
Low	P3

Note: In general Defect Seveority and Defect Priority are proportional to each other i.e

Defect Seveority	Defect Priority	Example
Low	High	Incorrect Logo
High	Low	Peak Hours Usage, calls, Construction of flyovers

High	High	Incorrect Functionality etc.
low	Low	GUI related issues

Defect Age:

- It is defined as the delay between the date of detection and date of closure is called as defect age.

  (Or)

  How long the bug was present in the development life cycle is called as defect age.

Traceability matrix:

- Mapping requirements and test cases is called as traceability matrix.
- It is also called as requirement traceability matrix.

Q) Why should I go for traceability matrix?

Answer:

In order to ensure whether all the requirements covered successfully (or) not, to ensure all the test cases executed properly (or) not, to check whether all the defects fixed (or) not we can opt for traceability.

Types of traceability's:

- Three types of traceability's are there
  - **Forward traceability:** Mapping requirements with test cases is called as forward traceability.
  - **Backward traceability:** Mapping test cases with requirements is called as backward traceability.
  - **Bi-directional traceability:** It is a combination of forward and backward traceability.

Advantages:

- To ensure requirement coverage.
- To ensure test case coverage.
- To ensure defect coverage.

- To get the alerts whenever requirements, test cases are modified.
- To provide clear documentary proof.

Test Closure

- After successful completion of all the phases of STLC in order to close the project following areas has to be analyzed
  - **Coverage analysis:**
    - Check whether all the requirements covered successfully (or) not.
    - Check whether all the test cases executed successfully (or) not.
    - Check whether all the defects fixed properly (or) not.
    - Check whether all the defects closed properly (or) not.
    - Check whether the project has been completed within the specified time duration i.e deadline (or) not.
  - **Defect analysis:**
    - Check how many defects found in the project.
    - Check how many defects fixed.
    - Check how many defects are closed.
    - Check whether the backlogs are covered successfully (or) not
    - Backlog is nothing but request pending.
  - **Performance analysis:**
    - Check how many hours he/she worked on the project.
    - Check how many test cases he/she written and executed.
    - Check how many defects he/she found and closed.
    - Check whether he/she completed the task within allocated duration (or) not.
  - **FTSR:**
    - FTSR – Final Test Summary Report.
    - This document is prepared by the team lead. Hence this document covers
      - Test plan
      - Test design
      - Test execution
      - Test report

    - Test closure

- **Signoff:**
    - After successful completion of the project, project manager announces signoff and hence testers and developers are released from the project and are assigned to the new project.
    - Based on the performance of the previous project some of the developers and testers will be promoted to higher levels and hence this team will take care of maintenance.

DRAWBACKS OF MANUAL TESTING

- More number of resources is required.
- Huge manpower.
- Time consuming.
- 24/7 testing is not possible without tester interaction.
- No accuracy.
- Less efficiency.
- It is not reliable when new patches are released.
- It is not reliable to test with huge amount of test data.
- It is not easier to conduct functional, regression and performance testing.
- Chances of skipping the test cases while executing.

NOTE: Hence the best solution to overcome all above drawbacks is "**Automation Testing**".