UNIT-I

INTRODUCTION AND FUNDAMENTAL OF TESTING

1.1 Challenges in Software Projects

- Handling the **manpower attrition** is the main challenges in software organization.
- Developing **zero-defect software**.
- Developing and Delivery of good software within the **time frame and within the budget**.
- Providing **good security features** to the software.
- Increased **complexity of the project.**
- Shortage of **skilled people**.

1.2 Reasons for Software Failure

- **People -** People related problems like communication, personal skills and ethics.
- **Technology-** Introducing new technology leads to lot of problems.
- **Time**-compressing the development time.
- **Money-** project must be completed within the budget.

1.3 Skill Sets for Testing Professionals

- Good understanding of **software quality assurance** and **quality management** standards such as CMMI (Capability Maturity Model Integration)
- Testing professionals must have knowledge of testing process, testing types, methods, designing and executing tests.
- Excellent communication and inter-personal skills required to inform bad news that is bugs
- Good documentation skills to prepare the testing process document
- Good **listening skills**
- Good understanding of the **code of ethics** and **professional practice**

1.4 Tasks Handled by Testing Professionals

- Understand the application domain
  - Understanding the application domain likes
    1. Client or server based DBMS
    2. Data communication software.
    3. Embedded software.
    4. Websites.
    5. Telecom software.
- Learn about the development environment
  - Development environments, operating systems like windows, Unix, Linux.
  - Programming languages – C, C++, java, C# etc
- Learn how to use the application
  - Tester must learn how to test the application
- Study the source code
  - Study the source code is required to validate the source code
  - SRS Documents
- Study the requirements and specifications of software’s
- Study the acceptance test procedure
- Report the test results
- Be proud of your work

1.5 Fundamentals of Testing

- **What is testing?**
    - ✓ Testing is process to check whether the software meets the uses requirements or not.
    - ✓ The object of testing is to find the defects.
- **Testing V/S Debugging**

Testing	Debugging
Testing is to find the defects	Debugging is a process of line by line execution of the code and the fixing the errors.
Testing is done by the tester	Debugging is done by the developer.
Testing is identifying the errors.	Debugging is fixing of removing the identified error.

- **Testing and Bebugging <u>OR</u> Defect Seeding**
    - ✓ Bebugging means some bugs are intentionally introduced the software.
    - ✓ Bebugging is also known as **“defect seeding”**.
    - ✓ Defect Seeding is implanting the defect intentionally in the code to test the unit test case or specific condition or exception handling.
- **Verification and validation**

  Verification
    - ✓ **Are we building the product right**?
    - ✓ Verification is to check whether the software conforms to specifications.
    - ✓ Verification is carried out by the **development team**.
    - ✓ Verification is done by **Quality assurance team**.

  Validation
    - ✓ **Are we building the right product**?
    - ✓ Validation is to check whether the software meets the customer expectation and requirements.
    - ✓ Validation is carried out with the **involvement of customers**.
    - ✓ Validation is done by **Quality control team**.
- **Root cause Analysis**
    - ✓ Root cause analysis means to find out why bugs are occurred.
    - ✓ It is very important aspect of testing and debugging.
    - ✓ Example int temp =25;

      When tester is testing above line code, it works for 25^0 temperature. It cannot work below and above 25^0 temperatures.

1.6 Significance of Testing

- ✓ There is communication gap between the user and requirement analysts, software has not been developed as per requirements.
- ✓ Due to communication gap among the developers.
- ✓ Programmers have inability to do efficient coding so result would be in defective code.
- ✓ Software has been tested on a simulated environment. Not yet tested in actual environment
- ✓ Software has been developed in developer environment. It must be tested on the actual hardware before it is delivered to the customer.
- ✓ Need to check the privacy, safety and security of the data before delivery to customers.
- ✓ Due to constraints of time, software has developed in hurry, so quality needs to check.

1.7 Cost of Quality

- ✓ There are 3 cost of quality
    1. Failure coat

2. Appraisal cost
3. Prevention cost

1. Failure cost
   - Failure cost is the cost of fixing the product defects
2. Appraisal cost
   - It is the cost of assessing the product to check whether it has defects or not.
3. Prevention cost
   - It is the cost of modifying the process to avoid defects in the software.

1.8 Psychology of Testing

- ✓ Developer's point of view
  - Testing is the process to prove that the software works correctly.
- ✓ Tester point of view
  - Testing is the process to prove that the some bugs are found in software.
- ✓ Manager's point of view
  - Testing is the process to find the defects and minimize the risk.
- ✓ Test engineers and developers need to interact regularly because to develop quality software.

1.9 Testing Choices

- ✓ There are two testing choices
  1. In-House Testing
  2. Outsourcing or Outsourcing Testing

1. In-House Testing

- ➢ In-House Testing means do the testing process within the organization.

Advantages

- Intellectual Property will not be given to a third party.
- Management will have total control over the budget.
- Developers and Testers can interact easily and need bases.

Disadvantages OR Limitations OR Difficulties

- Maintenance of Infrastructure is costly.
- Necessary expertise may not be available always
- Getting the Man power, training them and keeping them is challenging.

2. Outsourcing or Outsourcing Testing

- ➢ Outsourcing means to give the sub contract of testing process to the third party.

Advantages

- Need not to maintain the infrastructure like servers, high bandwidth network, testing tools.
- Need not worry about recruiting and training the manpower.
- Faster at subcontracting premises
- Cost Saving

Disadvantages

- Intellectual Property may be transferred to third party.
- Time frame is not control under control of primary contract.
- If the outsourcing organization is located in different country, laws, copyrights and cyber-laws may be different.
- Bad experience if sub contractor is incorrect.

1.10 Who does the Testing?

1. Developer as Tester
2. Independent Testing Team

1. Developer as Tester

- Developers are familiar with user requirements, code activities, so developer can generate **test cases.**
- Developers are familiar with structure of programs, structure of modules, so he can do better testing.
- Developer knows the limitations/defects of software.
- Developer can carry out the following testing
  1. Unit/component testing
  2. Integration testing

2. Independent Testing Team

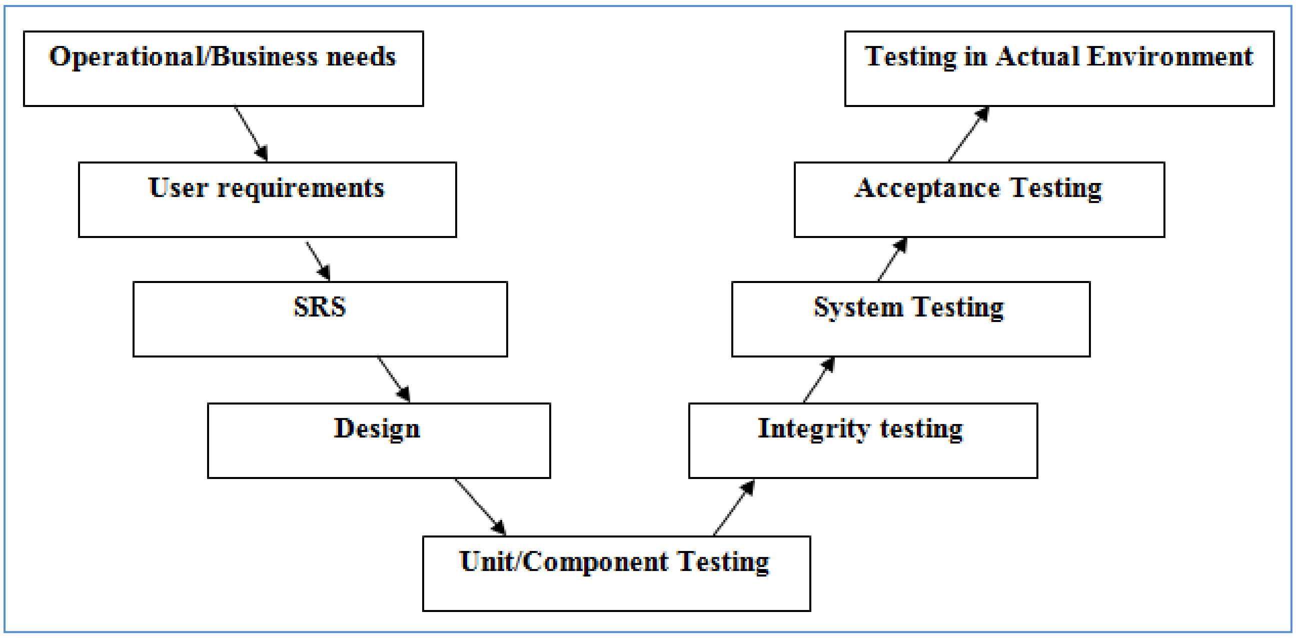
➢ Issues OR difficulties Or disadvantages

- Tester must study the work product such as SRS, design documents and modules.
- Tester may need to know the languages for reviewing the code.
- Tester must constantly interact with developers to come out the good testing result.
- Testing carry out the following types of testing.
  1. System testing
  2. Acceptance testing

1.11 Buddy Testing

➢ When two people sit together and the testing the software product is called buddy testing.
➢ If developer and tester together work, the software quality may be good.
➢ Software organization has separate quality assurance or testing team.
➢ In some organization, developer its self carrying the test.
➢ Buddy testing means quick testing. Few testers form one group and test at a time

1.12 V-Model



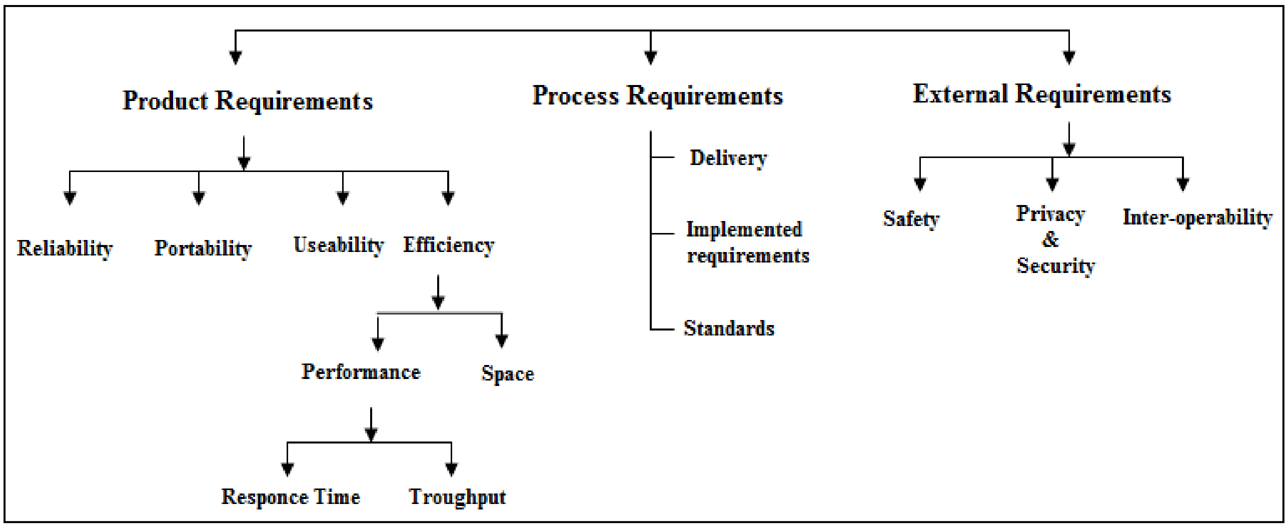
- The V-model shows the various development phases and corresponding testing phases
- The V-model has two parts
  1. Verification phases
  2. Validation phases
- The V-model indicates the various levels at which the software has to be tested
- The product requirement are understood from the customer and convert into user requirement
- Software Requirements and Specification(SRS) is prepared based on the user requirement
- Based on the Software Requirements and Specification(SRS) document , the design document is made
- The various models are designed and each module is divided into Components/Unit
- Unit testing is the testing at code level and eliminates all bugs in early stage
- Integration testing is associated with the design
- System testing resolved the issues in SRS
- Acceptance testing is associated with the user requirement and testing the product in user environment.

1.13 Testing and life cycle Models

1. **Waterfall Model –** In this model, testing starts after the integration module is done. Testing is done on mature product.
2. **Prototyping Model –** Prototyping is tested based on the preliminary user requirements.
3. **Evolutionary Model –** Software is developed incrementally and testing is also done incrementally.
4. **Spiral Model –** Study the risk factors of software and test the software.
5. **Synchronize and Stabilize Model -** In this model, Testing has to be start from beginning and give feedback to developer for further implements.

1.14 Testing the system

- Complete system has been tested based on SRS document.
- SRS document contains two requirements.
  1. **Functional Requirements**
  2. **Non-Functional Requirements**
- Functional requirements <u>specify what the system should do</u>?
- It specify the behaviour OR function of System.
- Non-Functional <u>Requirement describes how the system works</u>.
- Non-Functional Requirement are divided into 3 categorize.
  1. **Product Requirements**
  2. **Process Requirements**
  3. **External Requirements**



1.15 Testing Strategies

1.15.1 Static Testing and Dynamic Testing

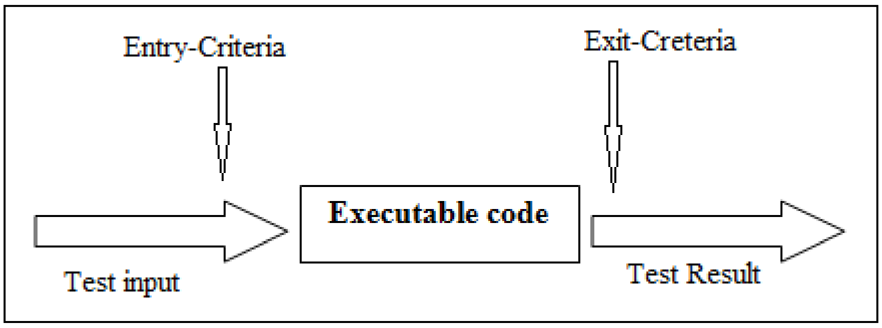
Static Testing	Dynamic Testing
Testing done without executing the program	Testing done by executing the program
This testing does verification process	Dynamic testing does validation process
Static testing is about prevention of defects	Dynamic testing is about finding and fixing the defects
Static testing gives assessment of code and documentation	Dynamic testing gives bugs/bottlenecks in the software system.
Static testing involves checklist and process to be followed	Dynamic testing involves test cases for execution
This testing can be performed before compilation	Dynamic testing is performed after compilation
Static testing covers the structural and statement coverage testing	Dynamic testing covers the executable file of the code
Cost of finding defects and fixing is less	Cost of finding and fixing defects is high

1.16 Why Testing is difficult?

- Test the software for both right and wrong inputs and check the functionality as well as performance
- Each and every line of code is need to test with help of inputs
- Give the inputs randomly and check that the software newer fails
- Test the software in different hardware and software environments
- Test the software like normal user is using it and check the error messages
- Test the functional and non-functional requirements

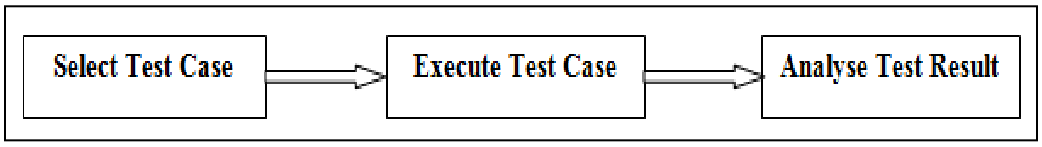
1.17 Testcase

- Test case is a document which specifies the input data, excepted output, entry-criteria and exit-criteria.



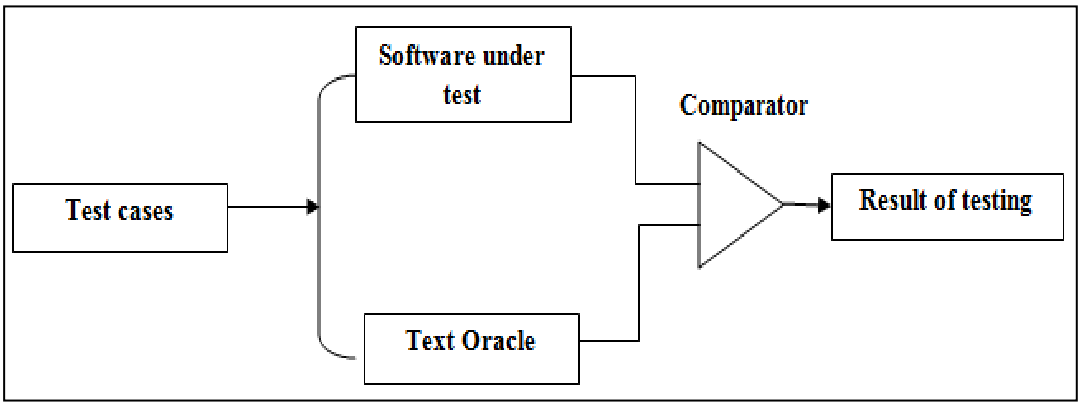
- It consists of
  1. **Entry-Criteria** : Pre-conditions before the test inputs are given
  2. **Test-input**: Inputs are given to execute code to produce the proper output
  3. **Exit-Criteria** : Post-conditions after the test inputs are executed
  4. **Test-result**: Output of the software
- The process of developing test-case can also help for find the problems in requirements and design
- Example :

  Program: Need to test the program take filename as input, open the file and write today date into file and close the file
  1. Test-Input: Filename “aaaa.txt”
  2. Entry-Criteria : File should not exists in the current directory
  3. Expected-Output: File should be in the current directory with today date
  4. Exit-Criteria : Program should terminates without errors
- Testcase Execution :



1.18 Test Oracle

- Automated testing tools act as “Test Oracle”
- Test Oracle is a repository of knowledge
- Test Oracle’s are the machines that check the correctness of the program for given test cases



1.19 Manual V/S Automated Testing

Manual Testing	Automated Testing
Executing the test cases manually without any tools **OR** Testing the software manually is called manual testing	Executing the test cases by using automation tools **OR** Testing the software is done by using automated tools
More testers required for manual testing	Less number of testers are required for automated testing
Every time testing is repeated because of human errors	Lot of repetition, unproductive work can easily done in automated testing
Manual testing is less reliable	Automated testing is more reliable
Manual testing is very slow	Automated testing is very fast
Non-programmable in manual testing	Programmable in automated testing

1.20 Testing software of different technologies

➢ The following are the different technologies will consider during the testing phase.
   1. Client-Server Software's
   2. Web-Based Applications
   3. Object-Oriented Software's
   4. Mobile-Based Applications

1. **Client-Server Software's**
   - ✓ In client-server system, all resources are available in server and multiple clients are accessing the resources simultaneously
   - ✓ In client-server system, tester needs to test performance and security.
2. **Web-Based application**
   - ✓ Tester need to test the web applications in the
     - different browsers
     - security
     - each and every link on every page needs to be tested
3. **Object-Oriented Software's**
   - ✓ Reusability is the most important feature of the object oriented programming language, so tester needs to test the reusable code.
4. **Mobile application**
   - ✓ Testing the mobile applications in different smart phone.

1.21 Metrics in Testing Phase

➢ The following are the metrics that are helps in monitoring the process of the project and also improving the testing process.

1. **Estimated Time**
   - Estimated time for the testing phase and actual time taken for the testing phase.
2. **Estimated Budget**
   - Estimated budget for the testing phase and actual money spent on the testing phase.
3. **Estimated Effort**
   - Estimated effort for the testing phase of actual effort.
4. **Defect density**

- Defect density is the number of defects found per 1000 lines of code is called defect density.

5. **Defects**
    - Number of defects found per unit time.
6. **Cost of Quality**
    - Failure cost
    - Appraisal cost
    - Prevention cost

1.22 Risk based testing

- ➢ Risk based testing means reducing the bugs in the software so that risk will automatically reduce.
- ➢ Types of risks
    1. Product Risks
    2. Project Risks
- ➢ **Product Risk**
    - Product risk means bugs are still exists in the product and delivering that software to the customer.
- ➢ **Project Risk**
    - Project risk means failure to complete the project within the time frame or within the budget
- ➢ **Steps of Risk-based testing**
    - Identifying the risk in projects
    - Analyse the constraints
    - Testing to be carried out based on the constraints
    - Identify the types of testing to be done
    - Identify whether use of testing tools will help in reducing the risks

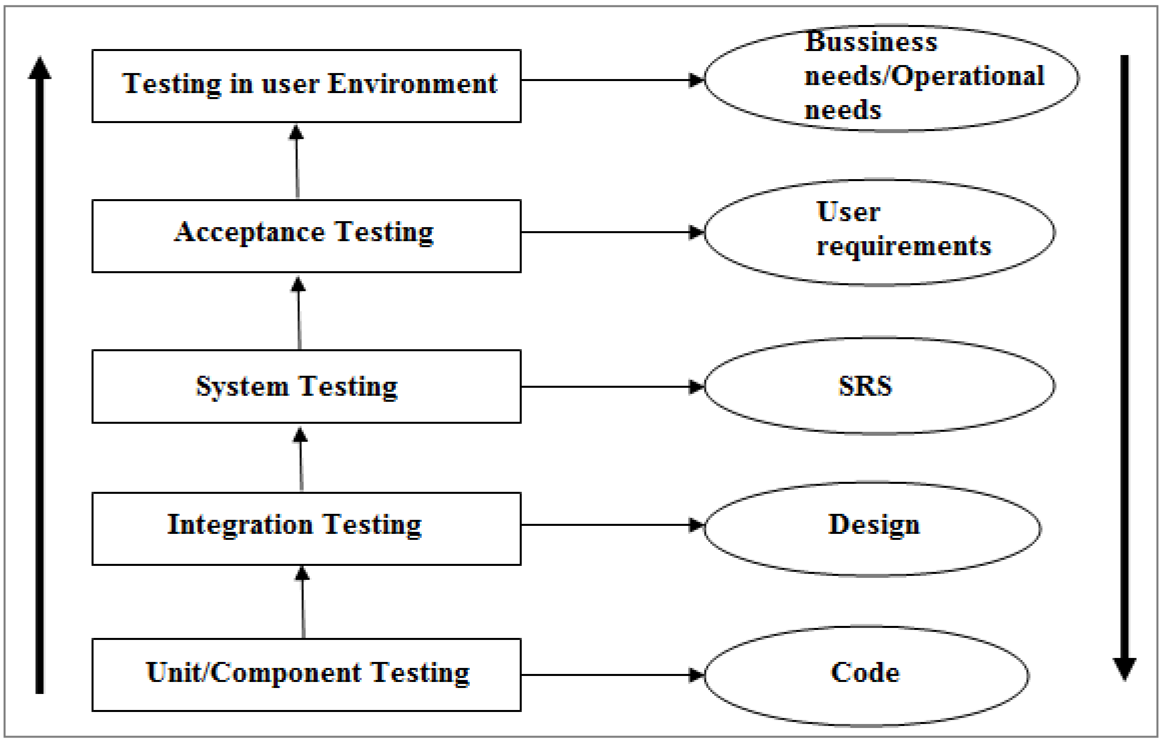
1.23 Myths and Realities of Testing

- ➢ Testing is easy
- ➢ Test engineers have no career growth
- ➢ If I learn testing tools, I will get a job
- ➢ Testing doesn’t involve coding
- ➢ Testing is a phase, that comes after the development phase of the software
- ➢ Testing is completed
- ➢ Developer’s know that their software will have bugs

Unit-II

Testing Levels and Types

2.1 Testing Levels

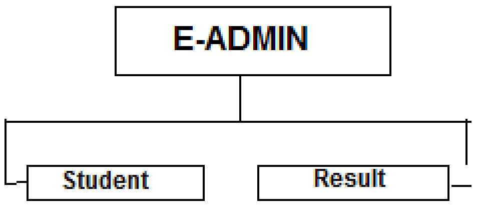


2.1.1 Unit/Component testing

- A unit or component testing is used to testing **individual modules**. Every module has to be tested separately to check whether it working as per specification.
- Unit/Component testing is test the **source code** and it is done by the **developers.**
- This testing is used to check the **functionalities** and ensures that code is written proper way.
- This testing improves the **design** and **quality.**
- Each units are tested one after the other.

2.1.2 Integration Testing

- Integration testing is a level of testing where individual **units are combined and tested as a group.**
- Integration testing is used to tests the **interface between components and interactions** to different parts the system.
- Integration testing follows **two approaches** called **top-down approach** and **bottom-up approach**.
- The purpose of integration testing is to verify the functional, performance, and reliability between the modules.
- Example



2.1.3 System Testing

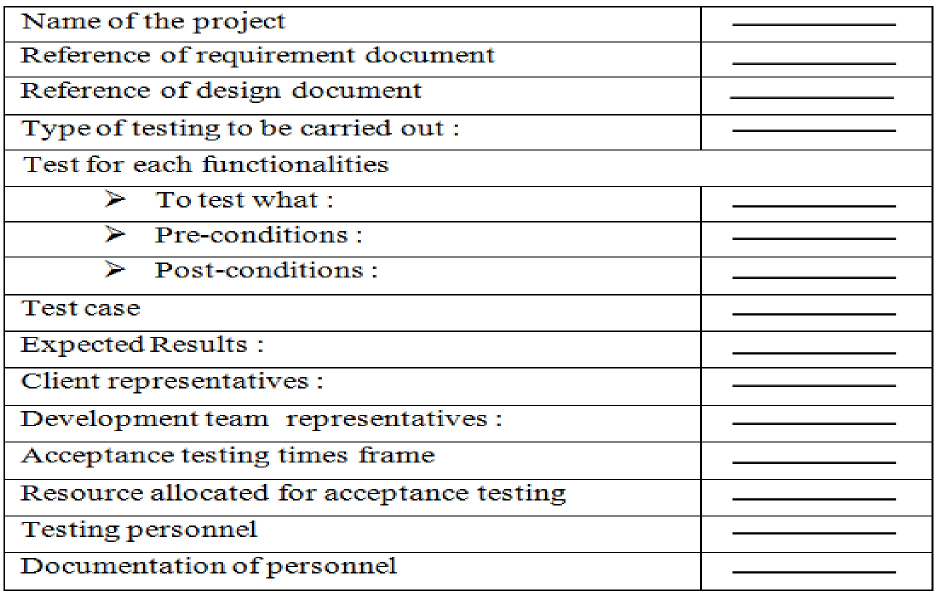
- The system testing is used to test the **functionalities of the system** from an end-to-end perspective.
- The **functionalities of the system have to be tested based on the SRS document**.
- This testing is also test the **non-functional requirements** such as
  - ✓ Performance requirement
  - ✓ Security
  - ✓ Portability
  - ✓ Reliability
  - ✓ Safety
  - ✓ International Standards

2.1.4 Acceptance Testing

- This testing **decides whether the client approves the product or not**
- Acceptance testing can be done either at the **developer or client premises**
- The goal of the acceptance testing is to establish the confidence in the system
- Types of acceptance testing
  1. User acceptance testing
  2. Operational acceptance testing
  3. Contractual fulfilment testing
  4. Regularity acceptance testing

1. **User acceptance testing:**
   - It is carried out by the developers / users based on the **test plan**
2. **Operational acceptance testing:**
   - Software is installed in the actual working environment and observe the **performance** for the period of time and check how the system meets the user operations
3. **Contractual fulfilment testing:**
   - If there is any special product like aircraft, then testing has to be done in actual environment / artificial environment
4. **Regulatory acceptance testing:**
   - The software has to be tested based on the international standards, rules and regulations of government.

Acceptance Test Plan (ATP) Format



Name of the project	
Reference of requirement document	
Reference of design document	
Type of testing to be carried out :	
Test for each functionalities	
➢ To test what :	
➢ Pre-conditions :	
➢ Post-conditions :	
Test case	
Expected Results :	
Client representatives :	
Development team representatives :	
Acceptance testing times frame	
Resource allocated for acceptance testing	
Testing personnel	
Documentation of personnel	

2.2 Testing Approaches

2.2.1 Static Testing vs Dynamic Testing

- **Note: Please refer 1.15 concept of Unit –I and page number 16**

2.2.2 Positive testing v/s Negative testing

- Positive testing is the process where the system is **validate against the valid input data.**
- In positive testing, tester always tests the applications for valid set of inputs and check if application produces the accepted result.
- Positive testing ensures that application does that what it is supposed to do.

Example:-

- Negative testing is process of where the system is validating against the invalid inputs.
- In negative testing, tester tests the applications for invalid set of inputs and check if application produces the accepted result.
- Negative testing referred as **failure testing** means software does not do anything

2.2.3 Top-down testing v/s Bottom-up testing

- In top-down testing, testing is conducted from **main module to sub-module**.
- If sub-module is not developed, then temporary program is called **"stub"** is used for **simulate the sub-module**.
- Top-down testing is difficult to carrying the testing at **system level**.
- In bottom-up testing, testing is conducted from **sub-module to main module**.
- If main module is not developed, a temporary program called "**driver**" **is used to simulate the main module**.
- Bottom-up testing is easy to test the system and also sub system.

2.2.4 Functional testing v/s Structural testing

- **Functional testing**
  - ✓ In functional testing, the **functionality of the module is tested** and structure is not considered.
  - ✓ In this testing, test cases are designed based on the functions of the module.
  - ✓ It is also called **"black-box"** testing.
- **Structural testing**
  - ✓ In structural testing, testing the **structure of the module** (code) or web page structures.
  - ✓ Structure testing may be either **system level** or **integration level**. System means menu structure or web page structure. Integration level means interconnected module.
  - ✓ In this testing, test cases are designed based on the structure of the module.

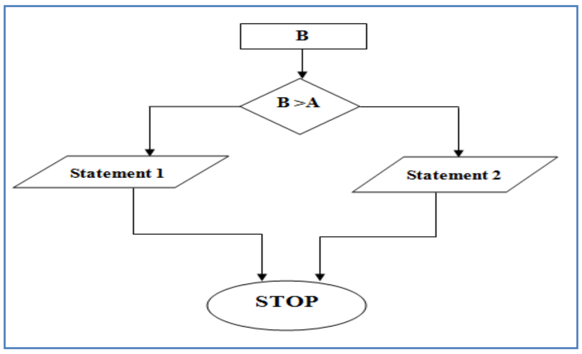
- ✓ It is also called as **white-box.**
- ✓ Structure testing involves
  1. Statement coverage
  2. Condition coverage
  3. Path coverage
  4. Functional coverage

1. Statement coverage

- ✓ It ensures **that each and every line in the code is executed**.
- ✓ Profiler in the tool indicates that which lines of code is execute and which line of code is not executed.

2. Condition coverage

- ✓ Each and every branch is executed at least once.
- ✓ Example :



3. Path coverage

- ✓ It ensures that each path is executed at least one.
- ✓ Path coverage deals with logical paths in the program
- ✓ Path coverage = (number of paths executed/ total number of paths)*100.

4. **Functional coverage**

- ✓ Each and every function is executes at least once.

2.3 Mutation Testing

- ➢ Mutation testing is **required to ensure that software does not fail**
- ➢ Mutation testing is used to **simulate wrong input and reduce the risk**
- ➢ It is a process of rewriting the source code in order to remove the redundancies in source code
- ➢ The mutation operators are
  1. Constant replacement
  2. Variable replacement
  3. Arithmetic operator replacement
  4. Relational operator replacement
  5. Goto label replacement
- ➢ Example :

```
int y;
x=sin(y);
printf(x);
```

2.4 Confirmation Testing

- **Re-testing** is a confirmation testing.
- When defect is found in software, the developer makes some change in the code to remove that defect. When the developer indicates that defect has been fixed, it has to be confirmed by the tester that defect has been removed, this is called Confirmation testing.
- Confirmation testing is done to make sure that there is **no defect in the software**

2.5 Regression testing

- Regression testing is done to ensure that a change in **one part of the program does not affect other part of the program**.
- Regression testing involves both **dynamic testing** and **static testing.**
- Regression testing is required when
  - ✓ Changes in requirement
  - ✓ Code modified according to the requirement
  - ✓ New features is added to the software
  - ✓ Defect fixing
  - ✓ Performance issues fix

2.6 Smoke Testing

- It is also called as "**build verification testing**".
- The term "smoke testing" is came from the **hardware testing**.
- This testing check the initial pass is done or catches the fire or smoke in the initial switch on.
- It is used to test the critical **functionalities of the program** is working fine.
- It helps to find issues introduced in integration of module.

2.7 Black Box Testing

- Black box testing is a **system testing** , in which the **functionality of the software is tested**
- **It is used to test the software without checking the internal structure of the software**
- It is carried out by the **tester**
- Implementation knowledge is not required to carry out black box testing
- Programming language is not required to carry out black box testing
- It is also called as "**Functional Testing**" or "**Exterior Testing**"
- The main objective of the black box testing is to check what functionality is performed by the system

2.8 White Box Testing

- White box testing is a used to **test the internal structure of the software**
- In this testing , **each and every line of code is tested**
- It is carried out by the **developer**
- Implementation knowledge is required to carry out white-box testing
- Programming language is required to carry out white-box testing
- It is also called as "**Structural Testing**" or "**Interior Testing**"
- The main objective of the **white box testing is to check how the system is performing**.

2.9 Interface testing

- ➢ Interface testing is used to test whether the communication between two subsystem are done correctly or not.
- ➢ Types or categories
  1. Hardware – Hardware Interface
  2. Hardware – Software Interface
  3. Software – Software interface
  4. Human – Computer interface

1. Hardware- Hardware Interface

- ✓ It is used to integrate two hardware devices.
- ✓ Example - interconnection of two computers

  It includes

  i. Physical connection

  ii. Electrical parameters

  iii. Protocol

2. Hardware-Software Interface

- ✓ It is difficult to test the embedded systems.
- ✓ Interface between hardware and application is done through the **operating system**.
- ✓ Example - adding the graphics card to system

3. Software-Software Interface

- ✓ It is used to testing the **integration of modules**
- ✓ it includes

  i. Parameter setting by one module with another module

  ii. Shared memory

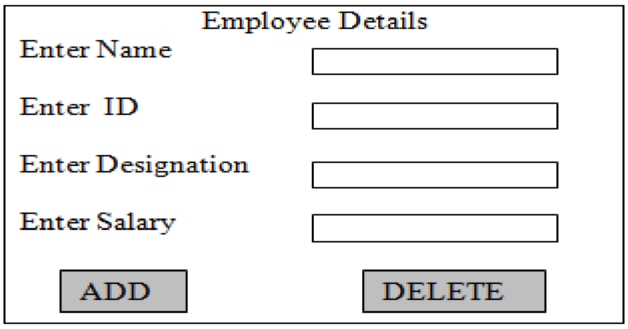
  iii. Calling function

4. Human -Computer Interface

- ✓ It is used to test the **functionalities** of the **<u>graphical user interface(GUI)</u>**

2.10 Use Case Testing

- ➢ Use case testing means **testing the sequence of operations and check whether the desired result is obtained or not.**
- ➢ **Example :**



Note: Explain this module

2.11 Gorilla testing

- It is also called as **monkey testing** or **random testing**.
- Gorilla testing is a randomly pressing of some keys.
- Gorilla testing is use to check whether **defensive programming** has been done or not through **defensive programming** software is made to tolerate wrong inputs.
- It improves the **quality of software**.

2.12 Alpha testing

- This type of testing is **conducted by the end users in the presence of development team**.
- In this testing, developers can rectify the problems encountered during the testing process.
- This type of testing is carry out for every project because will get early feedback from users.
- Alpha testing is final testing before the software is released to the public or clients.

2.13 Beta Testing

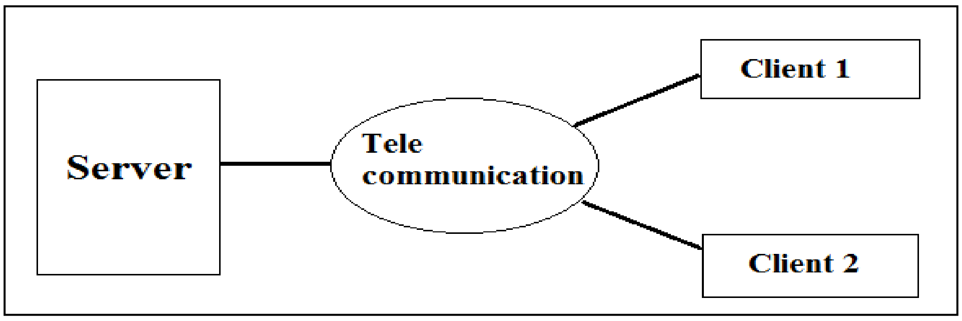
- This type of testing is done by **the client or users in the absence of the development team.**
- The beta testing is important for **generic software product** such as OS, database engines, compilers etc...
- In this testing, the following aspects need to be decided.
  1. **Number of beta test sites**
     - ✓ Numbers of test sites are required to be test the software under different environment.
  2. **Environment required for the beta testing**
     - ✓ Exact hardware and software configuration need to be specified.
  3. **Providing supporting services like help-line, help desk.**
  4. **Whether beta test software is priced or free.**
  5. **Defect reporting mechanism.**
  6. **Beta testing period.**

2.14 Field trail / operation testing

- Field trail / operation testing is used to test the software in **actual working environment**.
- It is similar to beta testing.
- This testing observes the performance of the software in actual working environment.
- The project team has to prepare the **hardware and software requirements for conducting the field trail.**
- The team has to prepare the log sheet during field trail. The log sheet contains
  1. Date
  2. Pre-conditions
  3. Post- conditions
  4. Description of problems
  5. Impact

2.15 Performance Testing / Load Testing

- Performance testing is used to test the software for <u>**some parameters such as response times, throughput, waiting time etc**</u>
- Performance testing is compulsory for client-server applications and web applications, because these applications are accessed simultaneously by large number of users.
- The web application parameters like response time, maximum number of hits per second measured through performance testing
- In real time or embedded systems the performance parameters are very significant.
- Performance testing is compulsory in **process controls and telecommunication software system**
- **Example - Networking**



2.16 Stress Testing

- Stress testing is the process of determining the **ability of computer network devices to maintain a certain levels of effectiveness under some conditions.**
- Stress testing is done on the **resource requirement**.
- It is also used to test the system that involves communication media.
- Example : To find the ability of systems in different RAM size like 120MB, 512MB, 1GB.

2.17 Conformance Testing

- This testing is used to testing the software to ensure <u>**that software meets a defined set of standards**</u>
- The standards are defined by
  - International Organization for Standardization [ISO]
  - World Wide Web consortium
  - International Telecommunication Union
  - American National Standards Institute
- The standards are specified to provide
  1. Interoperability among different equipment vender's (mobile to mobile communication)
  2. Safety to the users (damage)
  3. Compatibility with the government regulations
  4. Accessibility
  5. Privacy to the users
- Testing the software compliance to the requirement specified in the national bodies / international bodies

2.18 Security Testing

- It is a type of testing , that is done to check **whether the software or product is secured or not**
- It is used to find the vulnerabilities (mistakes or problems) of systems and networks.
- It is a process to determine that system protects data and maintains functionality as intended
- The security testing is performed to check whether there is any information leakage in the sense by encrypting the application

2.19 Maintenance Testing

- Maintenance testing is done on the already deployed software
- The deployed software needs to be enhanced , changed or migrated to other hardware
- This testing done during the enhancement, change and migration cycle
- This type of testing needs to maintain the software from time to time in order to avoid system breakdown
- It is considered as two parts
  1. Any change made in the software should be tested thoroughly
  2. Changes made in software does not affect the existing functionality of the software

2.20 Internationalization

- Internationalization is the process of designing and coding a product so you can be performed properly **when it is modified for different languages**.
- It is a process to test the modified software so that it is completely independent of any specific information.
- The following are the benefits for internationalization
  a. It reduces the overall testing cost
  b. It reduces supporting cost
  c. It has more flexible and scalability

2.21 Documentation testing

- Documentation testing means **testing the document those are supplied to the customer.**
- Most important document is **user manual** and other documents are
  - ✓ Administrator Manual
  - ✓ Developer Manual
  - ✓ Installation manual
  - ✓ Design Manual
- These documents are divided into 10 categories
  1. Planning
  2. Standards
  3. Development
  4. Software requirement verification
  5. Software design verification
  6. Software code verification
  7. Verification of integration process
  8. Verification of testing process
  9. Configuration management
  10. Software quality assurance

UNIT- III

STATIC TESTING TECHNIQUES

3.1 Static Testing and its Advantages

- In static testing, the software is tested without executing the code. It can be done manually or by using set of tools.
- Static testing can be done on work documents like SRS, Design and document code etc.

Advantages

1. Incorrect specifications and missing specifications can be found at early stage of the development.
2. Life cycle time of work product is reduced by finding incorrect design.
3. Improved maintainability of code and design.
4. Less cost efforts and time is required for development of product.
5. Prevention of defects.

3.2 Manual Review

- Manual review is the important mechanism for static testing.
- Manual review is used to find the defect in documents.
- Manual review can be done on any of the work products like
    1. SRS document
    2. Design document
    3. Source code
    4. Test plan
    5. Acceptance test plan
    6. Test cases and test scripts
    7. User manual

3.2.1 Formal Review Process

- Review can be done through meeting by the project manager.
- Formal review process consists of the following steps
    1. **Review Objectives**
        - To find the defects
        - To find the defects and discuss how to solve it.
        - Assign responsibility to persons for rectifying the defects.
    2. **Review Team**
        - ✓ The project manager has to create a review team. It consists of

I. Author

- Creator of the work product, which is being reviewed.

II. Reviewer

- Reviewers can give their expert opinion and find the defects objectively.
- Reviewers must have application domain knowledge, programming language knowledge and business process knowledge.

III. Secretary

- Secretary should record the proceedings of the meeting and write down the important points and take the action to follow-up.

IV. Moderator

- Moderator has capability to resolving issues and working towards the objectives.

V. Manager

- Manager identifies the objectives, plan the review meetings and provide the necessary infrastructure and resources.

VI. Plan for meeting

- It includes
  1. Preparation of activities
  2. Actual review meeting
  3. Rework
  4. Follow-up.
- Review meetings are also called as
  1. Informal Review
  2. Walkthroughs
  3. Inspections

3.2.2 Informal Reviews

- ➢ In this review, author and reviewers will sit together and go through the work product and try to find defects in the work product.
- ➢ This review does not follow any process to find the defects in a document.
- ➢ It is more effective if they are peer reviews.

3.2.3 Walkthroughs

- ➢ Walkthrough review means meeting can be conducted by project manager with two or more experts
- ➢ The author of the work product will explain the product details to all team members.
- ➢ It can be done formally or informally.
- ➢ If the walk through is formally minutes of the meeting or record.

3.2.4 Inspections

- ➢ The inspection are like **"formal reviews".**
- ➢ The inspection are then generally with help of checklist or any the standard document.
- ➢ Inspection is done by the team member along with moderator.

3.2.5 Making review successful.

- ➢ The following points needs to consider for review successful.
  1. Review is done to find defects not to find mistakes.
  2. The objective of the review meeting should be clearly defined.
  3. The moderator has to ensures that issues has to be solved.
  4. Review team should have expertise
  5. Checklist and coding guideline should be used in the review.
  6. Project manager should provide necessary resources for the review meeting.
  7. Review should help in improving the overall software development process.

3.3 Checklists

- A checklist is a catalogue of item that are recorded for tracking.
- Checklist are used to verify the work product by the review team.
- There are 4 types of checklists
  1. SRS Checklist
  2. Design Checklist
  3. GUI Checklist
  4. Website Checklist

I. SRS Checklist

- ✓ It contains the following items
  - Whether the document contains complete requirement or not?
  - Can easy to make changes in requirement?
  - Is there any inconsistency in software?
  - Is the SRS document validated by the client?
  - Is the SRS document usable during the development stage?
  - Whether the functional requirement and non-functional requirements specified in the document are verified or not.
- ✓ **Characteristics of SRS Checklist**
  1. Complete
  2. Verifiable
  3. Consistent
  4. Modifiable
  5. Usable

II. Design Checklist

- ✓ It contains the following items
  - Performance requirements meets or not?
  - Is there any design limitations?
  - Whether data structures are identified or not?
  - Is there any error handling mechanisms or not?
  - Is there any design modules?
  - Appropriate algorithms are selected or not?
  - Whether module interconnections are specified or nor?
  - Whether external interface specified or not?

III. GUI Checklist

- ✓ It contains the following items
  - Check whether the application is starting when the icon is double-clicked?
  - Is there every window has the correct title?
  - Whether every field validated or not?
  - If wrong input is given, then error message is getting displayed or not?
  - Is there any shortcut keys?

- Check whether the screens are consistent look and feel or not?

IV. Website Checklist

- ✓ It contains the following items
  - Whether the website is working in all browsers or not?
  - Check whether every link is working or not?
  - Is there search option or not?
  - Whether the every page user friendly or not?
  - Is there any audio content or not?
  - Is there any site map?
  - Whether the content downloaded in maximum 10 seconds?

3.4 Formal Code Reviews

- ➢ It includes
  - Whether coding guidelines are followed or not
  - Check whether there is any unreachable code
  - Check whether online documentation is available or not
  - Check whether the code is portable or not
  - Check whether there is any performance issues are found or not.

3.5 Coding Guidelines OR Programming Style

- ➢ Every programmer has to follow certain guidelines while writing coding.
- ➢ It includes
  1. Layout of the code should be in easy to read.
  2. File name, function name and variable names should be meaningful.
  3. Function size is limited to 40 or 50 lines.
  4. Nesting of code should be avoided.
  5. Error-handling mechanism should be used in code like exception.
  6. Entry-control and Exit-control structure should be used and goto must be avoided.
  7. User-defined types to be used to increase readability.
  8. Each file must have header, which contain name of the programmer, date and revision history.
  9. Unnecessary initialization of variable should avoid. Example : int a=10;
  10. Functionalities of the module must be explained briefly.
  11. Input and output parameters of each function must be explained briefly

3.5.1 C Coding Guidelines

- ➢ The following are the important guidelines in C language
  1. Folder name should be same as **<u>project name</u>**. This folder should have **<u>readme</u>** file that gives list of files and content of file.
  2. Backup of the project folder should be taken every day.
  3. File name should reflect the functionalities of the source code.
  4. Every file should contain header. It contains

- Name of the file.
- Name of author.
- Revision history

5. Identifiers should have meaningful name, identifiers like variable name, function name, structure or union name etc.
6. Comments should be relevant.

   Example : int temp=50; //Temperature is 50
7. Functions and variable should have relevant comments.
8. Constants and macros should be always in **uppercase**.

   Example - #define PI 3.142
9. There should be **<u>single space</u>** while typing operators like comma (,), semicolon (;) and etc.

   Example : a = 0 ;
10. There should be **<u>no space</u>** between the unary operator and operand.

    Example : b-;
11. There should be space after and before the binary operator

    Example: b++
12. Modify the header when the code is modified

3.5.2 Java Coding Guidelines

➢ Java coding guidelines

1. Folder name should be same as the projects name.
2. Backup of the project should be taken every day.
3. Filename should reflect the functionalities of the source code.
4. Java coding conventions should have comments.
5. Method name must have lowercase character and next word character must be uppercase case.

   Example - printAddressLabel ();
6. Constructors
   - Constructors name must be same as class name.
7. Components
   - Components (Buttons, Label, menu etc) must have appropriate name
   - Example : okButton, msgLabel etc
8. Constants
   - Constants are static final fields of class. Constant names must be upper case
   - Example : MIN_BALANCE
9. Local Variable
   - Local variable names must be full descriptor with correct naming
   - Example : String EMPLOYEE_NAME;
10. Exception Object
    - Use letter e for generic exceptions.
    - Example : ArithmeticException e
11. Classes, interfaces and packages should have first letter in uppercase and have correct naming.

3.6 Code Optimization

- It is a method to improve the **code quality and efficiency** so that it becomes smaller size, consumes less memory, executes more rapidly.
- **Code optimization guidelines**
    1. Eliminate dead code
    2. Remove unnecessary debugging code.
    3. Avoid recursion
    4. Avoid floating-point operations
    5. Use unsigned Integers
    6. Avoid large library routines
    7. Remove loop invariant code

1. **Eliminate dead code**
    - ✓ Dead code means code will not be executed at all.
    - ✓ Dead codes likes
        - Comments
        - Initialization of variable that will not be used at all.
        - Functions which will not be call at all.
2. **Remove unnecessary debugging code.**
    - ✓ Lot of code is written during the development time for debugging purpose that code needs to be removed.
3. **Avoid recursion**
    - ✓ Remove recursive function because it requires high-memory requirements.
4. **Avoid floating-point operations**
    - ✓ Floating point operations can be converted into fixed point operations.
5. **Use unsigned Integers**
    - ✓ Better to use unsigned integer variables.
6. **Avoid large library routines**
    - ✓ Large library routines needs to be avoided.
7. **Remove loop invariant code**
    - ✓ Code has no effect on the loop counter that kind of code needs to be removed from the loop.
    - ✓ Example

```
 for (i=0; i<5; i++)
{
    X=2*2;      // invariant code or unnecessary code
    Y=i*2;      // valid or relevant
}
```

3.7 Static Analysis Using Tools

- Manual review is a good mechanism for static analysis. The tools are also used to analysis of the source code.
- The tools are finding the number of defects like
    1. Whether the code is written as per the coding guidelines.
    2. Syntax mistakes.

3. Declaring variables & not using it.
4. Dead code or unreadable code.
5. Security.
6. Portability problems.
7. Total number of lines.
8. Number of comment lines.
9. Code complexity.

3.8 Tool for Readability Improvement /Indenting

- One of the challenge of developer is to maintain readability of code
- Linux/Unix system provides **indent** command to improve the readability of the code
- The indent command has following options

Option	**Description**
-bad	Blank lines
-bap	Blank lines after each functions
-bbb	Blank lines before comments
-bl	Braces after if statement
-br	Braces on if statement

3.9 Portability Testing Tool

- In UNIX, the **lint** command is used to check the portability of the code.
- The lint command provides the following messages
  - ✓ Unused variables and functions
  - ✓ Variables which are used first and set later
  - ✓ Unreachable break statements
  - ✓ Function that returns a value which is never used
  - ✓ Type casting problems
  - ✓ Non-portable characters use
- Example: $ lint hello.c

3.10 Symbolic Execution

- It is also called as **symbolic evaluation**.
- In this method, the program is executed with **symbolic data**.
- Example 1:

```
int x,a,b; //Here x,a and b are variables
x=a+b;  //X is a+b
```

- Example 2:

```
if(condition)  then
     Statements
 else
     Statements
```

- Limitation
  - Overflow cannot be detected.

UNIT-IV
DYNAMIC TESTING & TEST CASE DESIGN TECHNIQUES

4.1 Dynamic Testing

- In dynamic testing, the code is executed and tested.
- Dynamic testing includes the following steps
  1. Review work products
  2. Identify test objectives
  3. Identify test specifications & carryout test design
  4. Design the test cases
  5. Document test cases
  6. Execute the test cases
  7. Generate incident report
  8. Log the defects

4.2 Review Work Products

- Before testing the tester should study about the work product, these work products have been written during development.
- The work products are
  1. SRS document
  2. Design document
  3. Source code
  4. User manual
- Tester needs to develop the test specifications based on the documents.
- Test specification includes

Functional specification

- Design specification
- Code specification.

4.3 Identify test objectives

- Test objectives defines the various quality parameters, these parameters needs to be test.
- The quality parameters are
  1. Usability
  2. Reliability
  3. Performance
  4. Code coverage
  5. Portability
  6. Security
  7. Safety
  8. External Interface
  9. Conformance to a Standard.

4.4 Identify test specifications and carry out test design

- Test specification describes how to conduct testing process and managing the testing process.
- Test specification is obtained from the work product documents
- **The following are the steps in test design**
  1. Identification of test environment
  2. Identification of hardware platform in case of embedded system
  3. Identification of testing tools
  4. Study the specification and data
  5. Test script generation

4.5 Design test cases

- **Test case design includes the following steps**
  1. Identify the test inputs.
  2. Identify the test data.
  3. Work out the test procedure.
  4. Write any test scripts, if required.
  5. Identify special drives
  6. Specific condition or activities must be present before process begin
  7. Identify the entry criteria
  8. Identify the expected result
  9. Identify the exit criteria.

4.5.1 Guidelines for designing test cases

1. Test case should be simple. It should be only one expected output.
2. Test case should be reduces the errors
3. Test case does not have redundancy
4. The tester should identify valid inputs & invalid inputs
5. The tester identifies functional requirements like parameters
6. The tester identifies the checklists.

4.6 Test case design techniques

- There are 3 techniques for designing test cases

1. Black box test case design technique
2. White box test case design technique
3. Experience based test case design technique.

4.6.1 Black box test case design technique

- It has the 7 techniques
  1. Specification based Testing
  2. Equivalence partitioning
  3. Boundary value Analysis
  4. Decision tables
  5. Cause effect Analysis
  6. State transition diagrams
  7. Use case scenarios

1. Specification based Testing

- ✓ Test cases are design based on the "**specifications**".
- ✓ Input and output values are verified for each specifications
- ✓ **Example 1-**

Problem

To find given number is prime or not prime. Consider "n" is variable.

Test Case 1 : n=10
Expected Result is "Not Prime"
Test Case 2 : n=11
Expected Result is "Prime"

- ✓ **Example 2 –**

Problem

To find interest amount
1000> and <10,000 → 5% interest
10,000 >= and <=15,000 → 10% interest
15,000> → 15% interest

Test Case 1 : Principle Amount is Rs.500
Expected Result is "Not eligible for loan"
Test Case 2 : Principle Amount is Rs.5000 and time - 2 months
Expected Result : Interest Amount is Rs.500
Test Case 2 : Principle Amount is Rs.12,000 and time - 4 months
Expected Result : Interest Amount is Rs.4800
Test Case 2 : Principle Amount is Rs.16,000 and time - 5 months
Expected Result : Interest Amount is Rs.12,000

2. Equivalence partitioning

- ✓ It is also called as **equivalence class partitioning**.
- ✓ This technique is used to identify all possible valid inputs and invalid outputs,
- ✓ This technique divides the test condition into different 'groups'.

- ✓ In this technique ,inputs of the software are divided into groups, that are expected to exhibit similar behaviour.
- ✓ **Example :1**

Assume **age** text box is accepting value between 18 and 50

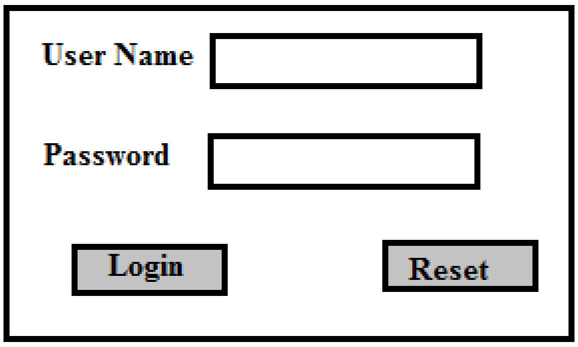
Age ☐ Condition is 18 to 50

Test case 1 : age 15 Expected Result is “**Invalid Age**”

Test case 2 : age 20 Expected Result is “**Valid Age**”

Test case 3: age 56 Expected Result is “**Invalid Age”**

- ✓ **Example :2 Login Page**

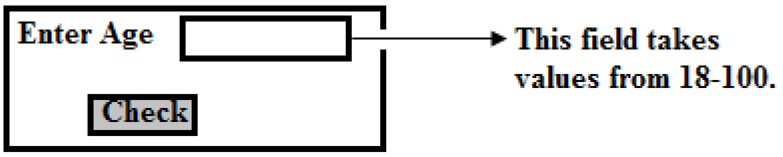


Consider the Correct User name : admin and Password : admin123. Test case is

User Name	Password	Expected Output
admin	admin123	Login Successful
ADMIN	admin123	Login Unsuccessful
admin		Login Unsuccessful
admin	ADMIN123	Login Unsuccessful

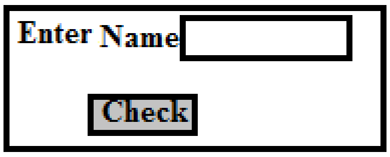
3. Boundary value Analysis

- ✓ This technique is used to identify boundary value of the input of software.
- ✓ Each inputs has its maximum and minimum values, these maximum and minimum values are the boundary values.
- ✓ In this technique, test case can be design to count both valid and invalid boundary values.
- ✓ **Example -1 To check the valid and invalid age for voters**



- Here the boundary value is 18 and 100.

- ✓ **Example -2 The Employee name must be at least 10 character and maximum 20 character**

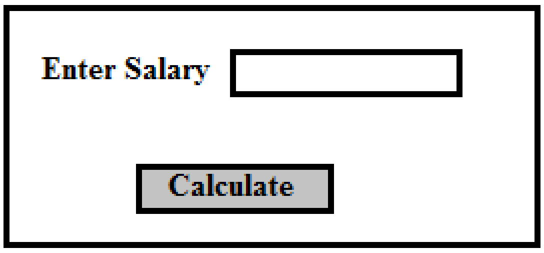


Inputs	Expected Result
Vijaya Prakash	Valid Name
Revanashiddappa timmanavar	Invalid Name

4. Decision tables

- ✓ Decision table technique is used to design test cases when there is many conditions are present in the software.
- ✓ This technique design the test cases to ensure that all the conditions are tested at least once.
- ✓ **Example – 1 To calculate income tax**

Salary	Tax rate
<=2,50,000	Nil
2,50,000 > and <=5,00000	10%
5,00,000 > and <=10,00,000	20%
>10,00,000	30%



Input	Expected Result
3,00,000	3,000
6,00,000	12,000
15,00,000	45,000

5. Cause effect Analysis

- ✓ This technique is used to identify input (cause) and action(effect) and draw the cause effect graph. This graph can be converted into cause-effect table
- ✓ This technique is used to generate the test cases based on the input and action
- ✓ The following are the steps
  1. Recognize and describe the input and actions
  2. Draw the cause effect graph
  3. Convert the graph to cause effect table
- ✓ **Example – 1 Consider the Withdraw form verification in bank account**

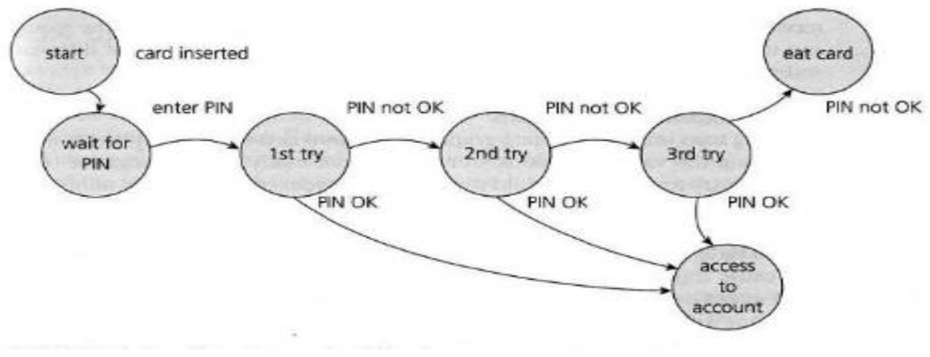
Cause 1	Account Number is correct
Cause 2	Signature on the withdraw form matches with bank account database
Cause 3	Enough money is available in the account
Action 1	Given money to customer
Action 2	Inform the customer no enough money in the account
Action 3	Call the police and inform he is fraud case

- ✓ **Test cases are**

Test case 1	Correct account number, signature matches and enough money
Test case 2	Correct account number, signature matches and no enough money
Test case 3	Correct account number, signature matches does not matches and enough money
Test case 4	Incorrect account number, signature matches and enough money
Test case 5	Incorrect account number, signature matches and no enough money
Test case 6	Incorrect account number, signature matches does not matches and enough money
Test case 7	Incorrect account number, signature matches does not matches and no enough money

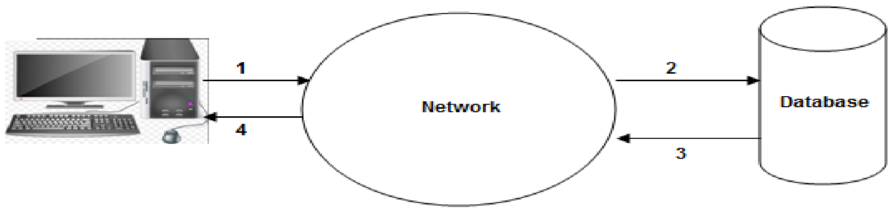
6. State transition diagrams

- ✓ It is used to identify the different states of a system.
- ✓ This technique is used to design the test cases to ensure that each and every state is tested.
- ✓ **Example -1 ATM System**



7. Use case scenarios

- ✓ This technique is used to design test cases to ensure that all actions are executed one after the other and desired result is obtained.
- ✓ **Example -1**



- Stage 1 : User must enter details
- Stage 2 : Details must processed in the database server
- Stage 3 : Response sent to the user system
- Stage 3 : Response received by the user system

4.6.2 White-box Test case design techniques

- ➢ White box testing is used to test the "**internal structure**" or **working of an application.**
- ➢ There are 4 techniques are available in white-box testing.
  1. Code Coverage
  2. Internal Boundary Value
  3. Structural Testing at Module level
  4. Structural Testing at System level

1. Code Coverage

- ✓ This technique is used to design the test cases based on the code.
- ✓ The "**quality of software"** can be identified when code coverage executed at system level
- ✓ Code Coverage involves
  1. Statement coverage
  2. Condition converge
  3. Path Coverage
- ✓ **Example –**
  - Tester can test the code by using code coverage tool, tester can identify which line of code is not executed.
  -

2. Internal Boundary Value

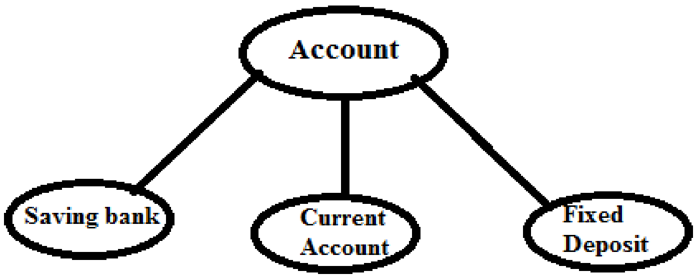
- ✓ In this technique, test cases can be designed by apply the boundary value of software.
- ✓ This technique is used to find whether the application is accepting the expected range of values and rejecting the values which fails out of range.
- ✓ **Example –** Consider the following loop

```
for(i=0;i<=100;i++)
{
    Statements;
}
```

  - This loop will execute 101 times, does the programmer really want 101 times or only 100 times?

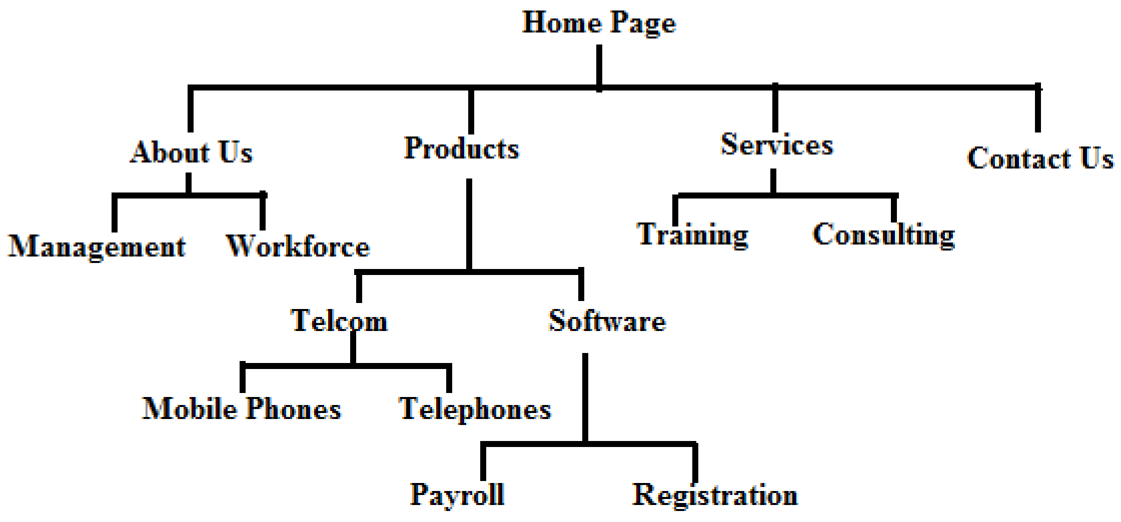
3. Structural Testing at Module level

- ✓ In this technique, test cases can be design based on functional module.
- ✓ In OOP concept, functional modules are designed.
- ✓ Tester can integrate all modules into functional modules and takes the testing at modules.
- ✓ The testing at module level is performed with knowledge about branches and paths in modules.
- ✓ **Example –** Consider the banking system



- There is one abstract super class called **Account** and classes called **Saving bank, Current Account and Fixed Deposit** are derived classes.
- Here, programmer cannot possible to create object for **Account** because it is abstract class, he can create object for type **Saving bank, Current Account and Fixed Deposit.**
- To test this module, tester can form accounts of various types and check the methods.

4. **Structural Testing at System level**
   - ✓ In this technique, test cases can be designed based on the complete system.
   - ✓ It is also used to design the test cases for integrated modules of software(master phase).
   - ✓ In this technique, testing can be done at complete system, it consists of number of modules. These modules are interconnected.
   - ✓ **Example** – Consider the structure of web sit



- There are 4 links from **home page**
  1. About Us 2. Products 3. Services 4. Contact Us
- Here tester needs to test all links and ensure that there is no missing links.

4.6.3 Experience based test case design technique

- ➢ This technique is used to select test case based on the testers experience and knowledge.
- ➢ There are two techniques
  1. Error Guessing
  2. Exploratory Testing

1. Error Guessing

- ✓ Error guessing comes through experience.
- ✓ In this technique, tester can easily guess what type of mistakes done by the developer during development of software.
- ✓ This technique is always applicable to experience tester.

2. Exploratory Testing

- ✓ Testing can be carried out without using the pre-defined test cases is called exploratory testing.
- ✓ In this technique, tester itself creating the test cases during the testing process.
- ✓ In this technique, test cases are designed and executed in real time without using built-in tools.
- ✓ In this technique, the test cases are designed and executed in "**real-time**".

4.6.4 Test cases for finger printing recognition System

- ➢ Fingerprint recognition system is compiling the given entry with free stored finger print data analysis.
- ➢ It includes the following test cases
  1. Testing the valid test cases
  2. Testing the invalid test cases
  3. Testing the performance of system.
  4. Testing the accuracy of system
  5. Testing the tolerance level of the system
  6. Testing the tolerance of the system
  7. Testing the security of the system

4.7 Document Test cases

- ➢ Test cases are need to be documented by giving details of design and execution of software.
- ➢ Document test case includes
  1. Test case ID
  2. Test case description
  3. Principle used for test case design
  4. Revision history
  5. Author name
  6. Functionality to be tested
  7. Test pre-condition and post-condition
  8. Test procedures
  9. Expected results

4.8 Incident Report OR Anomaly Report

- During the execution of test case, if the expected result and actual result does not match, then tester will generate a report called as "**incident report".**
- The incident report is also called as "**anomaly report**".
- The incident report includes the following details
    1. Data of incident report.
    2. Author of incident report.
    3. Description of the incident report.
    4. Executed result.
    5. Actual result.
    6. Priority assigned to the incident report.
    7. Present status.
    8. Change history
    9. Date of rectification and closed.

4.9 Defect Log <u>OR</u> Log the Defect

- Defect log is a one kind of recorder to store the defect. It has to be maintained for every project.
- Each defect is entered into the defect log.
- Test manager will study the defect log details and assign priority for rectification
- Defects in defect log have been rectified by the developer.

4.10 Test Document Standards

- Test documentations are
    1. Test specification document
    2. Test case design document
    3. Incident Report
    4. Defect Log
- The IEEE standards can be used for the documentation of the various work products. Those are
    1. IEEE 829-1998 → Standards for test documentation.
    2. IEEE 1028-1997 → Standards for test reviews.
    3. IEEE 1044-1993 → Standards for classification for software anomalies.
    4. IEEE 1044-1998 → Guide to classification for software anomalies.

4.11 Formal Method of Test

- The formal method of testing is used to improve the quality of software.
- Testing method is used to testing the product in two ways.
    1. The software has no error.
    2. Software must match to the user requirements.
- This method represents the user requirement and expected result.
- It shows the working flow of product.

UNIT-V

MANAGING THE TESTING PROCESS

5.1 Management Commitment

- ➢ The management commitment improves quality testing by providing necessary policy guidelines, budget and manpower.
- ➢ This management identifies the area of improvement and **quality goals.**
- ➢ It also measures the **current levels of performance**. This performance can be obtained by using following mechanisms
    1. Customer surveys
    2. Historical records
    3. Quality can takes from competition
    4. Industry standards benchmarks
    5. Own management criteria

1. **Customer survey**
    - The product quality can be obtained from the **customer feedback.**
2. **Historical records**
    - The product quality can be obtained from the records of bugs given by customer.
3. **Quality takes from competition**
    - In competition environment, organization growth, survive follows the market trends for observing the quality.
4. **Industry standards benchmarks**
    - Organization commit itself is a **six sigma quality**, total quality management. These management benchmarks can consider for implementation.
5. **Own management criteria**
    - Each management consists of some criteria for measuring **quality.** Some criteria are
        - ✓ Large customer base - Repeat orders from same customers.
        - ✓ Productivity of employees - Some of the modules are considered
        - ✓ Capability Maturity Module (CMM)
        - ✓ Capability Maturity Module Integration(CMMI)

5.1.1 Organization Structure

- ➢ The project manager controls the development and quality team and brings succeed to various groups like **domain experts, development experts, marketing experts and quality experts.**
- ➢ Organization structure is divided into two types
    1. **Hierarchical Structure**- The project manager will control entire product.
    2. **Flat Structure**- All individual work together as a team without any hierarchy.
- ➢ The management representative (MR) will responsible for **quality assurance.**

5.2 Testing Process Management

- ➢ This management carries the work in following simple steps
  1. Plan your work
  2. Track the work
  3. After work is completed, find out the mistakes

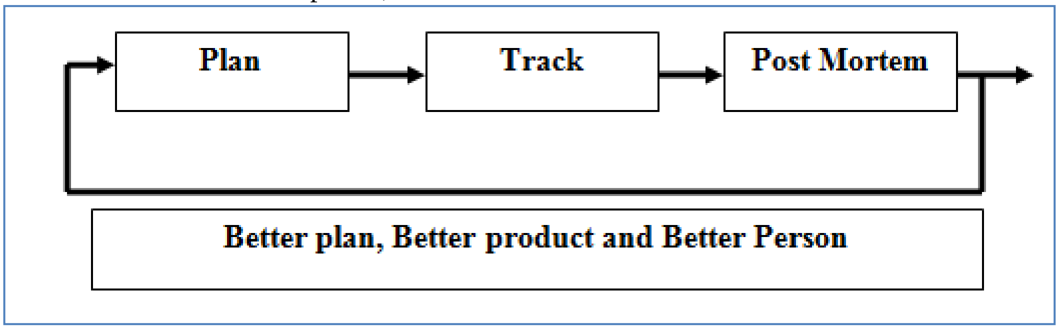


Fig 5.1 Quality Improvement Process

- ➢ It improves the process and delivery the better product.
- ➢ The **Plan-Do-Check-Act (PDCA)** module also delivery the quality software in 4 steps.

1. Plan – To carry out the activity as per plan.
2. Do – To carry out the activity
3. Check – To check whether the activity is going as per plan.
4. Act – To take any mid-course connections & improve the process.

5.2.1 Options for Managers

- ➢ The main activities of manager is
  1. Develop a software
  2. Buy Commercial-Off-The Shelf (COTS) software.

1. Develop a software

- ✓ Manager has to collect the user requirement & decides to develop software.
- ✓ The development team is responsible for developing software.
- ✓ Testing team is responsible for test the software.

2. COTS software

- ✓ COTS software is readily available that can be used without losing time.
- ✓ The following aspects need to be considered while purchasing it.

1. Selection of the software

- The software has been selected based on the requirements; compare the user requirements with features of COTS software.

2. Vender Selection

- The vender selection is based on the following factors
  1. Reputation of the vender
  2. Support to be provided for the software
  3. Local presence is available
  4. Licensing terms and conditions
  5. Training to be provided
  6. Cost of the software

3. Possibility to make modifications

- The vender must accept changes of software

4. Need for customization

- Installation of software product on customer site, it requires lot of customization cost.

5. Testing the COTS software

- The software has to be tested and ensure that, it meets all functions of requirements.

5.2.3 Testing Process Management Activities

➢ The manager has to carry out the following activities

1. Planning, budget and scheduling the test process.
2. Alignment of the test process to the project
3. Team formation
4. Infrastructure
5. Reviewing, Monitoring and risk management
6. Risk management
7. Metrics
8. Tracking the defects
9. Configuration management
10. Test closure and process improvement
11. Information Security.

5.3 Planning, budget and scheduling the test process

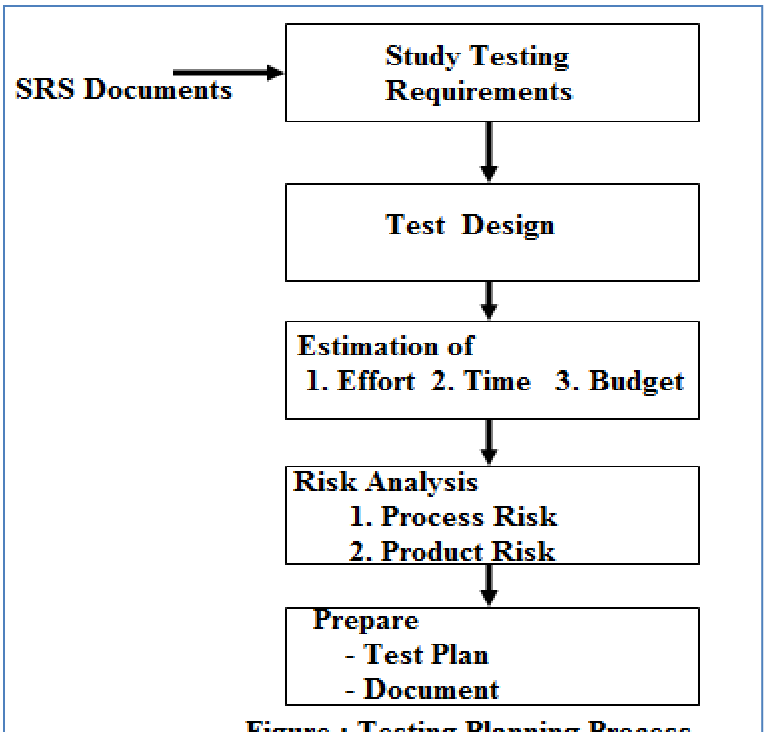


Figure : Testing Planning Process

➢ The effort, time and budget are planned for testing based on the project risk and product risk.

➢ **Test manager has to identify the objectives. The objectives are**
1. Find the defects and report to development team
2. Give the suggestions for improving software
3. To check the source code contains standards/guidelines
4. To check the non-functional testing like portability, security, safety etc…
5. To check whether the software meets functions and requirements of client.
6. Assigning task to testing team

➢ **Criteria for completion of testing**
- It is very important to identify the criteria for completion of testing at early stages of development.

➢ The budget, effort, scheduling has been carried out by using two approaches.

1. **Experience Based -** The budget, effort, time is estimated based on past experience.

2. Metrics Based - The metrics based is used estimate the effort in following way-
- Number of lines in function
- Number of classes and sub classes
- Number of tables, forms and reports
- Number of pages in websites

5.3.1 Test Plan

- The following is test plan format

Project Name	
Estimated start date	
Estimated end date	
Actual Start Date	
Actual End date	
Estimated Efforts (Hours/Months/Numbers of Testers)	
Estimated budget	
Test person name and responsibility	
Work product document	
Test requirement list	
Format for Defect Reporting 1. Defect Found 2. Type of Defect 3. Classification of Defects (Critical, Major, Minor) 4. Time for removal of defect 5. Whether the defect is removed	
Metrics to be Collected	

5.4 Alignment of the Process to the Project

- The alignment has been done by viewing the following aspects
  1. To check whether the software is either COTS software or built-in software.
  2. Check whether the software is developed in house or through subcontractor.
  3. Select software development life cycle.
  4. To check whether the language is either procedural language or object-oriented language
  5. To check the application domain such as network software, telecom software, embedded software.
  6. To check whether testing tool is used or not.

5.5 Team Formation

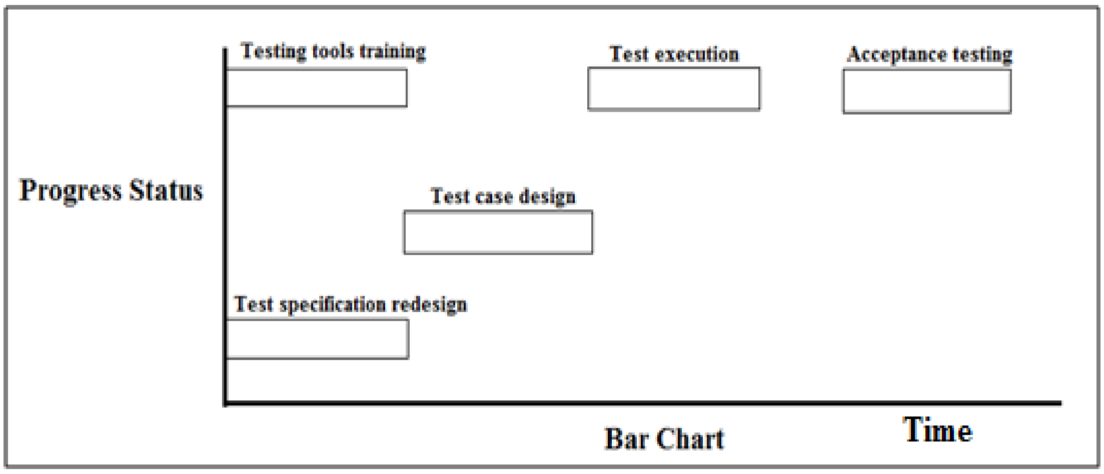
- The quality of testing and quality of product depends on the testing team.
- The team roles can be divided into 3 categories
  1. **Action-oriented roles -** Shaper, implementer, completer or finisher.
  2. **People-oriented roles -** Coordinator, team worker.
  3. **Cerebral roles -** Monitor, evaluator, specialist.
- The **team role** is very important for carrying team work. It includes profiles. There are 8 profiles
  1. Reporter advisor
  2. Creator-Innovator
  3. Explorer promoter

4. Assessor- Developer
5. Truster-Organizer
6. Concluder-Producer
7. Controller-Inspector
8. Upholder- Maintainer

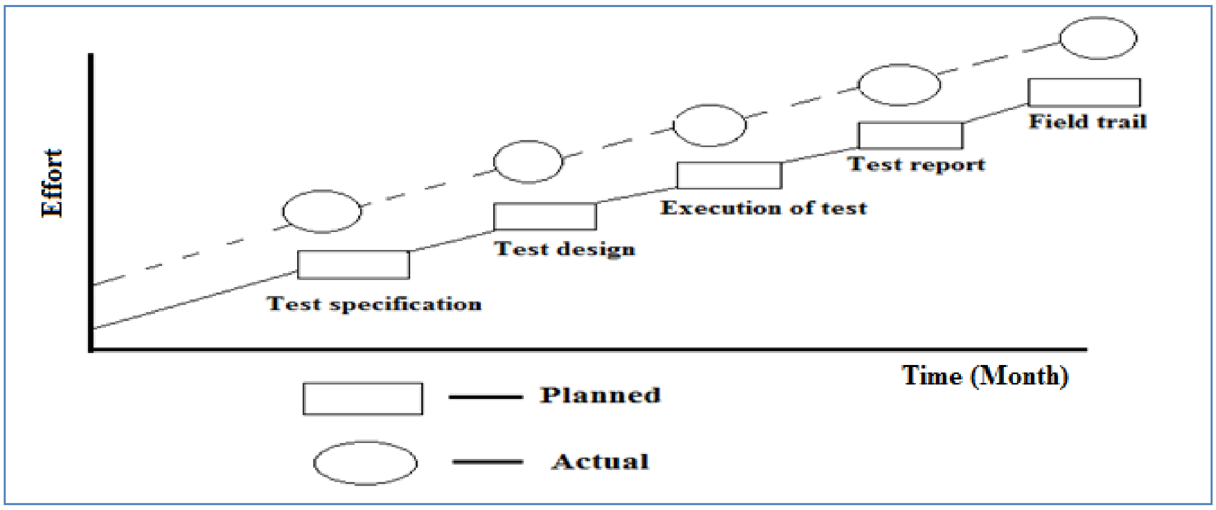
- **The test Engineer includes the following skill**
  1. Understanding the application domain concept
  2. Excellent knowledge of software quality
  3. Knowledge of software testing
  4. Experience in designing test cases
  5. Good oral and written communication skills
  6. Good inter-personal skills
  7. Excellent analytical abilities
- **Test engineer is responsible for**
  1. Study the work product and carry out the test specifications.
  2. Carry out the test design
  3. Test case design and generate test scripts.
  4. Document test cases.
  5. Test case execution and incident report.

5.6 Reviewing, Monitoring and Risk Management

- The manager has constantly review the work of the individual team members and monitor the overall progress and take corrective action.
- The **Gantt Chart** tool is used for reviewing the progress of project. This Gantt chart is also known as **bar chart**
- In bar chart, all activities are represented by using **bars**
- The bar chart is prepared based on the plan and checks each activities progress status.



- The **cost-schedule milestone graph** is also used to maintain estimated effort of each activity.



5.7 Metrics

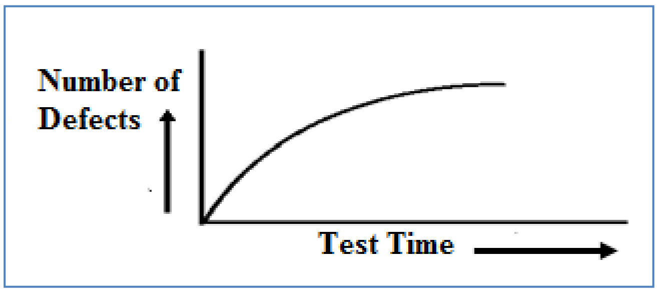
- The manager has to identify the following metrics.
  1. Estimated **effort, time and budget**. Actual values **of effort, time and budget** of the after completion of testing process
  2. Identify **number of lines, number of function, number of classes, number of tables, number of forms** in the database
  3. Number of **web pages and websites**
  4. Total number of defects found and identify defects in each category
  5. Total number of test cases executed and percentage of detected errors in test cases
  6. Cost of quality(failure cost, appraisal cost and prevention cost)
  7. Root cause of defects
  8. Reliability Metrics (MTBF, MTTR and MTTF)

5.8 Software Reliability

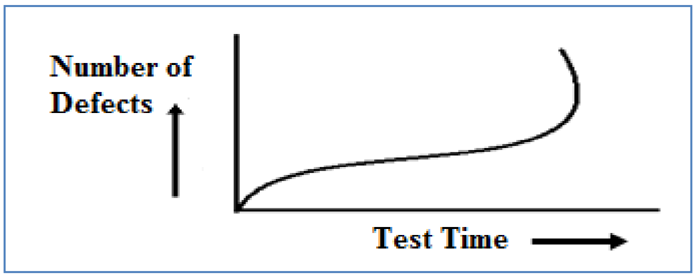
- The delivery of product is decided by metrics because metrics identifies the reliability of software
- There are 3 software reliability models
  1. Concave Model
  2. S-Shaped Model
  3. Two-Stage Model

1. Concave Model

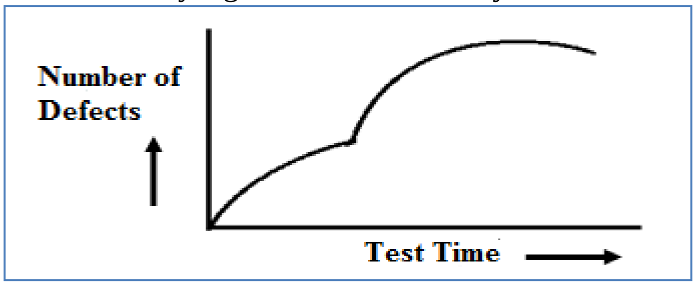
- These models are not efficient because assumptions are made in the model



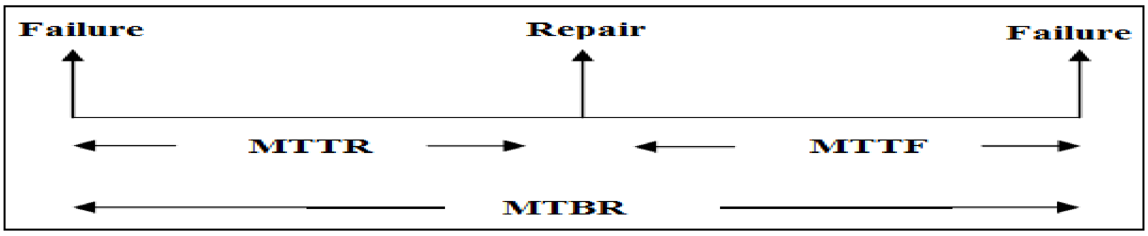
2. **S-Shaped Model**
   - These models are not efficient because assumptions are made in the model
   - There is an increase in the **<u>value of defect</u>**



3. **Two-Stage Model**
   - This model identify significant functionality of the software



➢ The reliability of software is identified by using following parameters
   - MTTF (Mean Time To Failure)
   - MTTR (Mean Time To Repair)
   - MTBF (Mean Time Between Failure)



➢ The MTBF is used to measure the reliability of the software product

5.9 Defect Tracking

➢ During testing process, all defects are stored in defect log (or incident report). The test manager has defines process to removing the defects.

➢ This process involves the following steps:

1. **Recognition of defect**
   - If there is difference between the expected result and actual result, then it is defect. This defect is recorded into **defect log**. The developer can solve the defect.
2. **Investigation, why defect occurred**
   - The manager has to finds the root cause of defect and assigns priority to defect and identify the person who will remove the defect.
3. **Defect removal**
   - When defect is removed by developer, the test manager need to verify whether defect is removed and declare that defect is removed

5.9.1 Classification of Defects

- The following are the classification of defects
    1. Critical Defect
    2. Major Defect
    3. Minor Defect

1. **Critical Defect**
    - Critical defect means that result may **crash the system or application**
2. **Major Defect**
    - Some of the defects are damages the some portion of code but not crashes the system
3. **Minor Defect**
    - Minor defects means spelling mistakes, incorrect syntax and incorrect ordering of the GUI tools

- Classification of defects is based on the functional/non-functional requirements
    1. Functional defect
    2. Non-functional defect
    3. Security defect
    4. Safety defect
    5. External-interface defect
    6. Portability related Defects
    7. Usability related Defects
    8. Testability related Defects

5.10 Configuration Management

- Configuration management is an activity, during testing process, tester finds the defects and initiates request for change
- The change request involves
    1. **Preventive maintenance**
        - Change of software makes that software will not fail in future
    2. **Adaptive maintenance**
        - It means change of requirements and change of software features
    3. **Corrective maintenance**
        - Change of software makes that software works correctly

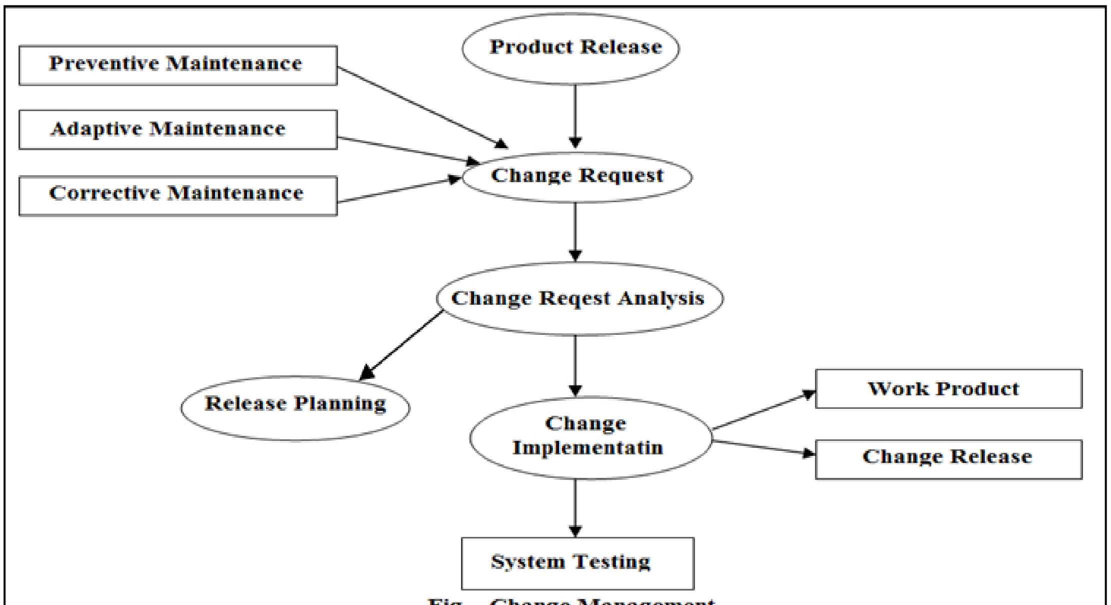
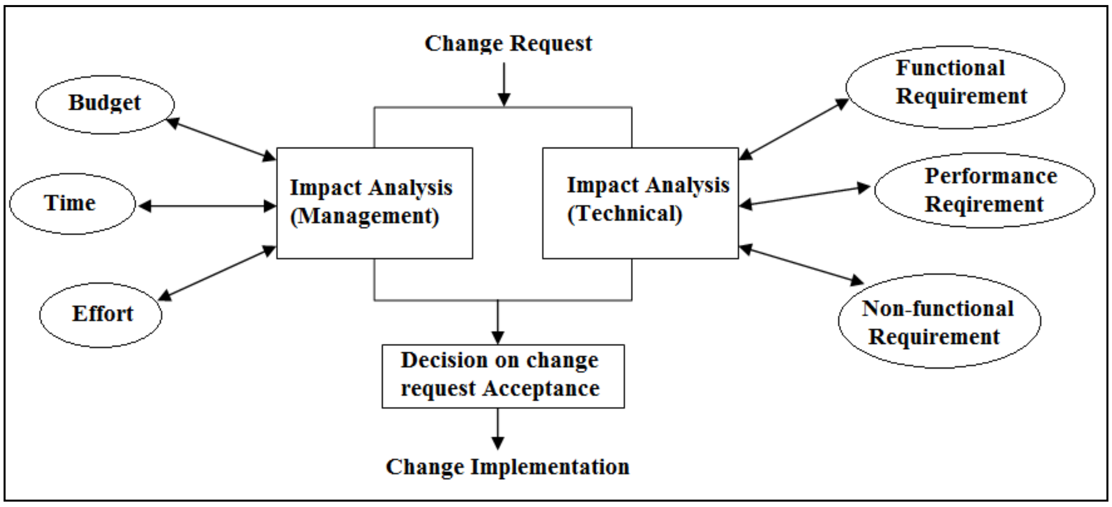


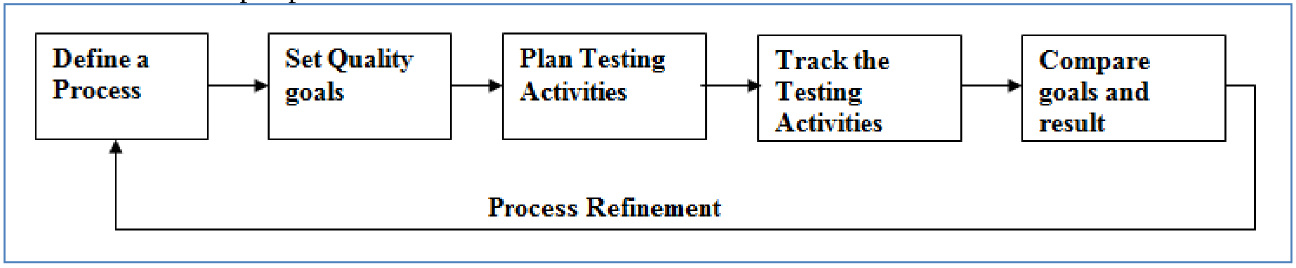
Fig - Change Management

- The changes are made not only in the code but entire product is changed and tests the entire software and released
- The changes of both **functional requirements and non-functional requirements** has been studied by **technical team**
- The changes of **management requirements like budget, time and efforts** has been studied by **management team**


Change Request
Budget
Time
Effort
Impact Analysis (Management)
Impact Analysis (Technical)
Functional Requirement
Performance Reqirement
Non-functional Requirement
Decision on change request Acceptance
Change Implementation

5.11 Test Closure and Process Improvement

- The manager and all team members will gain the lot of experience from project.
- This experience helps in improving the testing process
- The testing team defines a process and set of quality goals. The testing activities can be carried out as per plan.



- If testing process is unable to complete within the estimated time or budget, the manager has to find out the reason and need to improve the testing process.
- If there are more defects in design then design process has to be improved.
- If there are more defects in code, then programmer needs to be trained.
- If testing process is not executed well, than manager needs to trained.

5.11.1 Software Testing Maturity Model (SW-TMM)

- This model is used to improve the testing process systematically. It involves 5 levels:

  Level 1 – **Initial**

  Level 2 – **Phase definition**

  Level 3 – **Integration**

  Level 4 – **Management and Measurement**

  Level 5 – **Optimization and quality control**

Level 1 - **Initial**

The software process is characterized as “**ad-hoc**” and “**Chaotic**”.

1. Testing is done at ad-hoc and chaotic
2. Testing shows the software work correctly
3. It requires trained staff and testing tool

Level 2 - **Phase definition**

1. Testing is identified as separate phase
2. Testing is different from debugging
3. It uses basic testing methods
4. It shows that software meets requirements

Level 3 - **Integration**

1. Integrate testing into entire life cycle
2. Establish formal testing process like control reviewing, monitor.
3. Testing is based on system requirements
4. Tools are used for testing

Level 4 - **Management and Measurement**

1. Metrics are used for managing the testing process
2. Quality parameters are tested like reliability, usability, security etc…
3. Test cases are collected and recorded into database for reuse and regression testing
4. Defects are solved

Level 5 - **Optimization and quality control**

1. Testing process is well defined & managed.
2. Testing effort and cost are monitored.
3. Testing tools are selected for process
4. Testing process is continuously improved.

5.12 Information Security

- The test engineer and test manager has to provide security to the information system .
- A number of international standardization bodies and government organization has initiates the security to information system.
- It provides framework for **security assurance** similar to software quality assurance.
- The following are some of the major cyber laws was given different governments

1. **Health insurance portability and accountability Act (HIPAA)** was released by departments of health and human services. This act gives relation to information in following 5 areas -
    1. Administrative control
    2. Physical control
    3. Technical control
    4. Organizational requirements
    5. Policy, procedures and documentation requirements.
2. **Common criteria (ISO 15408) standard**, product is evaluated based on 5 evaluation levels
    1. Eval 1 - The product is tested for it functionality

2. Eval 2 - The product is tested structurally
3. Eval 3 - The product is methodically designed, tested and reviewed.
4. Eval 4 - The product is methodically tested and checked.
5. Eval 5 - The product is formally designed and tested.
6. Eval 6 - The product is formally verified, designed and tested.
7. Eval 7 - The product is semi-formally verified design and tested

3. ISO 17799 standard provides 4 security assurances

1. Organizational assurance – Internal to the organization
2. Service provider assurance – External third party suppliers
3. Trading partner assurance – Business partners
4. Insurance processing facility assurance – product suppliers.

UNIT-VI

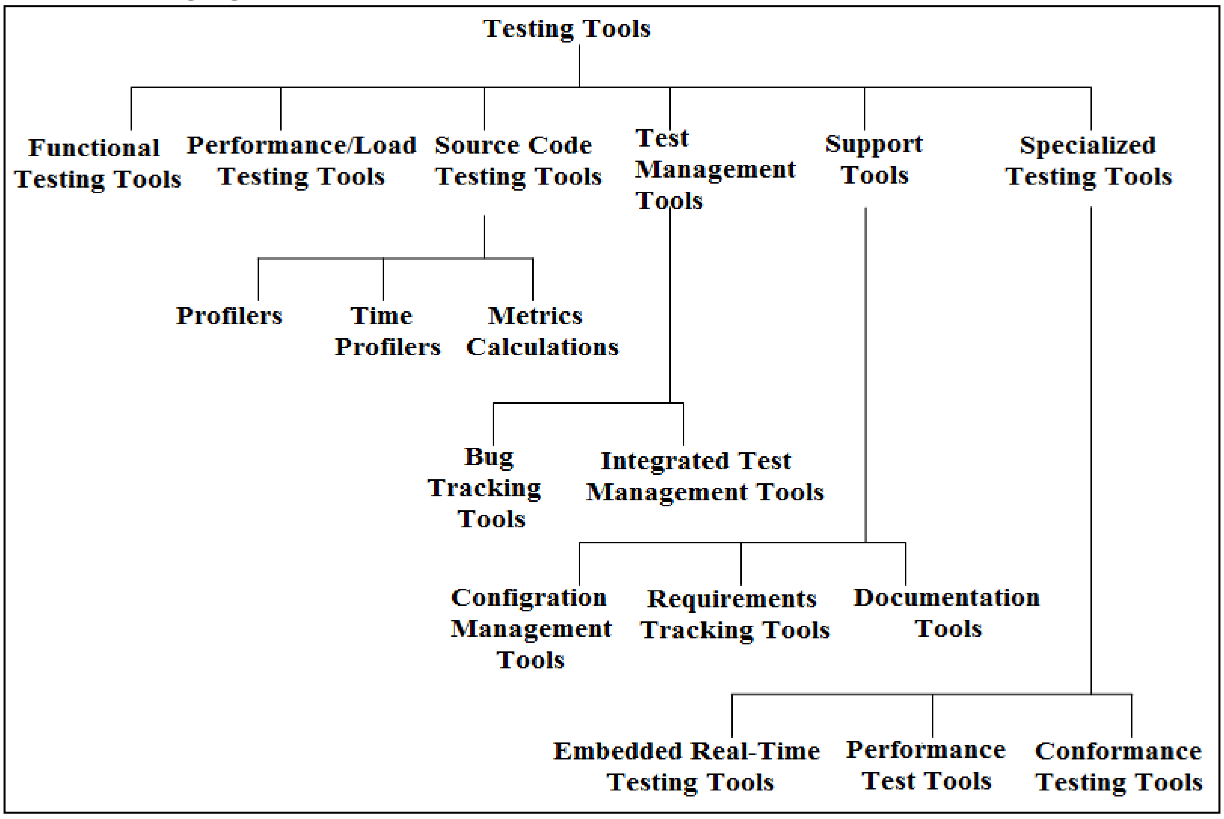
SOFTWARE TESTING TOOL & CODE OF ETHICS FOR SOFTWARE PROFESSIONAL

6.1 Need for Tools OR Advantages of Tools

- Tools can improve the quality and reliability of software.
- By using tools, testing of software can be done in different environments with less number of testers.
- It reduces the effort, budget and time required to testing process.
- Testing process can be planned and managed effectively
- Testing reports can be generated automatically
- Testing documents (Test cases, log sheet, test report) can be archived effectively.
- Managing the testing process is easy.

6.2 Classification of Tools

- The following figure shows the classification of tools


Testing Tools
Functional Testing Tools
Performance/Load Testing Tools
Source Code Testing Tools
Test Management Tools
Support Tools
Specialized Testing Tools
Profilers
Time Profilers
Metrics Calculations
Bug Tracking Tools
Integrated Test Management Tools
Configration Management Tools
Requirements Tracking Tools
Documentation Tools
Embedded Real-Time Testing Tools
Performance Test Tools
Conformance Testing Tools

1. **Functional / Regression Testing Tools**
   - ✓ These tools are used to test the web applications such as websites, GUI applications etc
   - ✓ These tools are used to carry out the **black-box testing**

   Note : More Details refer 2.2.4 and 2.5 topics of Unit-II
2. **Performance Testing Tools**
   - ✓ These tools are used to test the database applications, website, Client-Server applications etc
   - ✓ These tools are used to carry out the performance testing or stress testing.

   Note : More Details refer 2.15 Topic of Unit-II

3. **Source Code Testing Tools**
   - ✓ These tools are used to test the source code of the software applications.
   - ✓ These tools are used to carry out the white-box testing.
   - ✓ These tools are divided into two categories
     1. Static Analysis Tools
     2. Dynamic Analysis Tools

   1. **Static Analysis Tools**
      - Static analysis tools are used to test the source code without executing it.
      - These tools are used to find the following defects
        1. Whether code standards /guidelines are followed or not.
        2. Whether code is portable
        3. Variables declared but unused
        4. Dead code or unreachable code
        5. Obtaining the coding metrics.
   2. **Dynamic Analysis Tools**
      - Dynamic analysis tools are used to find out the major defects in code by executing it.
      - These tools are used to detect memory errors such as memory allocation errors, memory leaks, buffer overflow etc.

4. **Test Management Tools**

   Note : Refer the 6.2.1 sub topic
5. **Supporting Tools**
   - ✓ These tools helps the developers/tester in different phases of software development.
   - ✓ The following are the some of the supporting tools
   1. **Requirement management tools**
      - These tools track the requirement whether the requirement is converted into design.
      - These tools also check whether requirements are tested or not.
   2. **Design Tools**
      - These tools are used to carry out the analysis and design of the software.
   3. **Traceability Tools**
      - These tools help in tracing whether all specifications are tested or not.
   4. **Unit Testing Tools**
      - These tools carry out the unit testing.

6.2.1 Testing Process Management Tools

- ➢ Testing management tools provides the option to do the testing process in systematic manner.
- ➢ Testing management tools provides **defect tracking** option for project manager. Defect tracking will help to track the defects.
- ➢ These tools also provide the option to storing the information about the defect in a database and periodically updating the status.
- ➢ These tools includes the following details
  - Who detected the defect?
  - Entry - Criteria
  - Exit - Criteria

- Date and time when the defect was found
- Priority of the defect
- To whom defect rectification has been assigned.
- Status of defect

6.3 Risk Associated with the Tools OR Disadvantages of Tools

- ➢ The management does not estimate the time, effort and budget required to introducing a tool into the development process because management has to spend budget on training the employees on the tool, carrying out pilot project and if the pilot project is successful, tool can be introduce in the development process.
- ➢ After completion of pilot project, the success criteria must be defined clearly.
- ➢ After introducing the tool, still some application required manual testing. The tool may not be capable of doing all types of testing.
- ➢ Unrealistic expectations from the tool

6.4 Does your Organization Need Tools

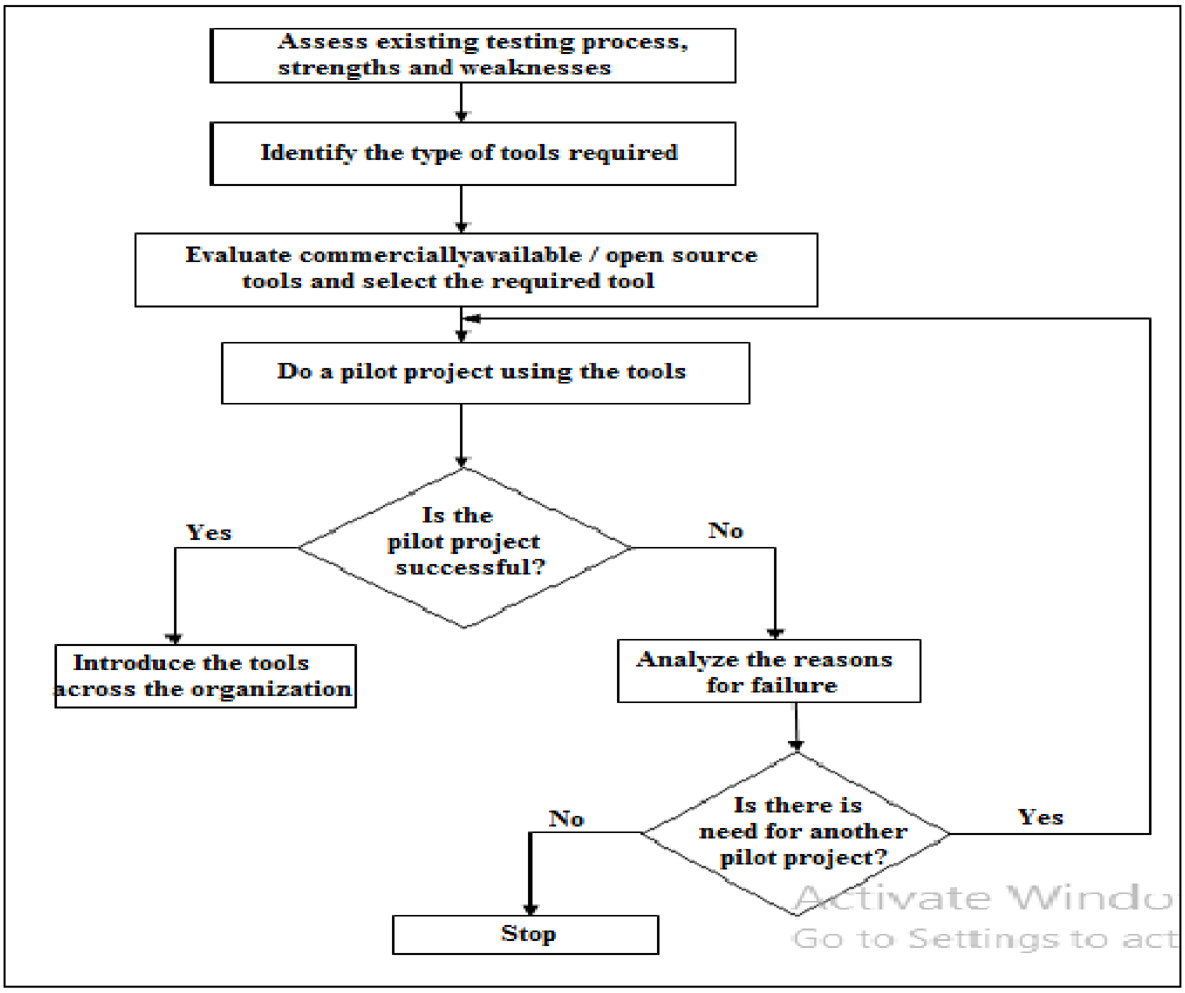
- ➢ Tools will improve the testing process. The budget, effort and time will spend on corrective maintenance in manual testing is more than the cost of testing tools.
- ➢ By using tools, test engineer can do the productive work by generating the test cases.
- ➢ Testing tools remove the manual work and test engineers will enjoy the testing process
- ➢ In manual testing, it is difficult to bring out a quality product if performance testing is part of requirements. So tools are required.
- ➢ Organizations will believe in process-oriented approach when testing tools are used.
- ➢ If project teams are located at different locations, web-based management tool will be very effective for testing process.

6.5 Selecting Tools

- ➢ Selecting tools can be done based on the following criteria.
  1. If software wants modify, then need to use functional/regression testing tools.
  2. If Client-Server or web application, then need to use load testing.
  3. Select the tools based on the environments (like Windows, Linux, solaris etc)
  4. If source code needs to test, then source code testing tools
  5. If organization wants to implement process-oriented testing, then testing process management tools must be used.
  6. The manager has check whether tool vendor is in local or different country or city.
  7. Whether vendor will provide training to employees about tools
  8. What is the cost of the tool?
  9. What is the cost of yearly upgrades?
  10. What about the license of the tool?

6.6 Introducing the Tools in the Testing Process

- It is very important for management to follow a systematic procedure for introducing a tool in the organization.


Assess existing testing process, strengths and weaknesses
Identify the type of tools required
Evaluate commerciallyavailable / open source tools and select the required tool
Do a pilot project using the tools
Is the pilot project successful?
Yes
No
Introduce the tools across the organization
Analyze the reasons for failure
Is there is need for another pilot project?
No
Yes
Stop

Activate Windo
Go to Settings to act

- The following steps are considered during the introducing the tools in the organization
    1. Management must identify the strength and weakness of the present testing process.
    2. Depending on the project requirements, identify the types of tools required for organization.
    3. Evaluate whether tools are commercially available or freely available and select the tools
    4. Test the pilot project in selected tools and identify effectiveness of the tools
    5. If the pilot project is successful, introduce the tools for other projects. If pilot project is not successful, then analyze the need for another pilot project.
    6. If another pilot project is required, then process the execution of pilot project otherwise stop the introducing tools in the organization.

6.7 Code of Ethics for Software Professionals

6.7.1 Human Ethics

- Every human some responsibilities towards the **society and surrounding environment**.
- Every human has to follow certain principles, rules and regulations set by the government for country growth.
- If rules framed by government are not correct, every human has to fight against it. **Ethical issues are highly subjective**.
- Every human must encounter conflicts as **what is good** and **what is bad**

6.7.2 Professional Ethics

- A professionals are doctors, engineers or charted accountant has further obligations to the society because professionals play a greater role in designing the further for the rest of the society.
- IEEE is the largest professional body of electrical and electronics engineers, framed the code of ethics as follows
    1. To accept responsibility in making engineering decisions consistent with safety, health and welfare of the public.
    2. To avoid real or perceived conflicts of interest.
    3. To be honest and realistic in stating claims
    4. To reject bribery in all its forms.
    5. To improve the understanding of technology.
    6. To maintain and improve our technical competence.

6.7.3 Ethical Issues in Software Engineering

- Software engineers have the power to do good or bad to the society. **Every software professionals has to use the knowledge and skills for the benefits of the society**.
- Every software professionals has to follow a **code of ethics with respect to improve the quality of software.**

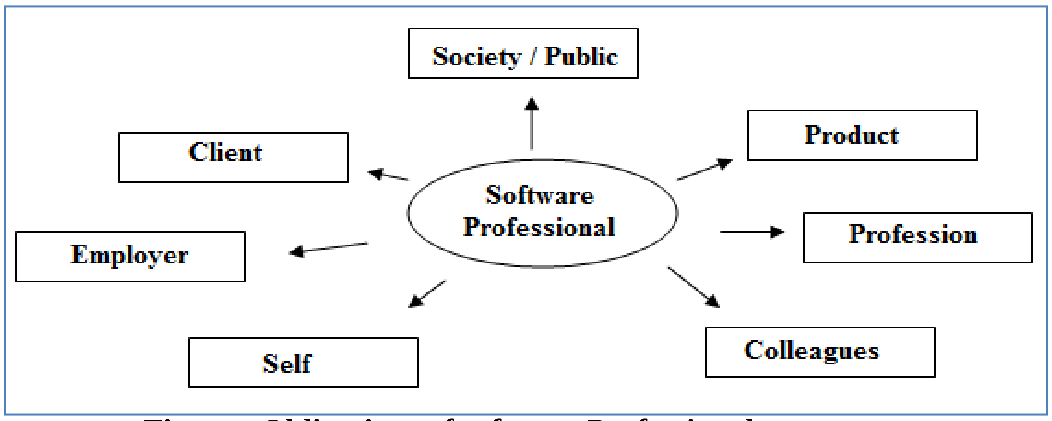


Figure : Obligations of software Professionals

- Every software professional has obligations to
    1. Society
    2. Self
    3. Profession
    4. Product
    5. Employer
    6. Client

7. Colleagues

- These obligations have to be fulfilled by every person as
  - A human being
  - A professional
  - A Software engineering professional

6.7.4 Software Engineering Code of Ethics and Professional Practice

- Preamble consists of 8 principles. The following are the 8 principles has to follow the software engineers.
  1. Public
  2. Client and Employer
  3. Product
  4. Judgement
  5. Management
  6. Profession
  7. Colleagues
  8. Self

1. **Principle 1 : Public**
   - ✓ Accept full responsibility for their own work.
   - ✓ Moderate the interest of the software engineer, employer, client and users with public.
   - ✓ Approve the software only if it is safe, meets specifications, passes appropriate tests and does not harm the environment.
   - ✓ Consider issues of physical disabilities, allocation of resources, economic disadvantages.
   - ✓ Be encouraged to volunteer professional skills to good causes and contribute to public education.
2. **Principle 2 : Client and Employer**
   - ✓ Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education
   - ✓ Do not use the software obtained illegally or unethically.
   - ✓ Ensures that all documents are approved by authorized person.
   - ✓ Identify documents, collect evidence, and report to the client or employer promptly.
   - ✓ Keep any confidential information gained in their professional work.
3. **Principle 3 : Product**
   - ✓ Client/Employer/public should accept high quality, acceptable cost and reasonable schedule of product.
   - ✓ Ensure proper and achievable goals and objectives for any project on which they work.
   - ✓ Define and identify ethical, economic, cultural, legal and environmental issues related work projects
   - ✓ Ensures an appropriate method is used any project.
   - ✓ Work must have professional standards
   - ✓ Maintain the integrity of data.
   - ✓ Treat all forms of software maintenance with the same professionalism as new development.
4. **Principle 4 : Judgement**
   - ✓ Temper all technical judgements and maintain human values.

- ✓ Only endorse documents either prepared under their supervision or within their areas of competence
- ✓ Maintain professional objectivity with respect to any software
- ✓ Not engage in illegal activities like double billing or other improper financial practices.
- ✓ Disclose to all concerned parties those conflicts of interest that cannot be escaped.
- ✓ Refuse to participate, as member or advisors or any professional body concerned with software related issues.

5. **Principle 5 : Management**
   - ✓ Ensure good management for any project which includes effective procedures for promotion of quality and reduction of risk.
   - ✓ Ensure that software engineers are informed of standards before being held them.
   - ✓ Assign work based on the education and experience
   - ✓ Offer fair and just remuneration
   - ✓ Do not ask software engineer to do inconsistency with the code.
   - ✓ Not punish anyone for expressing ethical concerns about a project.
6. **Principle 6 : Profession**
   - ✓ Develop an organization environment ethically.
   - ✓ Promote public knowledge of software engineering.
   - ✓ Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications
   - ✓ Not promote their own interest at the expense of the profession, client or employer.
   - ✓ Avoid association with businesses and organizations which are in conflict with this code
7. **Principle 7 : Colleagues**
   - ✓ Encourage colleagues to improve in profession
   - ✓ Assist colleagues in professional development
   - ✓ Review the work of others in an objectively and proper documented
   - ✓ Give a fair hearing to the opinions or complaints of a colleague.
   - ✓ Take opinion from other expert professionals if wants own areas of competence.
8. **Principle 8 : Self**
   - ✓ Develop the knowledge in the area of analysis, specification, design, development, maintenance and testing of software.
   - ✓ Improve their ability to create safe, reliable and useful quality software at reasonable cost and within reasonable time,
   - ✓ Improve their ability to produce accurate and well-written documentation.
   - ✓ Improve their understanding of the software and related documents of the particular work
   - ✓ Not influence others to undertake any action that involves a breach of this code

6.7.5 Ethical Issues :Right versus Wrong

- ➢ Every software engineers should always use their own judgment decide what is right and what is wrong, but judgment should be based on realistic and background information.
- ➢ There are two different view points
  1. Right
  2. Wrong
- ➢ Decide yourself what is right and what is wrong.
- ➢ Every decisions of professionals ensures that no damage is done to human life and society.
- ➢ Every software professional's main objective must be improve the quality of life,