# SOFTWARE ENGINEERING AND TOOLS

| | |
|---|---|
| **Subject** | **: Software Engineering & Tools** |
| **Subject Code** | **: CSE605** |
| **Total Hours** | **: 42** |
| **Full Marks** | **: 80 + 20 = 100** |

Module I

Overview of System Analysis & Design , Business System Concept, System Development Life Cycle, Waterfall Model , Spiral Model, Feasibility Analysis, Technical Feasibility, Cost- Benefit Analysis, COCOMO model. 10 Hrs

Module II

System Requirement Specification – DFD, Data Dictionary, ER diagram, Process Organization & Interactions. 06 Hrs

System Design – Problem Partitioning, Top-Down And Bottop-Up design ;Decision tree, decision table and structured English; Functional vs. Object- Oriented approach. 06 Hrs

Module III

Coding & Documentation - Structured Programming, OO Programming, Information Hiding, Reuse, System Documentation.

Testing – Levels of Testing, Integration Testing, Test case Specification, Reliability Assessment . , Validation & Verification Metrics, Monitoring & Control. **Usability Testing**

**User Interface Testing:** What makes a Good UI? , Follows standards or Guidelines, Intuitive, Consistent, Flexible, Comfortable, Correct, Useful. Testing for the Disabled: Accessibility Testing: - It's the Law, accessibility features in software. 06 Hrs

**Web site Testing**

Web Page Fundamentals, Black-Box Testing: - Text, Hyperlinks, graphics, forms, object and other simple miscellaneous Functionality. Gray Box Testing, Black box testing, White Box Testing, Configuration and compatibility testing, Usability Testing, Introducing Automation

Module IV

Software Project Management – Project Scheduling , Staffing, Software Configuration Management, Quality Assurance, Project Monitoring. 06 Hrs

**The Future Software Quality Assurance :** Quality is free, testing and quality assurance in the workplace , software testing , Quality Assurance, other names for software testing groups, Test management and organizonal structures, Capability Maturity Model (CMM), ISO 9000 CASE TOOLS : Concepts, use and application. 08 Hrs

**Your Careers As a Software Tester:** Your job as a software tester, finding software testing position, gaining hands-on experience, Internet links, Professional Organizations.

# MODULE 1

## Overview of system analysis and design

Systems development is systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. Here, in this tutorial, we will primarily focus on −

- Systems analysis
- Systems design

### Systems Analysis

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

Analysis specifies what the system should do.

### Systems Design

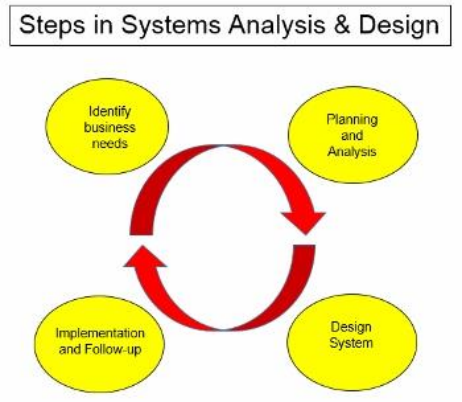
It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System Design focuses on how to accomplish the objective of the system.

System Analysis and Design (SAD) mainly focuses on −

- Systems
- Processes
- Technology

There are four main steps involved in systems design and analysis.




## 1. Identifying Business Needs

It is important for a company to weigh the pros and cons of changing their currents systems already in place. Identifying the value of the new system is incredibly important for this process to be successful. Just like building your dream home, you want to make sure it's an improvement from your current home.

Businesses must first identify what their needs are, and they can do this by asking the following questions:

- What value do we want to bring to the organization?
- Is it to improve efficiency in the workplace?
- How do we want to change our current processes?
- What results are we trying to achieve?
- What are the budgetary concerns?

This stage involves all levels of the organization, to collaborate the business needs and the overall ability to develop the new information system.

## 2. Planning and Analyzing

Once the business needs are set, planning and analysis begins. This process should begin by identifying the extent to which they are trying to use this new system. This process is similar to a blueprint one might have for a home, where you lay out the big picture. Part of determining this is identifying who the users of the new system will be and what the system will be used

for as well as addressing any privacy concerns. This initial part is important to make sure that the business objectives are being met, and it lays the foundation for the 'big picture'.

By looking into the needs of the users of the system, this helps identify the reach or span in which the new system will be used. It determines the cross functionality of the system and if it will be used for one or many departments.

Some questions one might ask include the following:

- What data should we store, and how does that, in turn, provide information to the company?

- Do we want the marketing department information to be integrated with the sales departments?

Asking questions like this will help to identify the 'who' and 'what' the system will be used for. Part of this stage should also address any privacy concerns the company may have as far as who should or should not have access to the information. Certain information is more sensitive than others and only certain users should be given specific access.

### 3. Designing a System

This phase will start by researching what necessary hardware and/or software is needed to use the system. At this point, the company will look at their physical structure and find ways to mimic this and any new changes in the new information system.

Questions asked during this phase include the following:

- How do we want to set up and store the data we need?

- How should we capture the data?

- How can we ensure that the information is accurate or complete?

Once this is answered, logical design is also incredibly important to make the system logical to daily users. This system should be used to improve organization systems, and consideration for ease of use and user trainability are essential in this phase.

## What is a System?

The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.

A system is “an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal.”

## Constraints of a System

A system must have three basic constraints −

- A system must have some **structure and behavior** which is designed to achieve a predefined objective.

- **Interconnectivity** and **interdependence** must exist among the system components.

- The **objectives of the organization** have a **higher priority** than the objectives of its subsystems.

For example, traffic management system, payroll system, automatic library system, human resources information system.

## Properties of a System

A system has the following properties −

### 1.Organization

Organization implies structure and order. It is the arrangement of components that helps to achieve predetermined objectives.

### 2.Interaction

It is defined by the manner in which the components operate with each other.

For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

### 3.Interdependence

Interdependence means how the components of a system depend on one another. For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

### 4.Integration

Integration is concerned with how a system components are connected together. It means that the parts of the system work together within the system even if each part performs a unique function.

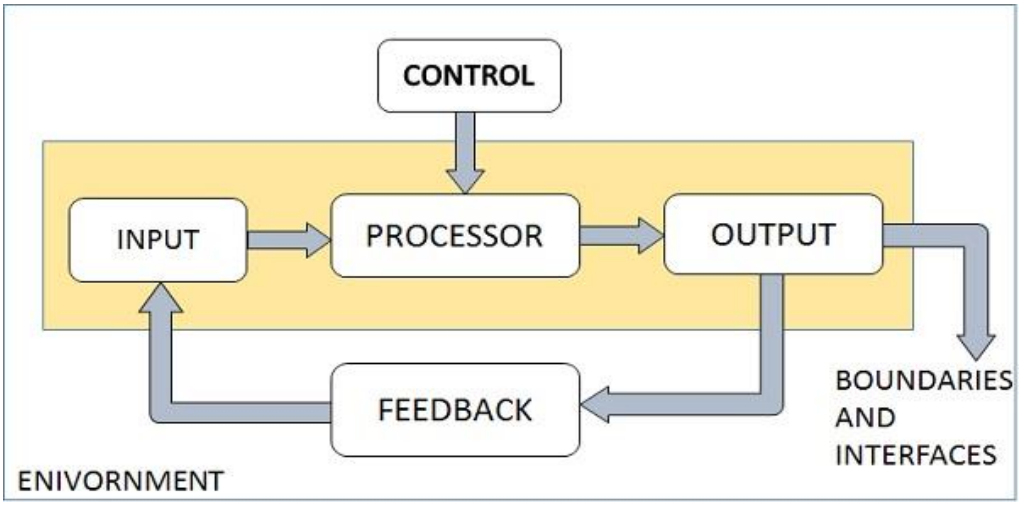
### 5.Central Objective

The objective of system must be central. It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.

The users must know the main objective of a computer application early in the analysis for a successful design and conversion.

## Elements of a System

The following diagram shows the elements of a system −




### Outputs and Inputs

- The main aim of a system is to produce an output which is useful for its user.
- Inputs are the information that enters into the system for processing.
- Output is the outcome of processing.

## Processor(s)

- The processor is the element of a system that involves the actual transformation of input into output.

- It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.

- As the output specifications change, so does the processing. In some cases, input is also modified to enable the processor for handling the transformation.

## Control

- The control element guides the system.

- It is the decision–making subsystem that controls the pattern of activities governing input, processing, and output.

- The behavior of a computer System is controlled by the Operating System and software. In order to keep system in balance, what and how much input is needed is determined by Output Specifications.

## Feedback

- Feedback provides the control in a dynamic system.

- Positive feedback is routine in nature that encourages the performance of the system.

- Negative feedback is informational in nature that provides the controller with information for action.

## Environment

- The environment is the “supersystem” within which an organization operates.

- It is the source of external elements that strike on the system.

- It determines how a system must function. For example, vendors and competitors of organization’s environment, may provide constraints that affect the actual performance of the business.

## Boundaries and Interface

- A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.

- Each system has boundaries that determine its sphere of influence and control.

- The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

# Types of Systems

The systems can be divided into the following types −

## Physical or Abstract Systems

- Physical systems are tangible entities. We can touch and feel them.

- Physical System may be static or dynamic in nature. For example, desks and chairs are the physical parts of computer center which are static. A programmed computer is a dynamic system in which programs, data, and applications can change according to the user's needs.

- Abstract systems are non-physical entities or conceptual that may be formulas, representation or model of a real system.

## Open or Closed Systems

- An open system must interact with its environment. It receives inputs from and delivers outputs to the outside of the system. For example, an information system which must adapt to the changing environmental conditions.

- A closed system does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

## Adaptive and Non Adaptive System

- Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals.

- Non Adaptive System is the system which does not respond to the environment. For example, machines.

## Permanent or Temporary System

- Permanent System persists for long time. For example, business policies.

- Temporary System is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

## Natural and Manufactured System

- Natural systems are created by the nature. For example, Solar system, seasonal system.

- Manufactured System is the man-made system. For example, Rockets, dams, trains.

## Deterministic or Probabilistic System

- Deterministic system operates in a predictable manner and the interaction between system components is known with certainty. For example, two molecules of hydrogen and one molecule of oxygen makes water.

- Probabilistic System shows uncertain behavior. The exact output is not known. For example, Weather forecasting, mail delivery.

## Social, Human-Machine, Machine System

- Social System is made up of people. For example, social clubs, societies.

- In Human-Machine System, both human and machines are involved to perform a particular task. For example, Computer programming.

- Machine System is where human interference is neglected. All the tasks are performed by the machine. For example, an autonomous robot.

## Man–Made Information Systems

- It is an interconnected set of information resources to manage data for particular organization, under Direct Management Control (DMC).

- This system includes hardware, software, communication, data, and application for producing information according to the need of an organization.

Man-made information systems are divided into three types −

- Formal Information System − It is based on the flow of information in the form of memos, instructions, etc., from top level to lower levels of management.

- Informal Information System − This is employee based system which solves the day to day work related problems.

- Computer Based System − This system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

# Systems Models

## Schematic Models

- A schematic model is a 2-D chart that shows system elements and their linkages.

- Different arrows are used to show information flow, material flow, and information feedback.

## Flow System Models

- A flow system model shows the orderly flow of the material, energy, and information that hold the system together.

- Program Evaluation and Review Technique (PERT), for example, is used to abstract a real world system in model form.

## Static System Models

- They represent one pair of relationships such as activity–time or cost–quantity.

- The Gantt chart, for example, gives a static picture of an activity-time relationship.

## Dynamic System Models

- Business organizations are dynamic systems. A dynamic model approximates the type of organization or application that analysts deal with.

- It shows an ongoing, constantly changing status of the system. It consists of −

o Inputs that enter the system

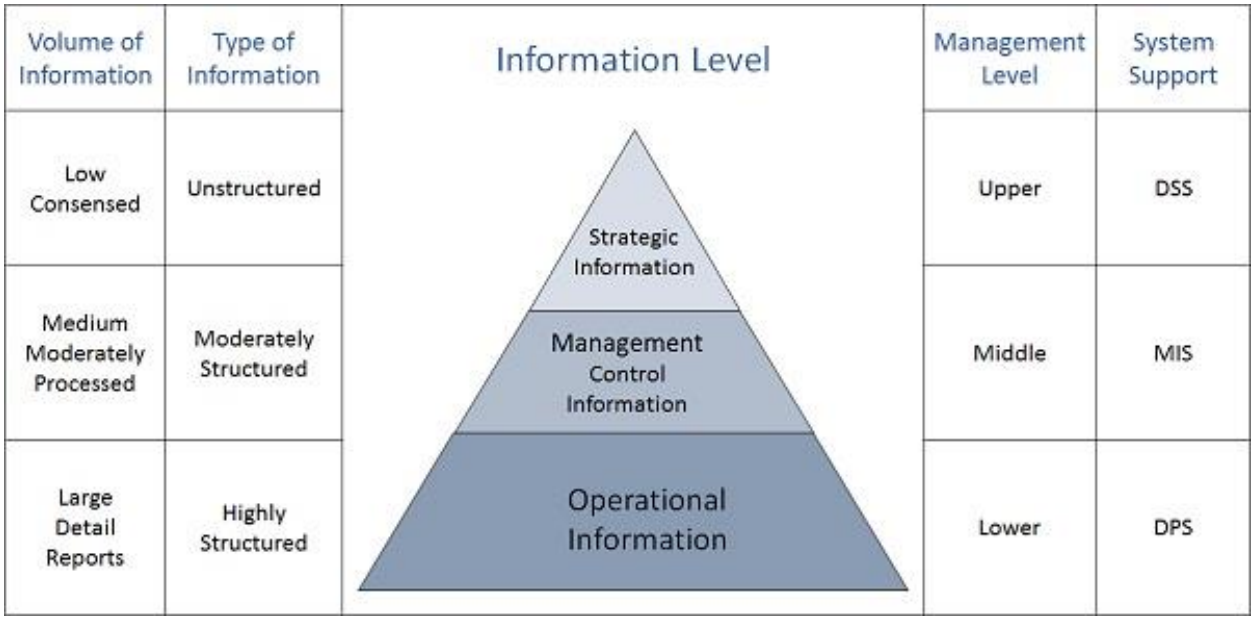
o The processor through which transformation takes place

o The program(s) required for processing

o The output(s) that result from processing.

# Categories of Information

There are three categories of information related to managerial levels and the decision managers make.




## Strategic Information

- This information is required by topmost management for long range planning policies for next few years. For example, trends in revenues, financial investment, and human resources, and population growth.

- This type of information is achieved with the aid of Decision Support System (DSS).

## Managerial Information

- This type of Information is required by middle management for short and intermediate range planning which is in terms of months. For example, sales analysis, cash flow projection, and annual financial statements.

- It is achieved with the aid of Management Information Systems (MIS).

## Operational information

- This type of information is required by low management for daily and short term planning to enforce day-to-day operational activities. For example, keeping employee attendance records, overdue purchase orders, and current stocks available.

- It is achieved with the aid of Data Processing Systems (DPS).

# Business system concept

The system helps the business organisations to achieve their goals.

A business system is a combination of policies, personnel, equipment and computer facilities to co-ordinate the activities of a business organisation.

It establishes the rules and procedures of that organisation, which are to be governed.

Business system decides how data must be handled and is methodically processed. It also controls the procedures of the processed data and the results to be displayed. For e.g. a system may automatically order parts for an inventory, monitor future corporate profits or post credit card sales to the on line customer accounts. The overall nature of the business system will reflect the efficiency of its designers.

## Objectives of Business System:

The objectives of business system are:

1. To meet the user and customer needs.

2. To cut down the operating costs and increase savings.

3. To smooth the flow data through various levels of the organisation.

4. To speed up the execution of results with the reliable data available in a system

5. To handle data efficiently and provide timely information to the management.

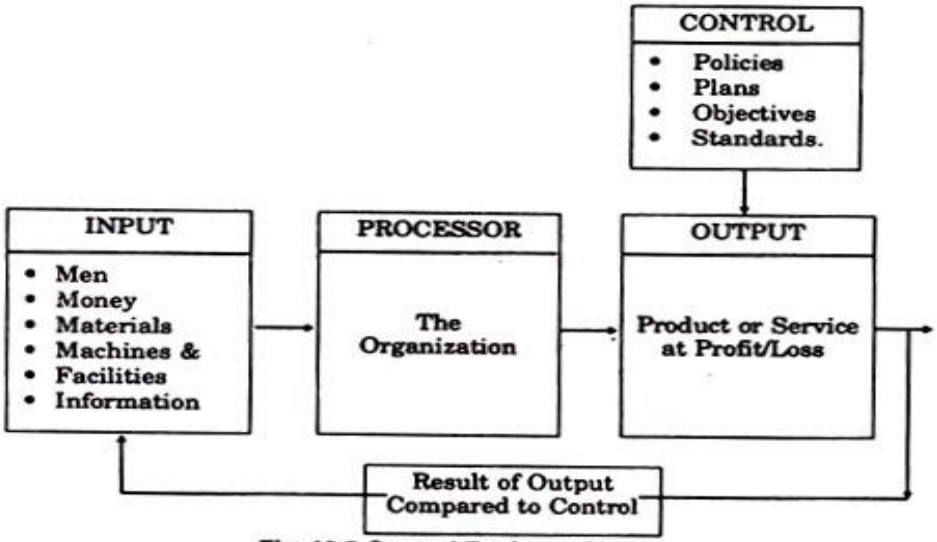



**Fig. 13.5 General Business System.**

6. To establish the most desirable distribution of data, services and equipment's throughout the organisation.

7. To define a proper method of handling business activities.

8. To eliminate duplicated, conflicting and unnecessary services.

## Types of Business Systems:

There are five major types of business systems (Fig. 13.6).

1. Payroll business system

2. Personnel business system

3. Accounts receivable system

4. Accounts payable system

5. Inventory system.

### 1. Payroll Business System:

A payroll system consists of all forms, procedures, files, equipment's, personnel, and computer support necessary to completely process the payment of employees. A payroll system fully handles all tax deductions, personal deductions, and the update of payroll data related to each employee.

It provides for the actual payment of employees, a record of that payment, the modification of all payroll records, and the preparation of payroll reports. The payroll system must also generate all tax documents to include pay-cheques, W-2 statements, 941 quarterly reports, and a wide range of state and municipal employment tax filings.

Another payroll responsibility is the accurate reporting of all personal deductions to include bonds, medical and life insurance, profit sharing plans, stock options, credit union deductions, and the garnishing of an employee's salary by a creditor.

These accumulated totals must be reported accurately to both the recipient of these movies and the individuals from whose salaries these amounts were deducted. The computer's support makes it possible to accurately and promptly process a payroll, providing the input data are properly handled on a timely basis.

**2. Personnel Business System:**

Personnel system describes varied aspects of an organisation's work force. The outputs generated by personnel systems are frequently used in compiling central & state labor power reports. Retail organisations are major users of accounts receivable systems, since these systems detail monies that are owed to an organisation.

Conversely, accounts payable systems focus on the monies that are owed to an organisation. These two systems parallel to each other, requiring the continued maintenance of files, their update reporting on movies due and owed, providing customer statements and invoices, and recording payments made.

**3. Accounts Receivable System:**

An account receivable systems are monitors the flow of money. An accounts receivable system monitors the people who owe money to a business. It provides the means to process all data for credit cards and other kinds of charge accounts.

The files contain the individual customer data, including names, addresses, financial charges like, payments received and current charges. The information is issued as monthly statements of each customer and also provides useful information for management's use.

**4. Accounts Payable System:**

Accounts payable system monitors the organisation to which money is owed. The file structures and input/output (I/O) formats are similar as the accounts receivable system. It contains the accounts of vendors to whom money is owed. Input will have goods and services received by the company while outputs include issue of payments and management reports.

**5. Inventory System:**

Inventory system monitors the status of items held in an inventory. These systems report on the quantities of goods on hand, as well as when items should be purchased to replenish stock and what critical items are needed. Inventory systems are crucial to organisations that maintain large and costly inventories.

# System Development Life Cycle (SDLC)

The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application.

An effective System Development Life Cycle (SDLC) should result in a high quality system that meets customer expectations, reaches completion within time and cost evaluations, and works effectively and efficiently in the current and planned Information Technology infrastructure.

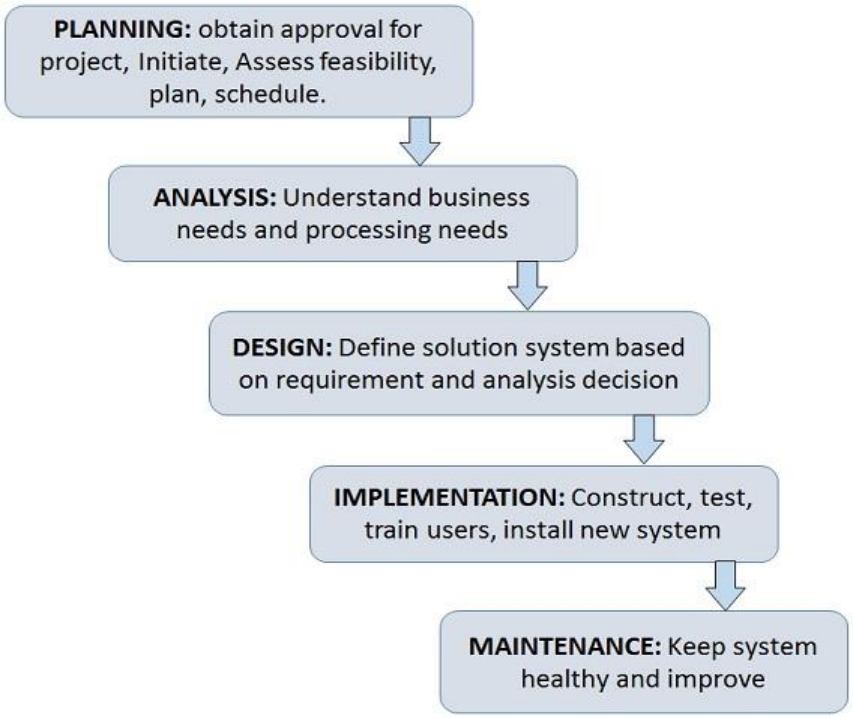
System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles.

SDLC is used by analysts to develop an information system. SDLC includes the following activities –

## SDLC Phases

The entire SDLC process divided into the following stages:

- Phase 1: Requirement collection and analysis




- Phase 2: Feasibility study:
- Phase 3: Design:
- Phase 4: Coding:
- Phase 5: Testing:
- Phase 6: Installation/Deployment:
- Phase 7: Maintenance:

## Phase 1: Requirement collection and analysis:

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

## Phase 2: Feasibility study:

Once the requirement analysis phase is completed the next step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

There are mainly five types of feasibilities checks:

- Economic: Can we complete the project within the budget or not?
- Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
- Operation feasibility: Can we create operations which is expected by the client?
- Technical: Need to check whether the current computer system can support the software
- Schedule: Decide that the project can be completed within the given schedule or not.

## Phase 3: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

**High-Level Design (HLD)**

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements

- Complete architecture diagrams along with technology details

**Low-Level Design(LLD)**

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

## Phase 4: Coding:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

## Phase 5: Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

## Phase 6: Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

## Phase 7: Maintenance:

Once the system is deployed, and customers start using the developed system, following 3 activities occur

- Bug fixing - bugs are reported because of some scenarios which are not tested at all
- Upgrade - Upgrading the application to the newer versions of the Software
- Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

# Popular SDLC models

Here, are some most important phases of SDLC life cycle:

### Waterfall model

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases. In this SDLC model, the outcome of one phase acts as the input for the next phase.

This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

### Incremental Approach

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

### V-Model

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

## Agile Model

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

## Spiral Model

The spiral model is a risk-driven process model. This SDLC model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc.

This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

## Big bang model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements is either unknown or final release date is not given.

Conclusion

- The SDLC is a systematic process for building software that ensures the quality and correctness of the software built
- The full form SDLC is Software Development Lifecycle.
- SDLC process provides a framework for a standard set of activities and deliverables
- Seven different SDLC stages are 1) Requirement collection and analysis 2) Feasibility study: 3) Design 4) Coding 5) Testing: 6) Installation/Deployment and 7) Maintenance
- The senior team members conduct the requirement analysis phase
- Feasibility Study stage includes everything which should be designed and developed during the project life cycle

- In the Design phase, the system and software design documents are prepared as per the requirement specification document

- In the coding phase, developers start build the entire system by writing code using the chosen programming language

- Testing is the next phase which is conducted to verify that the entire application works according to the customer requirement.

- Installation and deployment face begins when the software testing phase is over, and no bugs or errors left in the system

- Bug fixing, upgrade, and engagement actions covered in the maintenance face

- Waterfall, Incremental, Agile, V model, Spiral, Big Bang are some of the popular SDLC models

- SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software

## *Advantages and disadvantages of SDLC*

Benefits of abiding by a clearly defined SDLC model include:

- Having a clear view of an entire project, workers involved, estimated costs and timelines.

- Gives project managers a projected base cost of the project.

- Goals and standards are clearly defined.

- Developers can move back a step if something does not go as expected.

## *Disadvantages, however, can include:*

- Due to assumptions made at the beginning of a project, if an unexpected circumstance complicates the development of a system, then it may stockpile into more complications down the road. As an example, if newly installed hardware does not work correctly, then it may increase the time a system is in development, increasing the cost.

- Some methods are not flexible.

- It can be complicated to estimate the overall cost at the beginning of a project.
- Testing at the end of development may slow down some development teams.

# Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

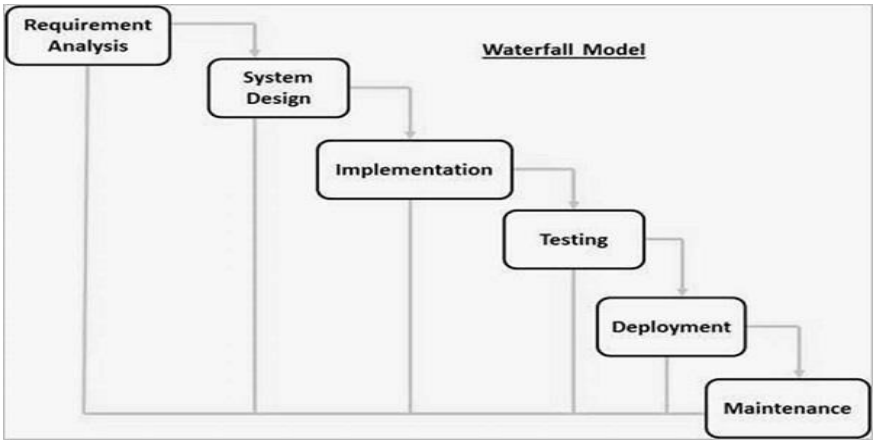
The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap. It was introduced in 1970 by Winston Royce.

## Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.




Let us now learn about each of these phases in brief details:

**1. Requirements analysis and specification:** The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

o Requirement gathering and analysis: Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (inconsistent requirement is one in which some part of the requirement contradicts with some other part).

o Requirement specification: These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

**2. Design**: The aim of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.

**3. Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing. In coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.

**4. Integration and System testing**: Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

System testing consists three different kinds of testing activities as described below :

o Alpha testing: Alpha testing is the system testing performed by the development team.

o Beta testing: Beta testing is the system testing performed by a friendly set of customers.

o Acceptance testing: After the software has been delivered, the customer performed the acceptance testing to determine whether to accept the delivered software or to reject it.

**5. Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**6. Maintainence**: Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is the 60% of the total effort spent to develop a full software. There are basically three types of maintenance :

o Corrective Maintenance: This type of maintenance is carried out to correct errors that were not discovered during the product development phase.

o Perfective Maintenance: This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.

o Adaptive Maintenance: Adaptive maintenance is usually required for porting the software to work in a new environment such as work on a new computer platform or with a new operating system.

## *Advantages of Classical Waterfall Model*

Classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered as the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model:

- This model is very simple and is easy to understand.
- Phases in this model are processed one at a time.
- Each stage in the model is clearly defined.
- This model has very clear and well undestood milestones.
- Process, actions and results are very well documented.
- Reinforces good habits: define-before- design, design-before-code.
- This model works well for smaller projects and projects where requirements are well understood.

## *Drawbacks of Classical Waterfall Model*

Classical waterfall model suffers from various shortcomings, basically we can't use it in real projects, but we use other software development lifecycle models which are based on the classical waterfall model. Below are some major drawbacks of this model:

- No feedback path: In classical waterfall model evolution of a software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phases. Therefore, it does not incorporate any mechanism for error correction.

- Difficult to accommodate change requests: This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customers' requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.

- No overlapping of phases: This model recommends that new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap.

No working software is produced until late during the life cycle.

High amounts of risk and uncertainty.

Not a good model for complex and object-oriented projects.

Poor model for long and ongoing projects.

Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.

It is difficult to measure progress within stages.

Cannot accommodate changing requirements.

Adjusting scope during the life cycle can end a project.

Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

# Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

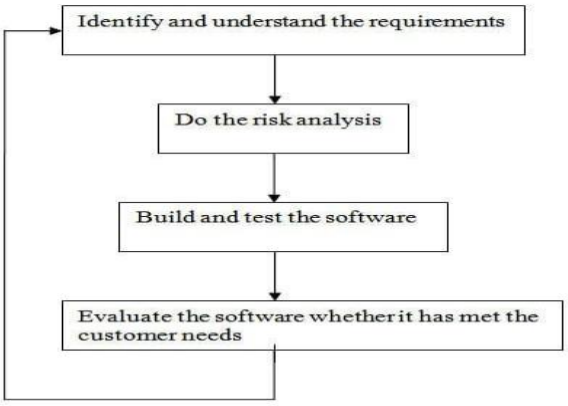
- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

# Spiral Model

Spiral Model is a combination of a waterfall model and iterative model. Each phase in spiral model begins with a design goal and ends with the client reviewing the progress. The spiral model was first mentioned by Barry Boehm in his 1986 paper.

The development team in Spiral-SDLC model starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.

To explain in simpler terms, the steps involved in the spiral model are:




SDLC Spiral Model - © www.SoftwareTestingHelp.com

A spiral model has 4 phases described below:

1. Planning phase

2. Risk analysis phase

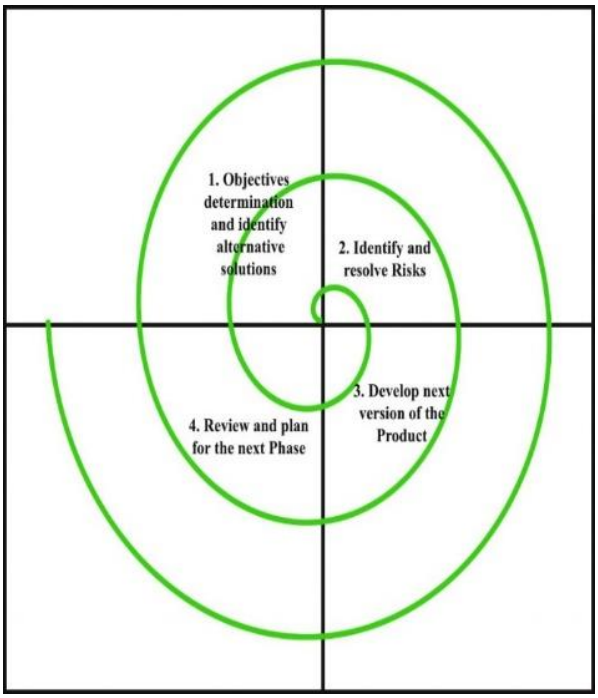
3. Engineering phase

4. Evaluation phase.

Activities which are performed in the spiral model phases are shown below:

| Phase Name | Activities performed | Deliverables / Output |
|---|---|---|
| Planning | -Requirements are studied and gathered.<br>- Feasibility study<br>- Reviews and walkthroughs to streamline the requirements | Requirements understanding document<br><br>Finalized list of requirements. |
| Risk Analysis | Requirements are studied and brain storming sessions are done to identify the potential risks<br><br>Once the risks are identified , risk mitigation strategy is planned and finalized | Document which highlights all the risks and its mitigation plans. |
| Engineering | Actual development and testing if the software takes place in this phase | Code<br>Test cases and test results<br>Test summary report and defect report |
| Evaluation | Customers evaluate the software and provide their feedback and approval | Features implemented document |

# Pictorial representation of SDLC Spiral model




Different colors represent different spiral or iteration. For first iteration, represented in brown color , all the 4 activities (Planning, risk analysis, engineering and evaluation) are performed. After the evaluation phase is over for the first iteration (spiral), second iteration (spiral) starts The second iteration, which is represented in orange color, here again all the 4 activities (Planning, risk analysis, engineering and evaluation) are performed. In a similar way, third iteration is done shown in blue color and so on the process continues.

# When to Use Spiral model?

Spiral model is used in the following scenarios:

- When the project is large.
- Where the software needs continuous risk evaluation.
- Requirements are a bit complicated and require continuous clarification.
- Software requires significant changes.
- Where enough time frame is their to get end user feedback.
- Where releases are required to be frequent.

## *Advantages of using Spiral Model:*

- Development is fast
- Larger projects / software are created and handled in a strategic way
- Risk evaluation is proper.
- Control towards all the phases of development.
- More and more features are added in a systematic way.
- Software is produced early.
- Has room for customer feedback and the changes are implemented faster.

## *Disadvantages of using Spiral model:*

- Risk analysis is important phase so requires expert people.
- Is not beneficial for smaller projects.
- Spiral may go infinitely.
- Documentation is more as it has intermediate phases.
- It is costly for smaller projects.

## **Conclusion:**

Each spiral can be termed as a loop and each loop is a separate development process in a spiral model. The four activities (Planning, Risk analysis, engineering and evaluation) form the intermediary phases of a spiral model and is repeated again for each loop.

This model is very good to use for larger projects where you can develop and deliver smaller prototypes and can enhance it to make the larger software. The implementation of this model requires experienced resources as risk analysis is a very integral part of this model and risk analysis requires expertise and as a result this model becomes costly.

# COCOMO Model

COCOMO is one of the most widely used software estimation models in the world. This model is developed in 1981 by Barry Boehm to give estimation of number of man-months it will take to develop a software product.

COCOMO predicts the efforts and schedule of software product based on size of software.

In COCOMO, projects are categorized into three types:

1. Organic
2. Semidetached
3. Embedded

1**.Organic**: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects. Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.

2**. Semidetached**: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience in related systems but may be unfamiliar with some aspects of the order being developed. Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.

3. **Embedded**: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist. For Example: ATM, Air Traffic control.

For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.

## Types of Models:

COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

1. Basic COCOMO Model

2. Intermediate COCOMO Model

3. Detailed COCOMO Model

The first level, Basic COCOMO can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase wise henceforth producing a more accurate result. These two models are further discussed below.

Estimation of Effort: Calculations –

**Basic Model –**

$$E=a(KLOC)^b$$

The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a and b for the Basic Model for the different categories of system:

| Software Projects | a | b |
|---|---|---|
| Organic | 2.4 | 1.05 |
| Semi Detached | 3.0 | 1.12 |
| Embedded | 3.6 | 1.20 |

The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

**Intermediate Model –**

The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation.

Classification of Cost Drivers and their attributes:

(i) Product attributes –

- Required software reliability extent
- Size of the application database
- The complexity of the product

(ii) Hardware attributes –

- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time

(iii) Personnel attributes –

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

(iv) Project attributes –

- Use of software tools
- Application of software engineering methods

- Required development schedule

| Cost Drivers | Very Low | Low | Nominal | High | Very High | |
|---|---|---|---|---|---|---|
| Product Attributes | | | | | | |
| Required Software Reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | |
| Size of Application Database | | 0.94 | 1.00 | 1.08 | 1.16 | |
| Complexity of The Product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | |
| Hardware Attributes | | | | | | |
| Runtime Performance Constraints | | | | 1.00 | 1.11 | 1.30 |
| Memory Constraints | | | 1.00 | 1.06 | 1.21 | |
| Volatility of the virtual machine environment | | | | | 0.87 | 1.00 | 1.15 | 1.30 |
| Required turnabout time | 0.94 | 1.00 | 1.07 | 1.15 | | |
| Personnel attributes | | | | | | |
| Analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | |
| Software engineer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | |
| Virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 | | |
| Programming language experience | | 1.14 | 1.07 | 1.00 | 0.95 | |
| Project Attributes | | | | | | |
| Application of software engineering methods | | | | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 |
| Use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | |
| Required development schedule | | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 |

The project manager is to rate these 15 different parameters for a particular project on a scale of one to three. Then, depending on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the EAF (Effort Adjustment Factor). The Intermediate COCOMO formula now takes the form:

$$E = (a(KLOC)^b) * EAF$$

The values of a and b in case of the intermediate model are as follows:

| Software Projects | a | b |
|---|---|---|
| Organic | 3.2 | 1.05 |
| Semi Detached | 3.0 | 1.12 |
| Embeddedc | 2.8 | 1.20 |

**Detailed Model –**

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

- Planning and requirements
- System design
- Detailed design
- Module code and test
- Integration and test
- Cost Constructive model

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

# MODULE-2

## Software Requirement Specification

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system. To achieve this we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers creates test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing. Let's see how to test SRS and the important point to keep in mind while testing it.

1**. Correctness** of SRS should be checked. Since the whole testing phase is dependent on SRS, it is very important to check its correctness. There are some standards with which we can compare and verify.

2. **Ambiguity** should be avoided. Sometimes in SRS, some words have more than one meaning and this might confused testers making it difficult to get the exact reference. It is advisable to check for such ambiguous words and make the meaning clear for better understanding.

3**. Requirements** should be complete. When tester writes test cases, what exactly is required from the application, is the first thing which needs to be clear. For e.g. if application needs to send the specific data of some specific size then it should be clearly mentioned in SRS that how much data and what is the size limit to send.

4**. Consistent** requirements.The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another. This sets the standard and should be followed throughout the testing phase.

5**. Verification of expected result**: SRS should not have statements like “Work as expected”, it should be clearly stated that what is expected since different testers would have different thinking aspects and may draw different results from this statement.

6. **Testing environment**: some applications need specific conditions to test and also a particular environment for accurate result. SRS should have clear documentation on what type of environment is needed to set up.

7**. Pre-conditions defined clearly**: one of the most important part of test cases is pre-conditions. If they are not met properly then actual result will always be different expected result. Verify that in SRS, all the pre-conditions are mentioned clearly.

8**. Requirements ID**: these are the base of test case template. Based on requirement Ids, test case ids are written. Also, requirements ids make it easy to categorize modules so just by looking at them, tester will know which module to refer. SRS must have them such as id defines a particular module.

9. **Security and Performance criteria**: security is priority when a software is tested especially when it is built in such a way that it contains some crucial information when leaked can cause harm to business. Tester should check that all the security related requirements are properly defined and are clear to him. Also, when we talk about performance of a software, it plays a very important role in business so all the requirements related to performance must be clear to the tester and he must also know when and how much stress or load testing should be done to test the performance.

10**. Assumption should be avoided**: sometimes when requirement is not cleared to tester, he tends to make some assumptions related to it, which is not a right way to do testing as assumptions could go wrong and hence, test results may vary. It is better to avoid assumptions and ask clients about all the “missing requirements” to have a better understanding of expected results.

11**. Deletion of irrelevant requirements**: there are more than one team who work on SRS so it might be possible that some irrelevant requirements are included in SRS. Based on the understanding of the software, tester can find out which are these requirements and remove them to avoid confusions and reduce work load.

12**. Freeze requirements:** when an ambiguous or incomplete requirement is sent to client to analyze and tester gets a reply, that requirement result will be updated in the next SRS version and client will freeze that requirement. Freezing here means that result will not change again until and unless some major addition or modification is introduced in the software.

# Characteristics of good SRS

Following are the features of a good SRS document:

1. Correctness: User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

2. Completeness: The SRS is complete if, and only if, it includes the following elements:

(1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

(2). Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

Note: It is essential to specify the responses to both valid and invalid values.

(3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

3. Consistency: The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:

(1). The specified characteristics of real-world objects may conflicts. For example,

(a) The format of an output report may be described in one requirement as tabular but in another as textual.

(b) One condition may state that all lights shall be green while another states that all lights shall be blue.

(2). There may be a reasonable or temporal conflict between the two specified actions. For example,

(a) One requirement may determine that the program will add two inputs, and another may determine that the program will multiply them.

(b) One condition may state that "A" must always follow "B," while other requires that "A and B" co-occurs.

(3). Two or more requirements may define the same real-world object but use different terms for that object. For example, a program's request for user input may be called a "prompt" in

one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.

4. Unambiguousness: SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

5. Ranking for importance and stability: The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.

Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

6. Modifiability: SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

7. Verifiability: SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

8. Traceability: The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

There are two types of Traceability:

1. Backward Traceability: This depends upon each requirement explicitly referencing its source in earlier documents.

2. Forward Traceability: This depends upon each element in the SRS having a unique name or reference number.

The forward traceability of the SRS is especially crucial when the software product enters the operation and maintenance phase. As code and design document is modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

9. Design Independence: There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

10. Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

11. Understandable by the customer: An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

12. The right level of abstraction: If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas,for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

## Properties of a good SRS document

The essential properties of a good SRS document are the following:

Concise: The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

Structured: It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.

Black-box view: It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.

Conceptual integrity: It should show conceptual integrity so that the reader can merely understand it. Response to undesired events: It should characterize acceptable responses to unwanted events. These are called system response to exceptional conditions.

Verifiable: All requirements of the system, as documented in the SRS document, should be correct. This means that it **should** be possible to decide whether or not requirements have been met in an implementation.

## What is Structured Analysis?

Structured Analysis is a development method that allows the analyst to understand the system and its activities in a logical way.

It is a systematic approach, which uses graphical tools that analyze and refine the objectives of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes −

- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is logical rather than physical i.e., the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-level overviews to lower-level details.

## Structured Analysis Tools

During Structured Analysis, various tools and techniques are used for system development. They are −

- Data Flow Diagrams
- Data Dictionary
- Decision Trees
- Decision Tables
- Structured English
- Pseudocode

## Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

## Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system.For example in a Banking software system, how data is moved between different entities.

- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

## DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

- Entities - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.

- Process - Activities and action taken on the data are represented by Circle or Round-edged rectangles.

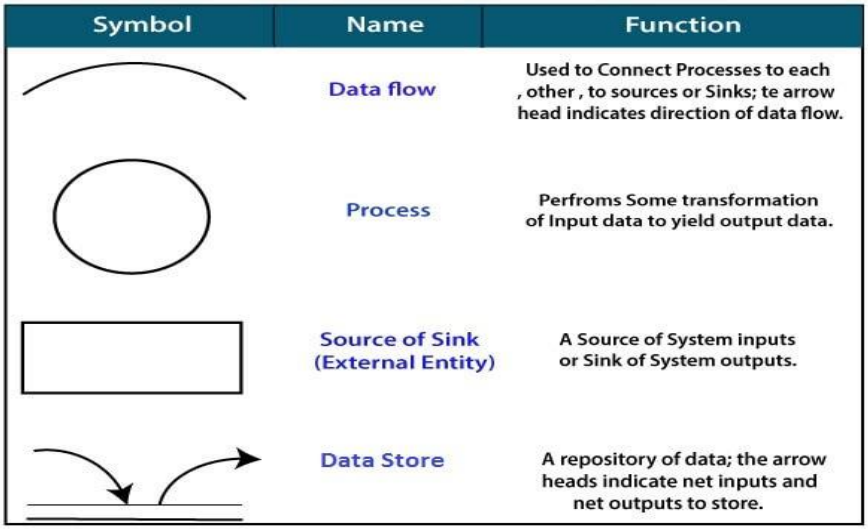
- Data Storage - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.

- Data Flow - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

The following observations about DFDs are essential:

- All names should be unique. This makes it easier to refer to elements in the DFD.
- Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
- Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
- Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:




**Symbols for Data Flow Diagrams**

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

Data Store: A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

Source or Sink: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.

# Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

## 0-level DFDM

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x1 and x2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

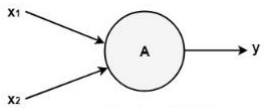



Fig: Level-0 DFD.

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.

## 1-level DFD

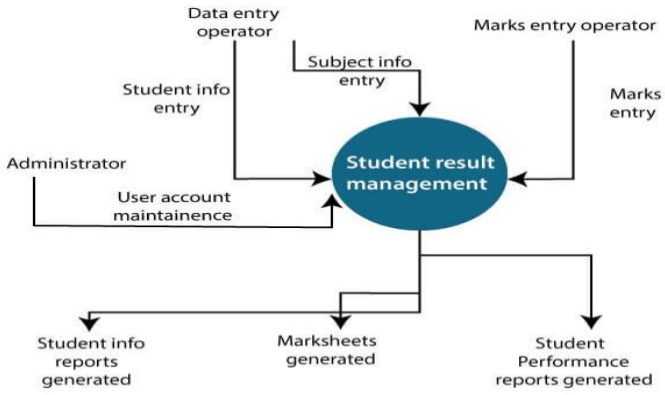



Fig: Level-0 DFD of result management system

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

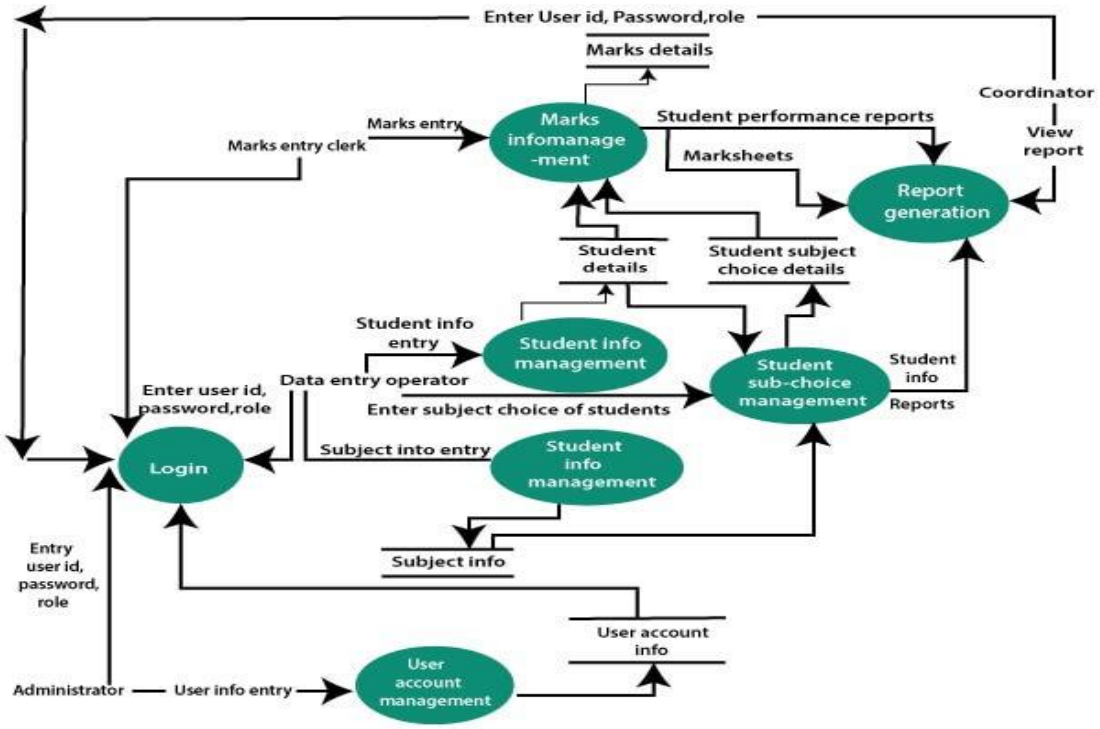
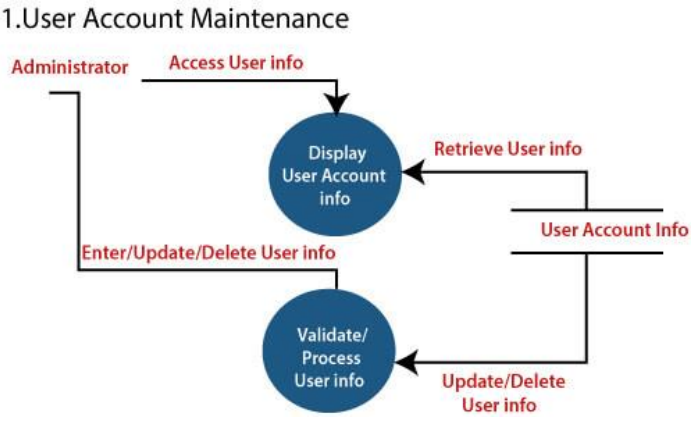
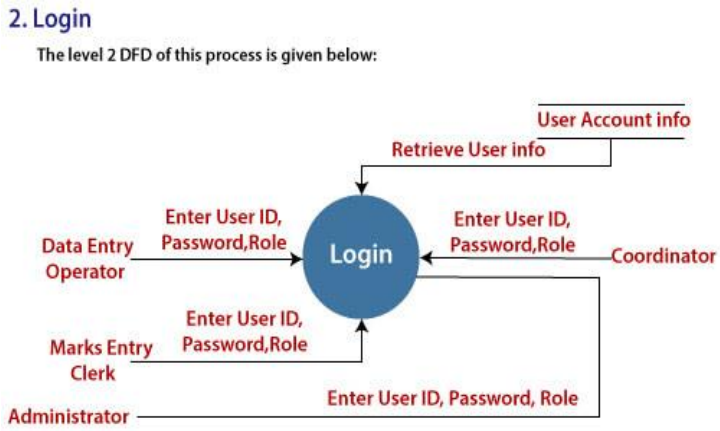



**Fig: Level-1 DFD of result management system**

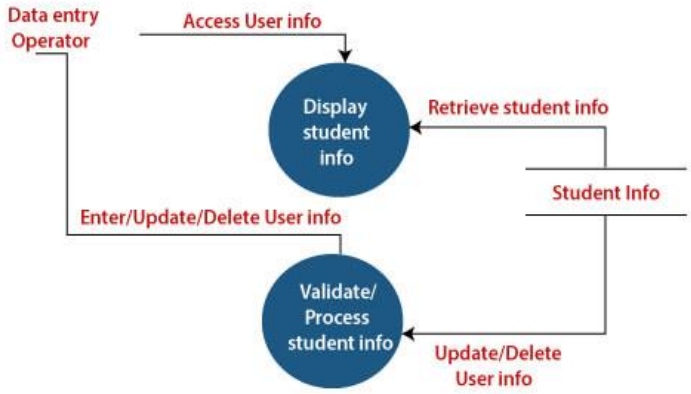
## 2-Level DFD

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.



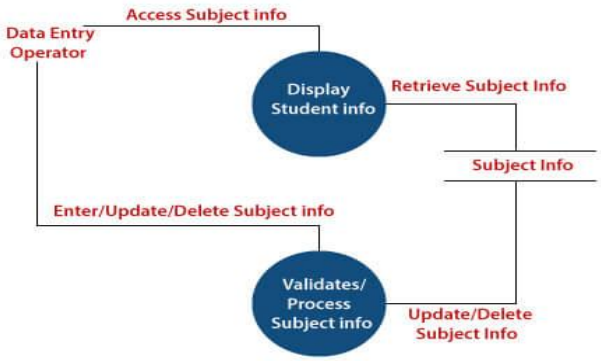



### 3. Student Information Management



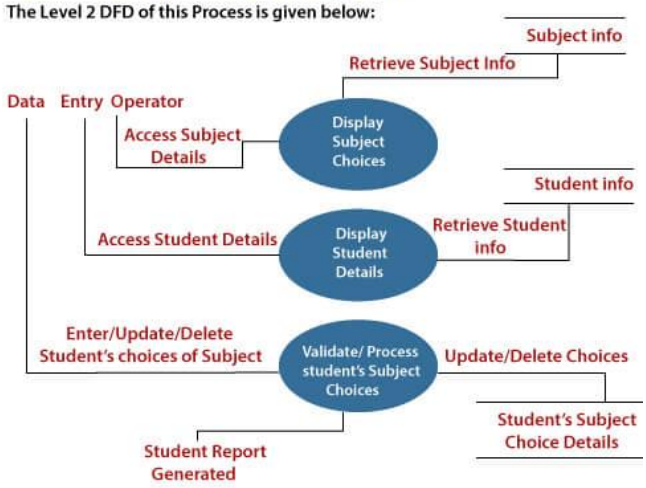

### 4. Subject Information Management

The level 2 DFD of this process is given below:



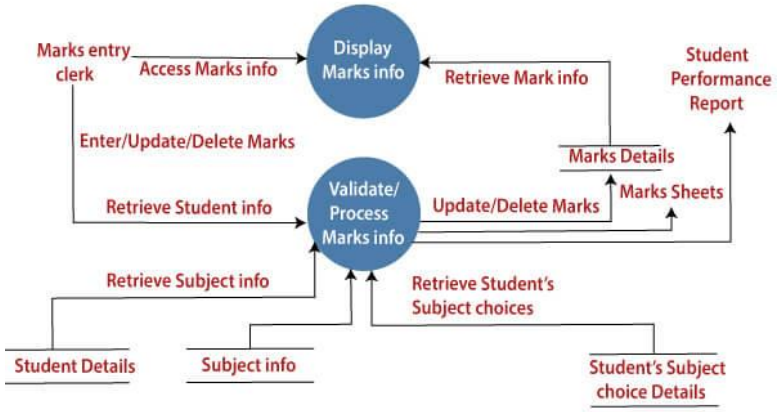

### 5. Student's Subject Choice Management

The Level 2 DFD of this Process is given below:




### 6. Marks Information Managment

The Level 2 DFD of this Process is given below:




# Data Dictionary

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary hold records about other objects in the database, such as data ownership, data relationships to other objects, and other data. The data dictionary is an essential component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

The data dictionary, in general, includes information about the following:

- Name of the data item
- Aliases
- Description/purpose

- Related data items
- Range of values
- Data structure definition/Forms

The **name of the data item** is self-explanatory.

**Aliases** include other names by which this data item is called DEO for Data Entry Operator and DR for Deputy Registrar.

**Description/purpose** is a textual description of what the data item is used for or why it exists.

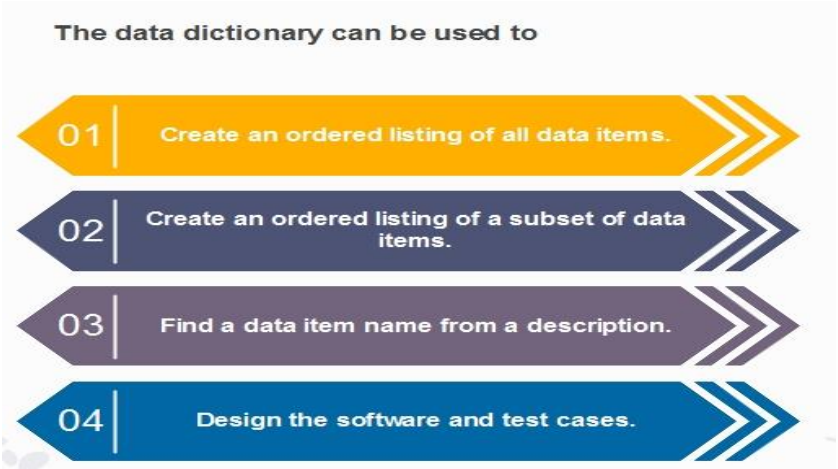
**Related data items** capture relationships between data items e.g., total_marks must always equal to internal_marks plus external_marks.

**Range of values** records all possible values, e.g. total marks must be positive and between 0 to 100.

**Data structure Forms:** Data flows capture the name of processes that generate or receive the data items. If the data item is primitive, then data structure form captures the physical structures of the data item. If the data is itself a data aggregate, then data structure form capture the composition of the data items in terms of other data items.

The mathematical operators used within the data dictionary are defined in the table:

| Notations | Meaning |
|---|---|
| x=a+b | x includes of data elements a and b. |
| x=[a/b] | x includes of either data elements a or b. |
| x=a | x includes of optimal data elements a. |
| x=y[a] | x includes of y or more occurrences of data element a |
| x=[a]z | x includes of z or fewer occurrences of data element a |
| x=y[a]z | x includes of some occurrences of data element a which are between y and z. |




## The different types of data dictionary are −

### Active Data Dictionary

If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.

So, the data dictionary is automatically updated by the database management system when any changes are made in the database. This is known as an active data dictionary as it is self updating.

### Passive Data Dictionary

This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary. That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

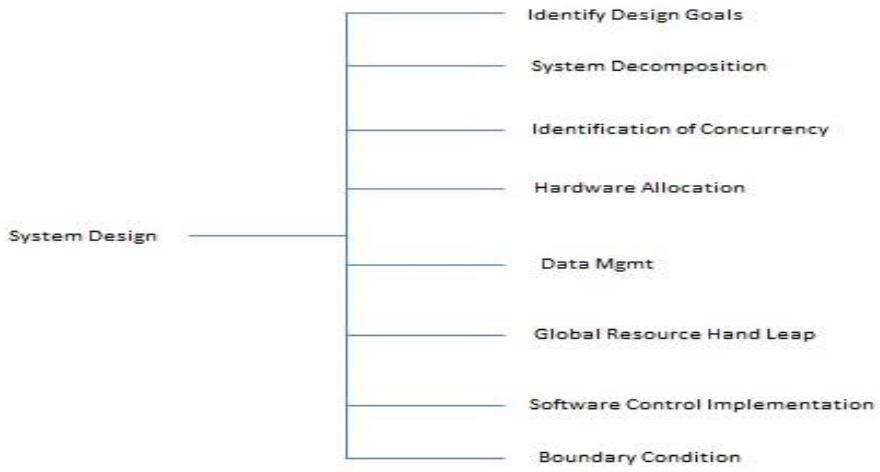
So, the passive data dictionary has to be manually updated to match the database. This needs careful handling or else the database and data dictionary are out of sync

# System Design

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. “how to implement?”

It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

In this phase, the complex activity of system development is divided into several smaller sub-activities, which coordinate with each other to achieve the main objective of system development.




## Inputs to System Design

System design takes the following inputs −

- Statement of work
- Requirement determination plan
- Current situation analysis
- Proposed system requirements including a conceptual data model, modified DFDs, and Metadata (data about data).

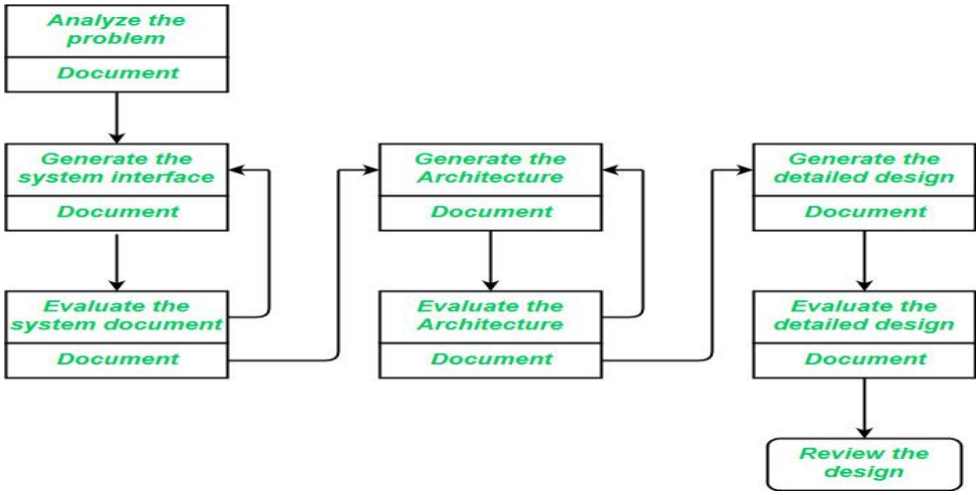
## Outputs for System Design

System design gives the following outputs −

- Infrastructure and organizational changes for the proposed system.
- A data schema, often a relational schema.
- Metadata to define the tables/files and columns/data-items.

- A function hierarchy diagram or web page map that graphically describes the program structure.
- Actual or pseudocode for each module in the program.
- A prototype for the proposed system.

# Types of System Design




## Logical Design

Logical design pertains to an abstract representation of the data flow, inputs, and outputs of the system. It describes the inputs (sources), outputs (destinations), databases (data stores), procedures (data flows) all in a format that meets the user requirements.

While preparing the logical design of a system, the system analyst specifies the user needs at level of detail that virtually determines the information flow into and out of the system and the required data sources. Data flow diagram, E-R diagram modeling are used.

## Physical Design

Physical design relates to the actual input and output processes of the system. It focuses on how data is entered into a system, verified, processed, and displayed as output.

It produces the working system by defining the design specification that specifies exactly what the candidate system does. It is concerned with user interface design, process design, and data design.

It consists of the following steps −

- Specifying the input/output media, designing the database, and specifying backup procedures.

- Planning system implementation.

- Devising a test and implementation plan, and specifying any new hardware and software.

- Updating costs, benefits, conversion dates, and system constraints.

## Architectural Design

It is also known as high level design that focuses on the design of system architecture. It describes the structure and behavior of the system. It defines the structure and relationship between various modules of system development process.

## Detailed Design

It follows Architectural design and focuses on development of each module.

# Elements of a System

- Architecture - This is the conceptual model that defines the structure, behavior and more views of a system. We can use flowcharts to represent and illustrate the architecture.

- Modules - This are components that handle one specific tasks in a system. A combination of the modules make up the system.

- Components - This provides a particular function or group of related functions. They are made up of modules.

- Interfaces - This is the shared boundary across which the components of a the system exchange information and relate.

- Data - This the management of the information and data flow.

## Major Tasks Performed During the System Design Process

## 1. Initialize design definition

- Plan for and Identify the technologies that will compose and implement the systems elements and their physical interfaces.

- Determine which technologies and system elements have a risk to become obsolete, or evolve during the operation stage of the system. Plan for their potential replacement.

- Document the design definition strategy, including the need for and requirements of any enabling systems, products, or services to perform the design.

## 2. Establish design characteristics

- Define the design characteristics relating to the architectural characteristics and check that they are implementable.

- Define the interfaces that were not defined by the System Architecture process or that need to be refined as the design details evolve.

- Define and document the design characteristics of each system element2.

## 3. Assess alternatives for obtaining system elements

- Assess the design options

- Select the most appropriate alternatives.

- If the decision is made to develop the system element, rest of the design definition process and the implementation process are used. If the decision is to buy or reuse a system element, the acquisition process may be used to obtain the system element.

## 4. Manage the design

- Capture and maintain the rationale for all selections among alternatives and decisions for the design, architecture characteristics.

- Assess and control the evolution of the design characteristics.

## Factors that Affect Technology Trade-offs during System Design

Scale of Product

- For example, enterprise software companies that are building system-level software prioritize reliability because customers need to use them. Each change needs to be rigorously tested, and often approved before it can be released.

- Meanwhile, consumer internet companies spend time and money on making their UX delightful so that people want to use them. Reliability is something they're willing to sacrifice.

Since many are web-based applications, they can iterate quickly and release changes frequently.

Time

- Learning new technologies sometimes often takes time. The trade-offs in this instance will be made according to which stack/technology will be in time with the set delivery dates. If switching to a new stack/technology will result in a major shift on the delivery dates and major inconveniences to the stakeholders then the switch can be held off until an appropriate time.

Cost

- On a larger scale Technology decisions are made based on which is more cost effective, where a comparison can be done on which will be more effective between buying an off the shelf system and customizing it or building a new system.

Efficiency

- Technology trade offs are also done based on which technology is more efficient for example choosing between ReactJs or AngularJs for a front end application.

User Experience and Support

- The amount of support and documentation available on a given technology can also be a determining factor on the decisions. Working with Technologies that have a large support base, comprehensive documentation and A good user experience is much easier and take a very short time to ramp up on due to the large amount of resources available to support it.

Maintainability

- maintainability in this case is the ease with which a product can be maintained in order to correct errors, fix bugs and add additional features. Trade-offs decisions will be made based on the maintainability of the Technology

Reliability

- In this case the trade offs are made based on the Technology that performs consistently well and consistently upgrading to more efficient versions.

Scalability

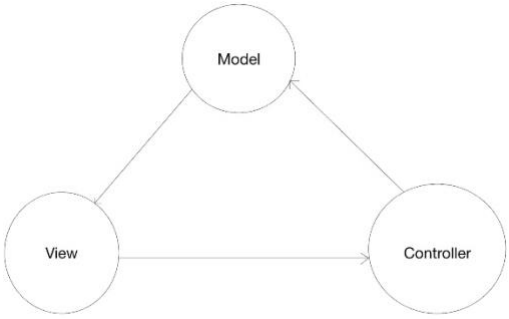
- Technology trade offs are also made based on the technologies that are more scalable and able to handle increase loads efficiently without a break in the system efficiency.

# MVC Design pattern

The Model View Controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information.

MVC mostly relates to the user Interface/interaction layer of an application.

In the MVC pattern the user sees the View which is updated by the model which is turn manipulated by the Controller.




## MVC Pattern

- The Model contains only the pure application data, it contains no logic describing how to present the data to a user. They are the parts of the application that implement the logic for the application's data domain. They retrieve and store model state in a database.

- The View presents the model's data to the user. The view can only be used to access the model's data. They are the components that display the application's user interface (UI).

- The Controller exists between the view and the model. It listens to events triggered by the view and executes the appropriate commands. They are the components that handle user interaction, work with the model, and ultimately select a view to render that displays UI.

## *Advantages of the MVC design pattern*

- Multiple developers can work simultaneously on the model, controller and views.

- MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.

- Low coupling — The very nature of the MVC framework is such that there is low coupling among models, views or controllers.

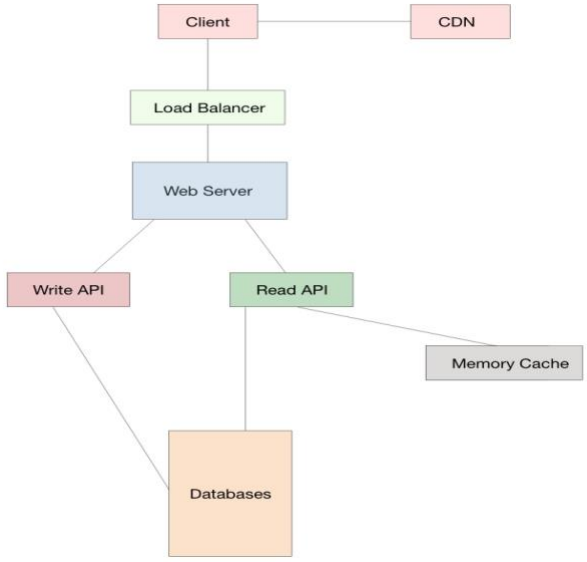
- Models can have multiple views.

- Ease of modification — Because of the separation of responsibilities, future development or modification is easier

## *Disadvantages*

- Knowledge on multiple technologies becomes the norm. Developers using MVC need to be skilled in multiple technologies.

Below is an example of a System Design




# Problem Partitioning

- When solving a small problem, the entire problem can be tackled at once. The complexity of large problems and the limitations of human minds do not allow large problems to be treated as huge monoliths.
- As basic aim of problem analysis is to obtain a clear understanding of the needs of the clients and the users.
- Frequently the client and the users do not understand or know all their needs, because the potential of the new system is often not fully appreciated.
- The analysts have to ensure that the real needs of the clients and the users are uncovered, even if they don't know them clearly.

- That is, the analysts are not just collecting and organizing information about the client's organization and its processes, but they also act as consultants who play an active role of helping the clients and users identify their needs.
- For solving larger problems, the basic principle is the time-tested principle of "divide and conquer.“

**"divide into smaller pieces, so that each piece can be conquered separately.“**

For software design, partition the problem into sub problems and then try to understand each sub problem and its relationship to other sub problems in an effort to understand the total problem.

That is goal is to divide the problem into manageably small pieces that can be solved separately, because the cost of solving the entire problem is more than the sum of the cost of solving all the pieces.

The different pieces cannot be entirely independent of each other, as they together form the system. The different pieces have to cooperate and communicate to solve the larger problem.

Problem partitioning also aids design verification.

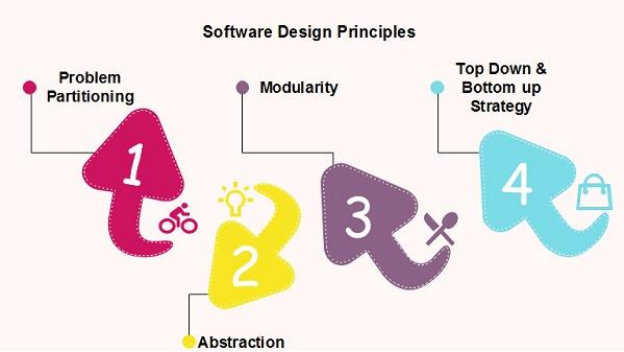
The concepts of state and projection can sometimes also be used effectively in the partitioning process.

A state of a system represents some conditions about the system. This approach is sometimes used in real-time software or process-control software.

Projection, different viewpoints of the system are defined and the system is then analyzed from these different perspectives. The different "projections" obtained are combined to form the analysis for the complete system. Analyzing the system from the different perspectives is often easier, as it limits and focuses the scope of the study.

Software design principles are concerned with providing means to handle the complexity of the design process effectively. Effectively managing the complexity will not only reduce the effort needed for design but can also reduce the scope of introducing errors during design.

Following are the principles of Software Design




## Problem Partitioning

For small problem, we can handle the entire problem at once but for the significant problem, divide the problems and conquer the problem it means to divide the problem into smaller pieces so that each piece can be captured separately.

For software design, the goal is to divide the problem into manageable pieces.

Benefits of Problem Partitioning

- Software is easy to understand
- Software becomes simple
- Software is easy to test
- Software is easy to modify
- Software is easy to maintain
- Software is easy to expand

These pieces cannot be entirely independent of each other as they together form the system. They have to cooperate and communicate to solve the problem. This communication adds complexity.

Note: As the number of partition increases = Cost of partition and complexity increases

## Abstraction

An abstraction is a tool that enables a designer to consider a component at an abstract level without bothering about the internal details of the implementation. Abstraction can be used for existing element as well as the component being designed.

Here, there are two common abstraction mechanisms

- Functional Abstraction
- Data Abstraction

Functional Abstraction

- A module is specified by the method it performs.
- The details of the algorithm to accomplish the functions are not visible to the user of the function.

Functional abstraction forms the basis for Function oriented design approaches.

Data Abstraction

Details of the data elements are not visible to the users of data. Data Abstraction forms the basis for Object Oriented design approaches.

## Modularity

Modularity specifies to the division of software into separate modules which are differently named and addressed and are integrated later on in to obtain the completely functional software. It is the only property that allows a program to be intellectually manageable. Single large programs are difficult to understand and read due to a large number of reference variables, control paths, global variables, etc.

The desirable properties of a modular system are:

- Each module is a well-defined system that can be used with other applications.
- Each module has single specified objectives.
- Modules can be separately compiled and saved in the library.
- Modules should be easier to use than to build.
- Modules are simpler from outside than inside.

## *Advantages of Modularity*

There are several advantages of Modularity

- It allows large programs to be written by several or different people

- It encourages the creation of commonly used routines to be placed in the library and used by other programs.
- It simplifies the overlay procedure of loading a large program into main storage.
- It provides more checkpoints to measure progress.
- It provides a framework for complete testing, more accessible to test
- It produced the well designed and more readable program.

## *Disadvantages of Modularity*

- Execution time maybe, but not certainly, longer
- Storage size perhaps, but is not certainly, increased
- Compilation and loading time may be longer
- Inter-module communication problems may be increased
- More linkage required, run-time may be longer, more source lines must be written, and more documentation has to be done

## Modular Design

Modular design reduces the design complexity and results in easier and faster implementation by allowing parallel development of various parts of a system. We discuss a different section of modular design in detail in this section:

1. Functional Independence: Functional independence is achieved by developing functions that perform only one kind of task and do not excessively interact with other modules. Independence is important because it makes implementation more accessible and faster. The independent modules are easier to maintain, test, and reduce error propagation and can be reused in other programs as well. Thus, functional independence is a good design feature which ensures software quality.

It is measured using two criteria:

- Cohesion: It measures the relative function strength of a module.
- Coupling: It measures the relative interdependence among modules.

2. Information hiding: The fundamental of Information hiding suggests that modules can be characterized by the design decisions that protect from the others, i.e., In other words, modules should be specified that data include within a module is inaccessible to other modules that do not need for such information.

The use of information hiding as design criteria for modular system provides the most significant benefits when modifications are required during testing's and later during software maintenance. This is because as most data and procedures are hidden from other parts of the software, inadvertent errors introduced during modifications are less likely to propagate to different locations within the software.

# Strategy of Design

A good system design strategy is to organize the program modules in such a method that are easy to develop and latter too, change. Structured design methods help developers to deal with the size and complexity of programs. Analysts generate instructions for the developers about how code should be composed and how pieces of code should fit together to form a program.

Importance :

- If any pre-existing code needs to be understood, organised and pieced together.
- It is common for the project team to have to write some code and produce original programs that support the application logic of the system.

To design a system, there are two possible approaches:

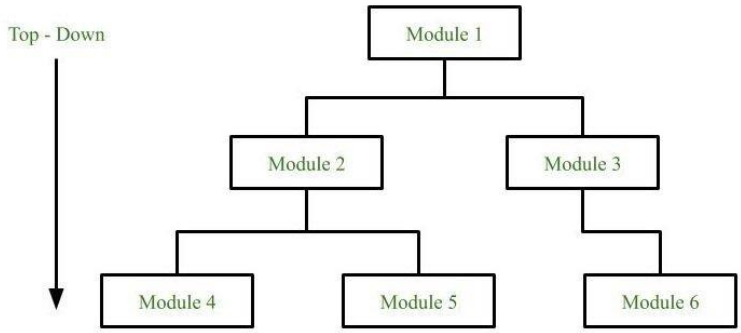
- Top-down Approach
- Bottom-up Approach

***Top-down Approach***: This approach starts with the identification of the main components and then decomposing them into their more detailed sub-components.

Each system is divided into several subsystems and components. Each of the subsystem is further divided into set of subsystems and components. This process of division facilitates in forming a system hierarchy structure. The complete software system is considered as a single entity and in relation to the characteristics, the system is split into sub-system and component. The same is done with each of the sub-system.

This process is continued until the lowest level of the system is reached. The design is started initially by defining the system as a whole and then keeps on adding definitions of the subsystems and components. When all the definitions are combined together, it turns out to be a complete system.

For the solutions of the software need to be developed from the ground level, top-down design best suits the purpose.




## *Advantages:*

- The main advantage of top down approach is that its strong focus on requirements helps to make a design responsive according to its requirements.

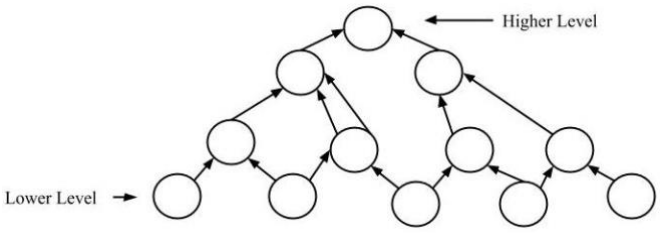
## *Disadvantages:*

- Project and system boundries tends to be application specification oriented. Thus it is more likely that advantages of component reuse will be missed.

- The system is likely to miss, the benefits of a well-structured, simple architecture.

***Bottom-up Approach***:The design starts with the lowest level components and subsystems. By using these components, the next immediate higher level components and subsystems are created or composed. The process is continued till all the components and subsystems are composed into a single component, which is considered as the complete system. The amount of abstraction grows high as the design moves to more high levels.

By using the basic information existing system, when a new system needs to be created, the bottom up strategy suits the purpose.




## *Advantages:*

- The economics can result when general solutions can be reused.
- It can be used to hide the low-level details of implementation and be merged with top-down technique.

## *Disadvantages:*

- It is not so closely related to the structure of the problem.
- High quality bottom-up solutions are very hard to construct.
- It leads to proliferation of 'potentially useful' functions rather than most approprite .

# ER Diagram

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

## Purpose of ERD

o The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.

o The ERD serves as a documentation tool.

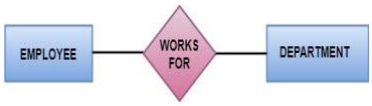
o Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

## Components of an ER Diagrams

### 1. Entity

An entity can be a real-world object, either animate or inanimate, that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

Entity Set

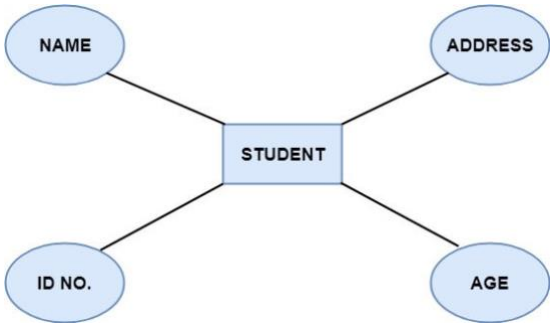



An entity set is a collection of related types of entities. An entity set may include entities with attribute sharing similar values. For example, a Student set may contain all the students of a school; likewise, a Teacher set may include all the teachers of a school from all faculties. Entity set need not be disjoint.

2. **Attributes**

Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



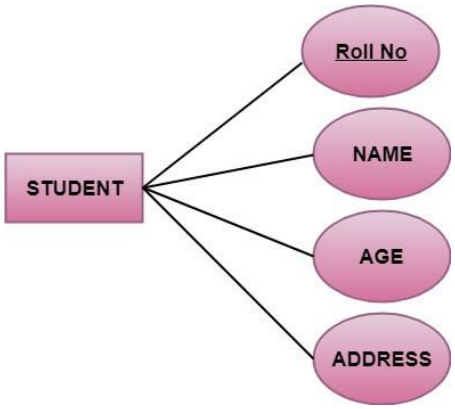

There are four types of Attributes:

1. Key attribute
2. Composite attribute
3. Single-valued attribute

4. Multi-valued attribute

5. Derived attribute

1. **Key attribute**: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll_number of a student makes him identifiable among students.



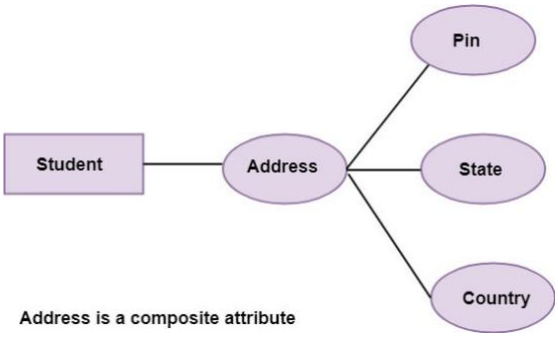

There are mainly three types of keys:

1. Super key: A set of attributes that collectively identifies an entity in the entity set.

2. Candidate key: A minimal super key is known as a candidate key. An entity set may have more than one candidate key.

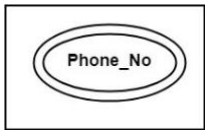
3. Primary key: A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

2. **Composite attribute**: An attribute that is a combination of other attributes is called a composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

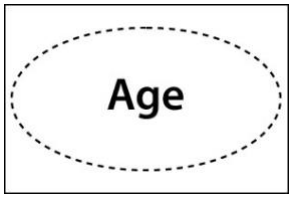


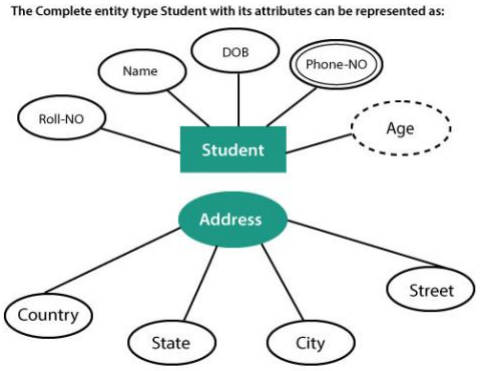

3. **Single-valued attribute**: Single-valued attribute contain a single value. For example, Social_Security_Number.

4. **<u>Multi-valued Attribute</u>**: If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.




5**<u>. Derived attribute</u>**: Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date_of_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.




3**. Relationships**

The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.


Fig: Relationships in ERD

Relationship set

A set of relationships of a similar type is known as a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

<u>Degree of a relationship set</u>

The number of participating entities in a relationship describes the degree of the relationship. The three most common relationships in E-R models are:

1. Unary (degree1)

2. Binary (degree2)

3. Ternary (degree3)

1. Unary relationship: This is also called recursive relationships. It is a relationship between the instances of one entity type. For example, one person is married to only one person.


Fig: Unary Relationship

2. Binary relationship: It is a relationship between the instances of two entity types. For example, the Teacher teaches the subject.


Fig: Binary Relationship

3. Ternary relationship: It is a relationship amongst instances of three entity types. In fig, the relationships "may have" provide the association of three entities, i.e., TEACHER, STUDENT, and SUBJECT. All three entities are many-to-many participants. There may be one or many participants in a ternary relationship.

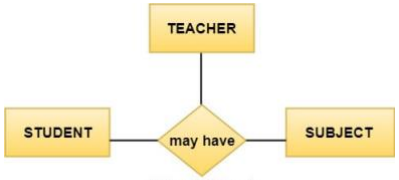



Fig: Ternary Relationship

In general, "n" entities can be related by the same relationship and is known as n-ary relationship.

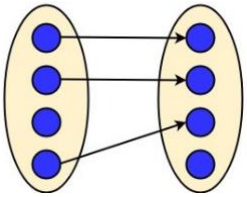
**Cardinality**

Cardinality describes the number of entities in one entity set, which can be associated with the number of entities of other sets via relationship set.

Types of Cardinalities

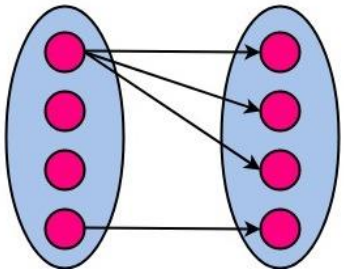
1. One to One: One entity from entity set A can be contained with at most one entity of entity set B and vice versa. Let us assume that each student has only one student ID, and each student ID is assigned to only one person. So, the relationship will be one to one.


**Using Sets, it can be represented as:**



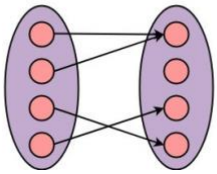
2. One to many: When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationships. For example, a client can place many orders; a order cannot be placed by many customers.

**Using Sets, it can be represented as:**

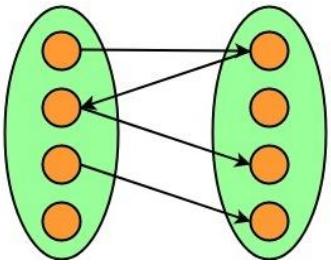
**3.** Many to One**:** More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A. For example - many students can study in a single college, but a student cannot study in many colleges at the same time.


**Using Sets, it can be represented as:**



4. Many to Many: One entity from A can be associated with more than one entity from B and vice-versa. For example, the student can be assigned to many projects, and a project can be assigned to many students.

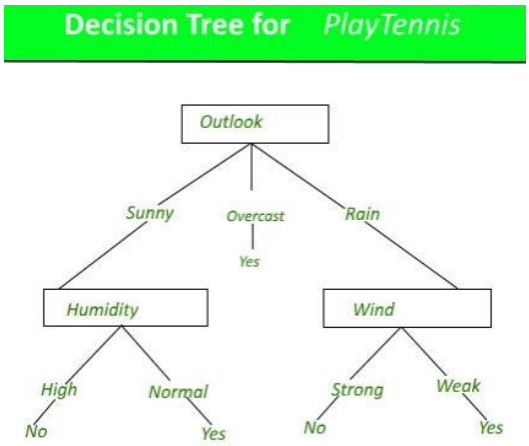

**Using Sets, it can be represented as:**



# Decision tree

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision. Decision trees can be drawn by hand or created with a graphics program or specialized software.

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.




A decision tree for the concept PlayTennis.

# Construction of Decision Tree :

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

## Decision Tree Representation :

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree,testing the attribute specified by this node,then moving down the tree branch corresponding to the value of the attribute as shown in the above figure.This process is then repeated for the subtree rooted at the new node.

The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returning the classification associated with the particular leaf.(in this case Yes or No).

For example,the instance

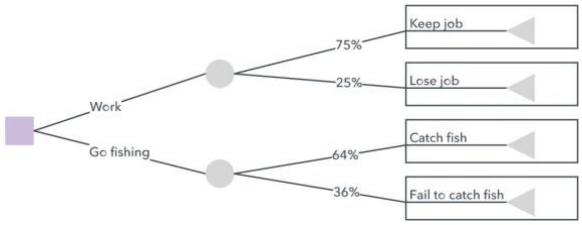
(Outlook = Rain, Temperature = Hot, Humidity = High, Wind = Strong )

would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance.

In other words we can say that decision tree represent a disjunction of conjunctions of constraints on the attribute values of instances.

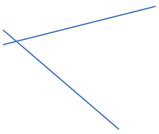
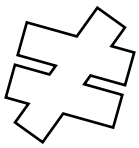
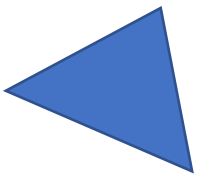
(Outlook = Sunny ^ Humidity = Normal) v (Outllok = Overcast) v (Outlook = Rain ^ Wind = Weak)

There are three different types of nodes: chance nodes, decision nodes, and end nodes. A chance node, represented by a circle, shows the probabilities of certain results. A decision node, represented by a square, shows a decision to be made, and an end node shows the final outcome of a decision path.




Decision trees can also be drawn with flowchart symbols.

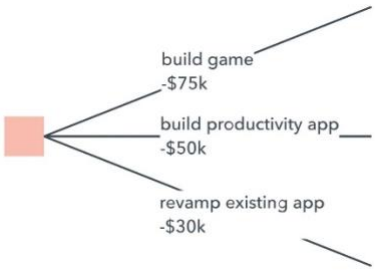
## Decision tree symbols

| Shape | Name | Meaning |
|---|---|---|
|  | Decision node | Indicates a decision to be made |
|  | Chance node | Shows multiple uncertain outcomes |
|  | Alternative branches | Each branch indicates a possible outcome or action |
|  | Rejected alternative | Shows a choice that was not selected |
|  | Endpoint node | Indicates a final outcome |

## How to draw a decision tree

To draw a decision tree, first pick a medium. You can draw it by hand on paper or a whiteboard, or you can use special decision tree software. In either case, here are the steps to follow:

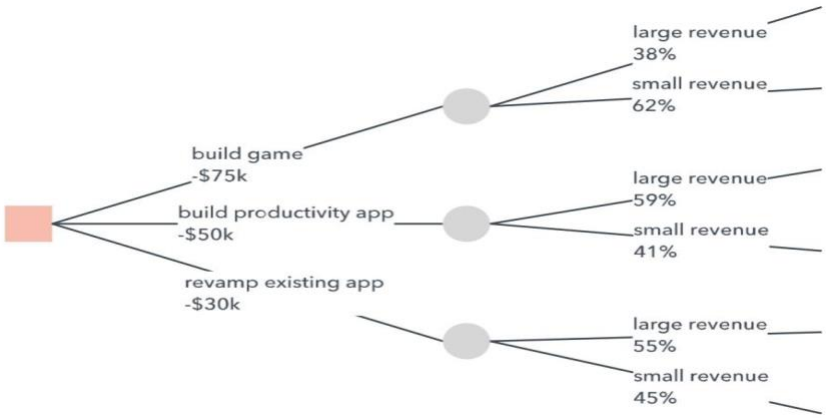
1.**Start with the main decision**. Draw a small box to represent this point, then draw a line from the box to the right for each possible solution or action. Label them accordingly.




2. **Add chance and decision nodes** to expand the tree as follows:

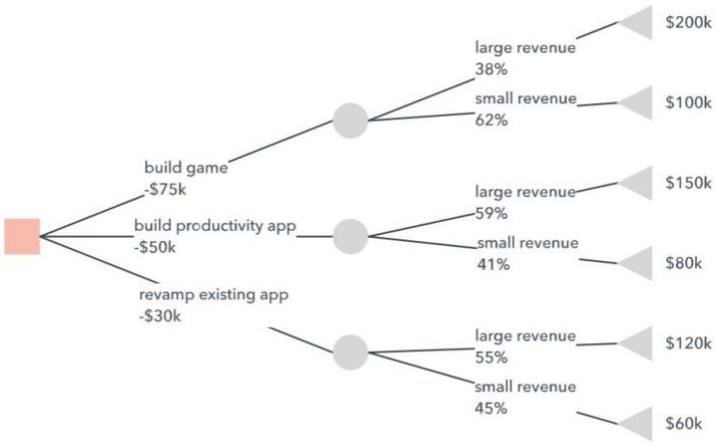
If another decision is necessary, draw another box.

 If the outcome is uncertain, draw a circle (circles represent chance nodes).If the problem is solved, leave it blank (for now).




From each decision node, draw possible solutions. From each chance node, draw lines representing possible outcomes. If you intend to analyze your options numerically, include the probability of each outcome and the cost of each action.

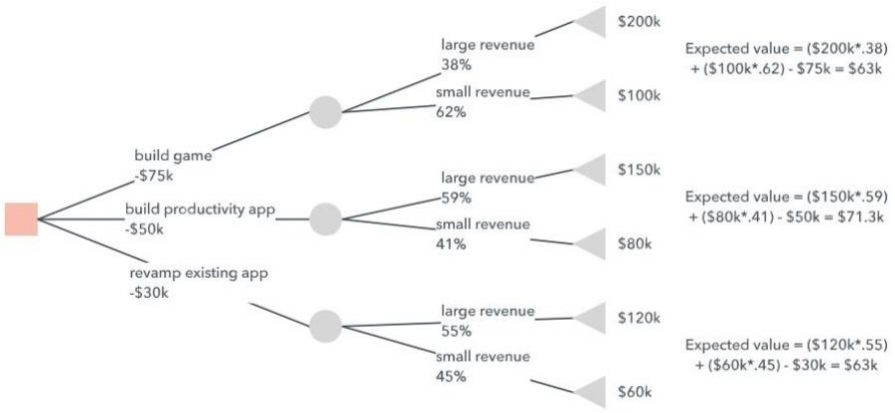
3.**Continue to expand until every line reaches an endpoint**, meaning that there are no more choices to be made or chance outcomes to consider. Then, assign a value to each possible outcome. It could be an abstract score or a financial value. Add triangles to signify endpoints.




## Decision tree analysis example

By calculating the expected utility or value of each choice in the tree, you can minimize risk and maximize the likelihood of reaching a desirable outcome.

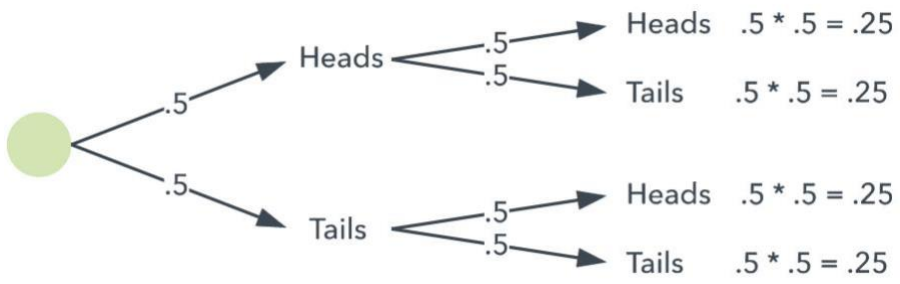
To calculate the expected utility of a choice, just subtract the cost of that decision from the expected benefits. The expected benefits are equal to the total value of all the outcomes that could result from that choice, with each value multiplied by the likelihood that it'll occur. Here's how we'd calculate these values for the example we made above:




When identifying which outcome is the most desirable, it's important to take the decision maker's utility preferences into account. For instance, some may prefer low-risk options while others are willing to take risks for a larger benefit.

When you use your decision tree with an accompanying probability model, you can use it to calculate the conditional probability of an event, or the likelihood that it'll happen, given that another event happens. To do so, simply start with the initial event, then follow the path from that event to the target event, multiplying the probability of each of those events together.

In this way, a decision tree can be used like a traditional tree diagram, which maps out the probabilities of certain events, such as flipping a coin twice.




## *Strengths and Weakness of Decision Tree approach*

The **strengths** of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The **weaknesses** of decision tree methods :

• Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.

• Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.

• Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

# Decision Table and Structured English

Decision table is a brief visual representation for specifying which actions to perform depending on given conditions. The information represented in decision tables can also be represented as decision trees or in a programming language using if-then-else and switch-case statements.

A decision table is a good way to settle with different combination inputs with their corresponding outputs and also called cause-effect table. Reason to call cause-effect table is a related logical diagramming technique called cause-effect graphing that is basically used to obtain the decision table.

## Importance of Decision Table:

- Decision tables are very much helpful in test design technique.
- It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.
- It provides a regular way of stating complex business rules, that is helpful for developers as well as for testers.
- It assists in development process with developer to do a better job. Testing with all combination might be impractical.
- A decision table is basically an outstanding technique used in both testing and requirements management.
- It is a structured exercise to prepare requirements when dealing with complex business rules.
- It is also used in model complicated logic.

## Components of a Decision Table

• Condition Stub − It is in the upper left quadrant which lists all the condition to be checked.

• Action Stub − It is in the lower left quadrant which outlines all the action to be carried out to meet such condition.

• Condition Entry − It is in upper right quadrant which provides answers to questions asked in condition stub quadrant.

• Action Entry − It is in lower right quadrant which indicates the appropriate action resulting from the answers to the conditions in the condition entry quadrant.

## Decision Table in test designing:

The entries in decision table are given by Decision Rules which define the relationships between combinations of conditions and courses of action. In rules section,

- Y shows the existence of a condition.
- N represents the condition, which is not satisfied.
- A blank - against action states it is to be ignored.
- X (or a check mark will do) against action states it is to be carried out.

## Blank Decision Table

| CONDITIONS | STEP 1 | STEP 2 | STEP 3 | STEP 4 |
|---|---|---|---|---|
| Condition 1 | | | | |
| Condition 2 | | | | |
| Condition 3 | | | | |
| Condition 4 | | | | |

## Decision Table: Combinations

| CONDITIONS | STEP 1 | STEP 2 | STEP 3 | STEP 4 |
|---|---|---|---|---|
| Condition 1 | Y | Y | N | N |
| Condition 2 | Y | N | Y | N |
| Condition 3 | Y | N | N | Y |
| Condition 4 | N | Y | Y | N |

For example, refer the following table −

| CONDITIONS | Rule 1 | Rule 2 | Rule 3 | Rule 4 |
|---|---|---|---|---|
| Advance payment made | Y | N | N | N |
| Purchase amount = Rs 10,000/- . | - | Y | Y | N |
| Regular Customer | - | Y | N | - |
| ACTIONS | | | | |
| Give 5% discount | X | X | - | - |
| Give no discount | - | - | X | X |

## *Advantage of Decision Table:*

- Any complex business flow can be easily converted into the test scenarios & test cases using this technique.

- Decision tables work iteratively that means the table created at the first iteration is used as input table for next tables. The iteration is done only if the initial table is not satisfactory.

- Simple to understand and everyone can use this method design the test scenarios & test cases.

- It provide complete coverage of test cases which help to reduce the rework on writing test scenarios & test cases.

- These tables guarantee that we consider every possible combination of condition values. This is known as its completeness property.

## Structured English

Structure English is derived from structured programming language which gives more understandable and precise description of process. It is based on procedural logic that uses construction and imperative sentences designed to perform operation for action.

- It is best used when sequences and loops in a program must be considered and the problem needs sequences of actions with decisions.

- It does not have strict syntax rule. It expresses all logic in terms of sequential decision structures and iterations.

For example, see the following sequence of actions −

```
if customer pays advance
  then
    Give 5% Discount
  else
    if purchase amount >=10,000
      then
        if  the customer is a regular customer
          then Give 5% Discount
        else  No Discount
      end if
    else No Discount
  end if
end if
```

Structured English is one more tool available to the analyst. It comes as an aid against the problems of ambiguous language in stating condition and actions in decisions and procedures. Here no trees or tables are employed, rather with narrative statements a procedure is described. Thus it does not show but states the decision rules. The analyst is first required to identify the conditions that occur in the process, subsequent decisions, which are to be made and the alternative actions to be taken.

Here the steps are clearly listed in the order in which they should be taken. There are no special symbols or formats involved unlike in the case of decision trees and tables, also the entire procedure can be stated quickly as only English like statements are used.

Structured English borrows heavily from structured programming as it uses logical construction and imperative statements designed to carry out instructions for actions. Using "IF", "THEN", "ELSE" and "So" statement decisions are made. In this structured description

terms from the data dictionary are widely used which makes the description compact and straight.

## Developing Structured Statements

Three basic types of statements are employed to describe the process.

1. Sequence Structures - A sequence structure is a single step or action included in a process. It is independent of the existence of any condition and when encountered it is always taken. Usually numerous such instructions are used together to describe a process.

2. Decision Structures - Here action sequences described are often included within decision structures that identify conditions. Therefore these structures occur when two or more actions can be taken as per the value of a specific condition. Once the condition is determined the actions are unconditional.

3. Iteration Structures- these are those structures, which are repeated, in routing operations such as DO WHILE statements.

#  Functional vs Object oriented approach

A programming paradigm is a style, or “way, ” of programming. Programming paradigms differ from one another based on the features and the style they support. There are several features that determine a programming paradigm such as modularity, objects, interrupts or events, control flow etc. Every programming paradigm has its own advantage so, it better to know where to use it before actually using it.

Object-oriented languages are good when you have a fixed set of operations on things, and as your code evolves, you primarily add new things. This can be accomplished by adding new classes which implement existing methods, and the existing classes are left alone.

Functional languages are good when you have a fixed set of things, and as your code evolves, you primarily add new operations on existing things. This can be accomplished by adding new functions which compute with existing data types, and the existing functions are left alone.

It is also possible to use both the programming paradigms according to our own need. As we have got languages like python, java, etc that supports both object oriented concept and are also functional by supporting various inbuilt functions.

# Function Oriented Design

In function-oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions.

Function oriented design inherits some properties of structured design where divide and conquer methodology is used.

This design mechanism divides the whole system into smaller functions, which provides means of abstraction by concealing the information and their operation.. These functional modules can share information among themselves by means of information passing and using information available globally.

Another characteristic of functions is that when a program calls a function, the function changes the state of the program, which sometimes is not acceptable by other modules. Function oriented design works well where the system state does not matter and program/functions work on input rather than on a state.

## Design Process

- The whole system is seen as how data flows in the system by means of data flow diagram.

- DFD depicts how functions changes data and state of entire system.

- The entire system is logically broken down into smaller units known as functions on the basis of their operation in the system.

- Each function is then described at large.

# Object Oriented Design

Object oriented design works around the entities and their characteristics instead of functions involved in the software system. This design strategies focuses on entities and its characteristics. The whole concept of software solution revolves around the engaged entities.

Let us see the important concepts of Object Oriented Design:

- Objects - All entities involved in the solution design are known as objects. For example, person, banks, company and customers are treated as objects. Every entity has some attributes associated to it and has some methods to perform on the attributes.

- Classes - A class is a generalized description of an object. An object is an instance of a class. Class defines all the attributes, which an object can have and methods, which defines the functionality of the object.

In the solution design, attributes are stored as variables and functionalities are defined by means of methods or procedures.

- Encapsulation - In OOD, the attributes (data variables) and methods (operation on the data) are bundled together is called encapsulation. Encapsulation not only bundles important information of an object together, but also restricts access of the data and methods from the outside world. This is called information hiding.

- Inheritance - OOD allows similar classes to stack up in hierarchical manner where the lower or sub-classes can import, implement and re-use allowed variables and methods from their immediate super classes. This property of OOD is known as inheritance. This makes it easier to define specific class and to create generalized classes from specific ones.

- Polymorphism - OOD languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned same name. This is called polymorphism, which allows a single interface performing tasks for different types. Depending upon how the function is invoked, respective portion of the code gets executed.

## ⊡ Difference between functional and Object oriented design

| Functional Programming | Object Oriented Programming |
|---|---|
| This programming paradigm emphasizes on the use of functions where each function performs a specific task. | This programming paradigm is based on object oriented concept. Classes are used where instance of objects are created |
| Fundamental elements used are variables and functions.The data in the functions are immutable(cannot be changed after creation). | Fundamental elements used are objects and methods and the data used here are mutable data. |
| Importance is not given to data but to functions. | Importance is given to data rather than procedures. |
| It follows declarative programming model. | It follows imperative programming model. |
| It uses recursion for iteration. | It uses loops for iteration. |

| Functional Programming | Object Oriented Programming |
| --- | --- |
| It is parallel programming supported. | It does not support parallel programming. |
| The statements in this programming paradigm does not need to follow a particular order while execution. | The statements in this programming paradigm need to follow a order i.e., bottom up approach while execution. |
| Does not have any access specifier. | Has three access specifiers namely, Public, Private and Protected. |
| To add new data and functions is not so easy. | Provides and easy way to add new data and functions. |
| No data hiding is possible. Hence, Security is not possible. | Provides data hiding. Hence, secured programs are possible. |

# MODULE-3

## Coding

The coding is the process of transforming the design of a system into a computer language format. This coding phase of software development is concerned with software translating design specification into the source code. It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified.

Coding is done by the coder or programmers who are independent people than the designer. The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a later stage. The cost of testing and maintenance can be significantly reduced with efficient coding.

## Goals of Coding

1. To translate the design of system into a computer language format: The coding is the process of transforming the design of a system into a computer language format, which can be executed by a computer and that perform tasks as specified by the design of operation during the design phase.

2. To reduce the cost of later phases: The cost of testing and maintenance can be significantly reduced with efficient coding.

3. Making the program more readable: Program should be easy to read and understand. It increases code understanding having readability and understandability as a clear objective of the coding activity can itself help in producing more maintainable software.

For implementing our design into code, we require a high-level functional language. A programming language should have the following characteristics:

## Characteristics of Programming Language

Following are the characteristics of Programming Language:

Readability: A good high-level language will allow programs to be written in some methods that resemble a quite-English description of the underlying functions. The coding may be done in an essentially self-documenting way.

Portability: High-level languages, being virtually machine-independent, should be easy to develop portable software.

Generality: Most high-level languages allow the writing of a vast collection of programs, thus relieving the programmer of the need to develop into an expert in many diverse languages.

Brevity: Language should have the ability to implement the algorithm with less amount of code. Programs mean in high-level languages are often significantly shorter than their low-level equivalents.

Error checking: A programmer is likely to make many errors in the development of a computer program. Many high-level languages invoke a lot of bugs checking both at compile-time and run-time.

Cost: The ultimate cost of a programming language is a task of many of its characteristics.

Quick translation: It should permit quick translation.

Efficiency: It should authorize the creation of an efficient object code.

Modularity: It is desirable that programs can be developed in the language as several separately compiled modules, with the appropriate structure for ensuring self-consistency among these modules.

Widely available: Language should be widely available, and it should be feasible to provide translators for all the major machines and all the primary operating systems.

A coding standard lists several rules to be followed during coding, such as the way variables are to be named, the way the code is to be laid out, error return conventions, etc.

## Coding Standards

General coding standards refers to how the developer writes code, so here we will discuss some essential standards regardless of the programming language being used.

The following are some representative coding standards:

1. Indentation: Proper and consistent indentation is essential in producing easy to read and maintainable programs.

Indentation should be used to:

- Emphasize the body of a control structure such as a loop or a select statement.
- Emphasize the body of a conditional statement

o Emphasize a new scope block

2. Inline comments: Inline comments analyze the functioning of the subroutine, or key aspects of the algorithm shall be frequently used.

3. Rules for limiting the use of global: These rules file what types of data can be declared global and what cannot.

4. Structured Programming: Structured (or Modular) Programming methods shall be used. "GOTO" statements shall not be used as they lead to "spaghetti" code, which is hard to read and maintain, except as outlined line in the FORTRAN Standards and Guidelines.

5. Naming conventions for global variables, local variables, and constant identifiers: A possible naming convention can be that global variable names always begin with a capital letter, local variable names are made of small letters, and constant names are always capital letters.

6. Error return conventions and exception handling system: Different functions in a program report the way error conditions are handled should be standard within an organization. For example, different tasks while encountering an error condition should either return a 0 or 1 consistently.

## Coding Guidelines

General coding guidelines provide the programmer with a set of the best methods which can be used to make programs more comfortable to read and maintain. Most of the examples use the C language syntax, but the guidelines can be tested to all languages.

The following are some representative coding guidelines recommended by many software development organizations.

1. Line Length: It is considered a good practice to keep the length of source code lines at or below 80 characters. Lines longer than this may not be visible properly on some terminals and tools. Some printers will truncate lines longer than 80 columns.

2. Spacing: The appropriate use of spaces within a line of code can improve readability.

Example:

Bad: cost=price+(price*sales_tax)

fprintf(stdout ,"The total cost is %5.2f\n",cost);

Better: cost = price + ( price * sales_tax )

fprintf (stdout,"The total cost is %5.2f\n",cost);

3. The code should be well-documented: As a rule of thumb, there must be at least one comment line on the average for every three-source line.

4. The length of any function should not exceed 10 source lines: A very lengthy function is generally very difficult to understand as it possibly carries out many various functions. For the same reason, lengthy functions are possible to have a disproportionately larger number of bugs.

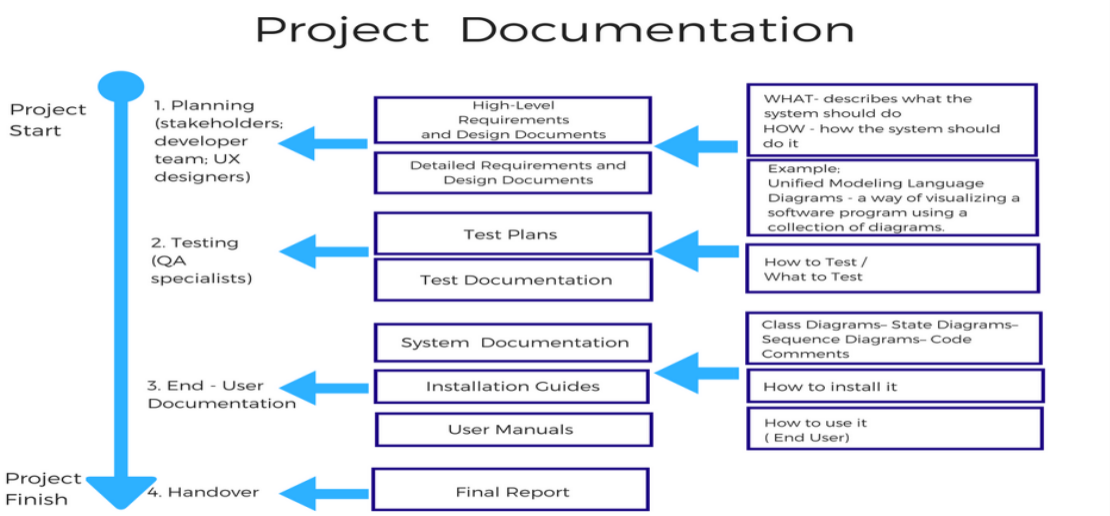
5. Do not use goto statements: Use of goto statements makes a program unstructured and very tough to understand.

6. Inline Comments: Inline comments promote readability.

7. Error Messages: Error handling is an essential aspect of computer programming. This does not only include adding the necessary logic to test for and handle errors but also involves making error messages meaningful.

# Documentation

Documentation in software engineering is the umbrella term that encompasses all written documents and materials dealing with a software product's development and use. All software development products, whether created by a small team or a large corporation, require some related documentation. And different types of documents are created through the whole software development lifecycle (SDLC). Documentation exists to explain product functionality, unify project-related information, and allow fordiscussing all significant questions arising between stakeholders and developers.

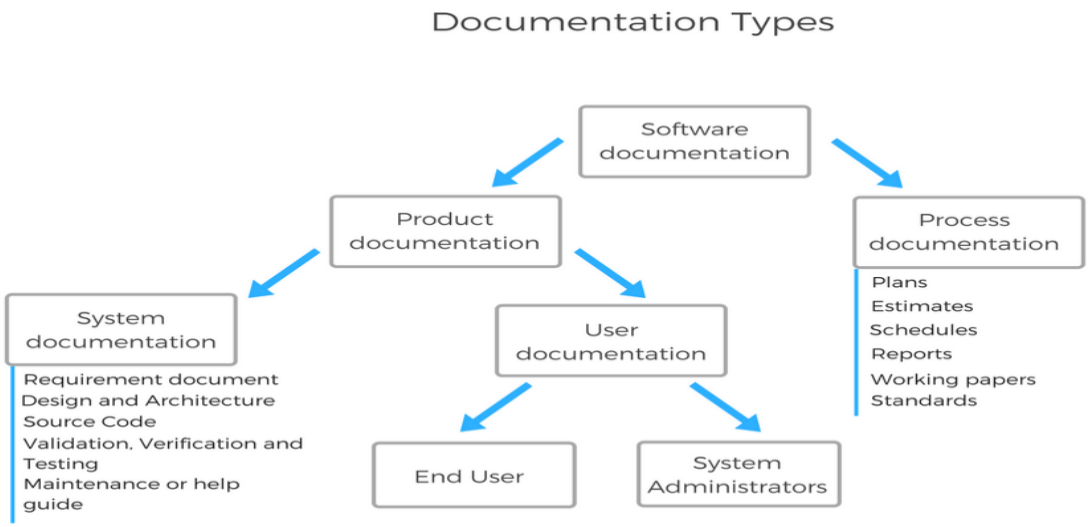



On top of that, documentation errors can set gaps between the visions of stakeholders and engineers and, as a result, a proposed solution won't meet stakeholders expectations. Consequently, managers should pay a lot of attention to documentation quality.

## Types of documentation

The main **goal** of effective documentation is to ensure that developers and stakeholders are headed in the same direction to accomplish the objectives of the project. To achieve them, plenty of documentation types exist.

Adhering to the following classifications.



All software documentation can be divided into two main categories:

- Product documentation
- Process documentation

**Product documentation** describes the product that is being developed and provides instructions on how to perform various tasks with it. Product documentation can be broken down into:

- System documentation and
- User documentation

**System documentation** represents documents that describe the system itself and its parts. It includes requirements documents, design decisions, architecture descriptions, program source code, and help guides.

**User documentation** covers manuals that are mainly prepared for end-users of the product and system administrators. User documentation includes tutorials, user guides, troubleshooting manuals, installation, and reference manuals.

**Process documentation** represents all documents produced during development and maintenance that describe… well, process. The common examples of process documentation are project plans, test schedules, reports, standards, meeting notes, or even business correspondence.

The main difference between process and product documentation is that the first one record the process of development and the second one describes the product that is being developed.

## Product: System documentation

System documentation provides an overview of the system and helps engineers and stakeholders understand the underlying technology. It usually consists of the requirements document, architecture design, source code, validation docs, verification and testing info, and a maintenance or help guide. It’s worth emphasizing that this list isn’t exhaustive. So, let’s have a look at the details of the main types.

## Requirements document

A requirements document provides information about the system functionality. Generally, requirements are the statements of what a system should do. It contains business rules, user stories, use cases, etc. This document should be clear and shouldn't be an extensive and solid wall of text. It should contain enough to outline the product's purpose, its features, functionalities, and behavior.

The best practice is to write a requirement document using a single, consistent template that all team members adhere to. The one web-page form will help you keep the document concise and save the time spent on accessing the information. Here's a look at an example of a one-web-page product-requirements document to understand various elements that should be included in your PRD. Nevertheless, you should remember that this isn't the one and only way to compile this document.

# Structured Programming

In structured programming, we sub-divide the whole program into small modules so that the program becomes easy to understand. The purpose of structured programming is to linearize control flow through a computer program so that the execution sequence follows the sequence in which the code is written. The dynamic structure of the program than resemble the static structure of the program. This enhances the readability, testability, and modifiability of the program. This linear flow of control can be managed by restricting the set of allowed applications construct to a single entry, single exit formats.

## Why we use Structured Programming?

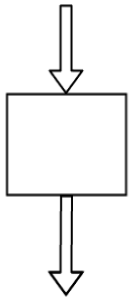
We use structured programming because it allows the programmer to understand the program easily. If a program consists of thousands of instructions and an error occurs then it is complicated to find that error in the whole program, but in structured programming, we can easily detect the error and then go to that location and correct it. This saves a lot of time.

**These are the following rules in structured programming:**

## Structured Rule One: Code Block

If the entry conditions are correct, but the exit conditions are wrong, the error must be in the block. This is not true if the execution is allowed to jump into a block. The error might be anywhere in the program. Debugging under these circumstances is much harder.

**Rule 1 of Structured Programming:** A code block is structured, as shown in the figure. In flow-charting condition, a box with a single entry point and single exit point are structured. Structured programming is a method of making it evident that the program is correct.

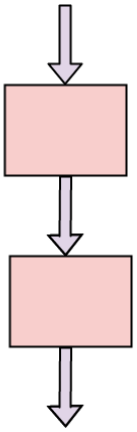


Rule1: Code block is structured

## Structure Rule Two: Sequence

A sequence of blocks is correct if the exit conditions of each block match the entry conditions of the following block. Execution enters each block at the block's entry point and leaves through the block's exit point. The whole series can be regarded as a single block, with an entry point and an exit point.

**Rule 2 of Structured Programming:** Two or more code blocks in the sequence are structured, as shown in the figure.



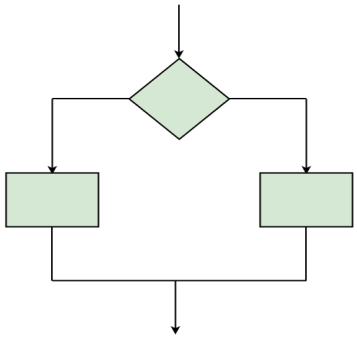
Rule2: A sequence of code blocks is structured

## Structured Rule Three: Alternation

If-then-else is frequently called alternation (because there are alternative options). In structured programming, each choice is a code block. If alternation is organized as in the flowchart at right, then there is one entry point (at the top) and one exit point (at the bottom). The structure should be coded so that if the entry conditions are fulfilled, then the exit conditions are satisfied (just like a code block).

**Rule 3 of Structured Programming:** The alternation of two code blocks is structured, as shown in the figure.

An example of an entry condition for an alternation method is: register $8 includes a signed integer. The exit condition may be: register $8 includes the absolute value of the signed number. The branch structure is used to fulfill the exit condition.

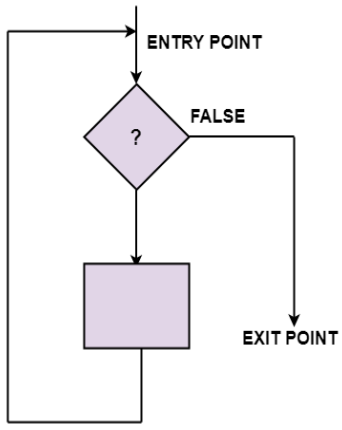


Rule 3: An alternation of code blocks is structured

## Structured Rule 4: Iteration

Iteration (while-loop) is organized as at right. It also has one entry point and one exit point. The entry point has conditions that must be satisfied, and the exit point has requirements that will be fulfilled. There are no jumps into the form from external points of the code.

**Rule 4 of Structured Programming:** The iteration of a code block is structured, as shown in the figure.

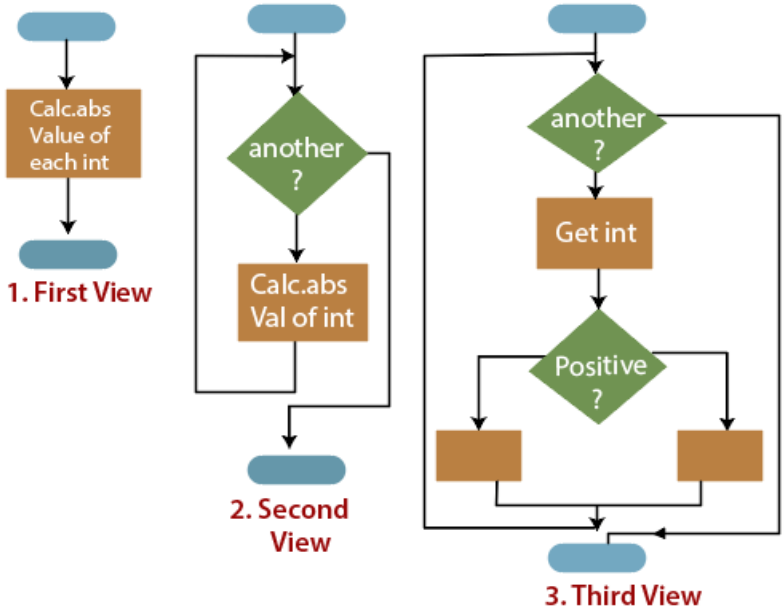



Rule 4: Iteration of code blocks is structured

## Structured Rule 5: Nested Structures

In flowcharting conditions, any code block can be spread into any of the structures. If there is a portion of the flowchart that has a single entry point and a single exit point, it can be summarized as a single code block.

**Rule 5 of Structured Programming:** A structure (of any size) that has a single entry point and a single exit point is equivalent to a code block. For example, we are designing a program to go through a list of signed integers calculating the absolute value of each one. We may **(1)** first regard the program as one block, then **(2)** sketch in the iteration required, and finally **(3)** put in the details of the loop body, as shown in the figure.




The other control structures are the case, do-until, do-while, and for are not needed. However, they are sometimes convenient and are usually regarded as part of structured programming. In assembly language, they add little convenience.

## Advantages of Structured Programming Approach:

- Easier to read and understand
- User Friendly
- Easier to Maintain
- Mainly problem based instead of being machine based
- Development is easier as it requires less effort and time
- Easier to Debug
- Machine-Independent, mostly.

## Disadvantages of Structured Programming Approach:

1. Since it is Machine-Independent, So it takes time to convert into machine code.
2. The converted machine code is not the same as for assembly language.
3. The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.

4. Usually the development in this approach takes longer time as it is language-dependent. Whereas in the case of assembly language, the development takes lesser time as it is fixed for the machine.

# OO Programming

Object-oriented programming (OOP) is a programming paradigm based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

The important features of object–oriented programming are −

- Bottom–up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object's data
- Interaction between objects through functions
- Reusability of design through creation of new classes by adding features to existing classes

Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Grady Booch has defined object–oriented programming as *"a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships"*

## What Are Class & Object?

It is the basic concept of OOP; an extended concept of the structure used in C. It is an abstract and user-defined data type. It consists of several variables and functions. The primary purpose of the class is to store data and information. The members of a class define the behaviour of the class. A class is the blueprint of the object, but also, we can say the implementation of the class is the object. The class is not visible to the world, but the object is.

Class car

```
{
   int car_id;
   char colour[4];
   float engine_no;
   double distance;

   void distance_travelled();
   float petrol_used();
   char music_player();
   void display();
}
```

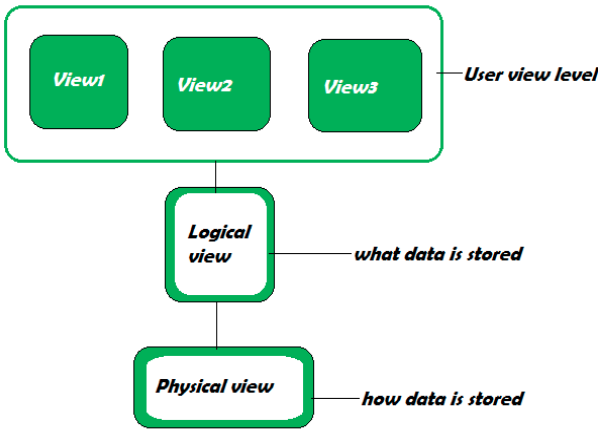
Here, the class car has properties car_id, colour, engine_no and distance. It resembles the real-world car that has the same specifications, which can be declared public (visible to everyone outside the class), protected and private (visible to none). Also, there are some methods such as distance_travelled(), petrol_used(), music_player() and display(). In the code given below, the car is a class and c1 is an object of the car.

```
#include <iostream>
using namespace std;

class car {
public:
   int car_id;
   double distance;

   void distance_travelled();

   void display(int a, int b)
   {
      cout << "car id is=\t" << a << "\ndistance travelled =\t" << b + 5;
   }
};

int main()
{
   car c1; // Declare c1 of type car
   c1.car_id = 321;
   c1.distance = 12;
   c1.display(321, 12);
```

```
    return 0;
}
```

## Data Abstraction –

Abstraction refers to the act of representing important and special features without including the background details or explanation about that feature. Data abstraction simplifies database design.



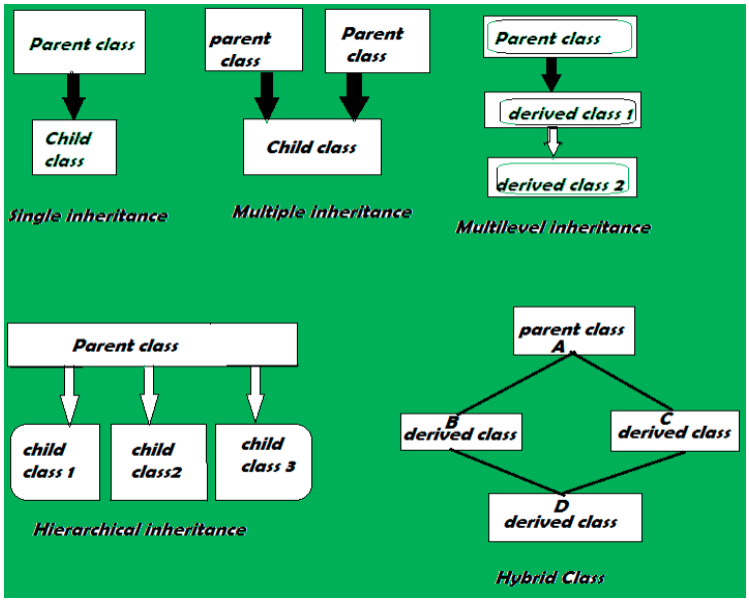

1. **Physical Level:**
   It describes how the records are stored, which are often hidden from the user. It can be described with the phrase, “block of storage.”
2. **Logical Level:**
   It describes data stored in the database and the relationships between the data. The programmers generally work at this level as they are aware of the functions needed to maintain the relationships between the data.
3. **View Level:**
   Application programs hide details of data types and information for security purposes. This level is generally implemented with the help of GUI, and details that are meant for the user are shown.

## Encapsulation –

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit, e.g., a class in Java. This concept is often used to hide the internal state representation of an object from the outside.

## Inheritance –

Inheritance is the ability of one class to inherit capabilities or properties of another class, called the parent class. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class which possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

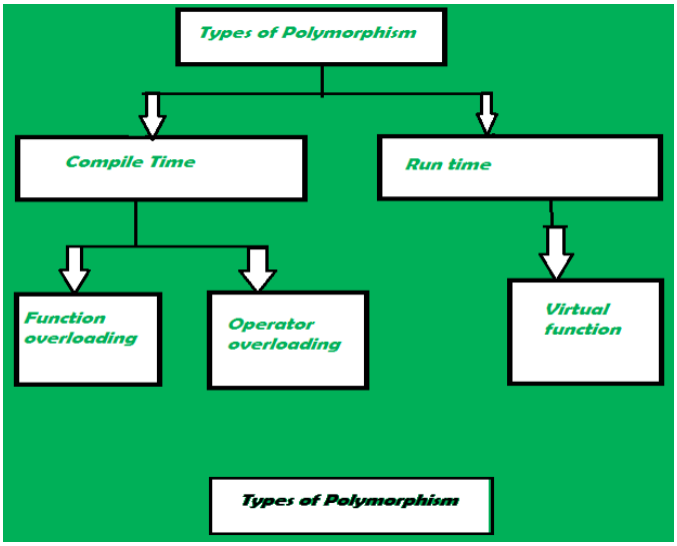



```
import java.io.*;

class GFG {
    public static void main(String[] args)
    {
        System.out.println("GfG!");

        Dog dog = new Dog();
        dog.name = "Bull dog";
        dog.color = "Brown";
        dog.bark();
        dog.run();

        Cat cat = new Cat();
        cat.name = "Rag doll";
        cat.pattern = "White and slight brownish";
        cat.meow();
        cat.run();

        Animal animal = new Animal();

        animal.name = "My favourite pets";

        animal.run();
    }
}
```

```
class Animal {
   String name;
   public void run()
   {

      System.out.println("Animal is running!");
   }
}

class Dog extends Animal {

/// the class dog is the child and animal is the parent

   String color;
   public void bark()
   {
      System.out.println(name + " Wooh ! Wooh !"
                  + "I am of colour " + color);
   }
}

class Cat extends Animal {

   String pattern;

   public void meow()
   {
      System.out.println(name + " Meow ! Meow !"
                  + "I am of colour " + pattern);
   }
}
```

## Polymorphism –

Polymorphism is the ability of data to be processed in more than one form. It allows the performance of the same task in various ways. It consists of method overloading and method overriding, i.e., writing the method once and performing a number of tasks using the same method name.




```
#include <iostream>
using namespace std;

void output(float);
void output(int);
void output(int, float);

int main()
{
   cout << "\nGfG!\n";
   int a = 23;
   float b = 2.3;

   output(a);
   output(b);
   output(a, b);

   return 0;
}

void output(int var)
{ // same function name but different task
   cout << "Integer number:\t" << var << endl;
}

void output(float var)
{ // same function name but different task
   cout << "Float number:\t" << var << endl;
```

```
}

void output(int var1, float var2)
{ // same function name but different task
   cout << "Integer number:\t" << var1;
   cout << " and float number:" << var2;
}
```

## Some important points to know about OOP:

1. OOP treats data as a critical element.
2. Emphasis is on data rather than procedure.
3. Decomposition of the problem into simpler modules.
4. Doesn't allow data to freely flow in the entire system, ie localized control flow.
5. Data is protected from external functions.

## Advantages of OOPs –

- It models the real world very well.
- With OOP, programs are easy to understand and maintain.
- OOP offers code reusability. Already created classes can be reused without having to write them again.
- OOP facilitates the quick development of programs where parallel development of classes is possible.
- With OOP, programs are easier to test, manage and debug.

## Disadvantages of OOP –

- With OOP, classes sometimes tend be over-generalised.
- The relations among classes become superficial at times.
- The OOP design is tricky and requires appropriate knowledge. Also, one needs to do proper planning and design for OOP programming.
- To program with OOP, the programmer needs proper skills such as that of design, programming and thinking in terms of objects and classes etc.

# Information Hiding

Information hiding is the process of hiding the details of an object or function. The hiding of these details results in an abstraction, which reduces the external complexity and makes the object or function easier to use. In addition, information hiding effectively decouples the calling code from the internal workings of the object or function being called, which makes it possible to change the hidden portions without having to also change the calling code. Encapsulation is a common technique programmers use to implement information hiding.

# Testing

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free.If testing is done successfully it will remove all the errors from the software.

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

## Software Validation

Validation is process of examining whether or not the software satisfies the user requirements. It is carried out at the end of the SDLC. If the software matches requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes on user requirements.

## Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications ?"
- Verifications concentrates on the design and system specifications.

Target of the test are -

- **Errors** - These are actual coding mistakes made by developers. In addition, there is a difference in output of software and desired output, is considered as an error.
- **Fault** - When error exists fault occurs. A fault, also known as a bug, is a result of an error which can cause system to fail.
- **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system

## Principles of Testing:-

(i)All the test should meet the customer requirements
(ii) To make our software testing should be performed by third party
(iii) Exhaustive testing is not possible.As we need the optimal amount of testing based on the risk assessment of the application.
(iv) All the test to be conducted should be planned before implementing it
(v) It follows pareto rule(80/20 rule) which states that 80% of errors comes from 20% of program components.
(vi) Start testing with small parts and extend it to large parts.

# Manual Vs Automated Testing

Testing can either be done manually or using an automated testing tool:

- **Manual** - This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.

  Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.

- **Automated** This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

A test needs to check if a webpage can be opened in Internet Explorer. This can be easily done with manual testing. But to check if the web-server can take the load of 1 million users, it is quite impossible to test manually.

There are software and hardware tools which helps tester in conducting load testing, stress testing, regression testing.

# Testing Approaches

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

Exhaustive tests are the best-desired method for a perfect testing. Every single possible value in the range of the input and output values is tested. It is not possible to test each and every value in real world scenario if the range of values is large.

## Black-box testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

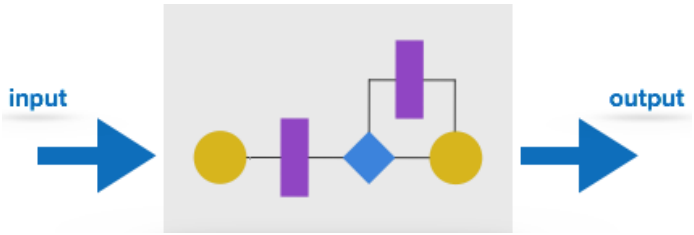

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** - The system changes state on provision of input. These systems are tested based on their states and input.

## White-box testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing.




In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- **Control-flow testing** - The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

## Types of Testing:-

*1.* Unit Testing

It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units.It is often done by programmer by using sample input and observing its corresponding outputs.
Example:

a) In a program we are checking if loop, method or
function is working fine
b) Misunderstood or incorrect, arithmetic precedence.
c) Incorrect initialization

*2.* Integration Testing

The objective is to take unit tested components and build a program structure that has been dictated by design.Integration testing is testing in which a group of components are combined to produce output.

Integration testing is of four types: (i) Top down (ii) Bottom up (iii) Sandwich (iv) Big-Bang Example

(a) Black Box testing:- It is used for validation.
In this we ignores internal working mechanism and

focuses on **what is the output?**.

(b) White Box testing:- It is used for verification.
In this we focus on internal mechanism i.e.
**how the output is achieved?**

*3.* Regression Testing

Every time new module is added leads to changes in program. This type of testing make sure that whole component works properly even after adding components to the complete program.
Example

In school record suppose we have module staff, students and finance combining these modules and checking if on integration these module works fine is regression testing

*4.* Smoke Testing

This test is done to make sure that software under testing is ready or stable for further testing It is called smoke test as testing initial pass is done to check if it did not catch the fire or smoked in the initial switch on.
Example:

If project has 2 modules so before going to module make sure that module 1 works properly

*5.* Alpha Testing

This is a type of validation testing.It is a type of *acceptance testing* which is done before the product is released to customers. It is typically done by QA people.
Example:

When software testing is performed internally within the organization

*6.* Beta Testing

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for the limited number of users for testing in real time environment
Example:

When software testing is performed for the limited number of people

*7.* System Testing

In this software is tested such that it works fine for different operating system.It is covered under the black box testing technique. In this we just focus on required input and output without focusing on internal working.

In this we have security testing, recovery testing , stress testing and performance testing

Example:

```
This include functional as well as non functional
testing
```

*8.* Stress Testing

In this we gives unfavorable conditions to the system and check how they perform in those condition.

Example:

```
(a) Test cases that require maximum memory or other
    resources are executed
(b) Test cases that may cause thrashing in a virtual
    operating system
(c) Test cases that may cause excessive disk requirement
```

*9.* Performance Testing

It is designed to test the run-time performance of software within the context of an integrated system.It is used to test speed and effectiveness of program.

Example:

```
Checking number of processor cycles.
```

# Testing Documentation

Testing documents are prepared at different stages -

## Before Testing

Testing starts with test cases generation. Following documents are needed for reference –

- **SRS document** - Functional Requirements document
- **Test Policy document** - This describes how far testing should take place before releasing the product.
- **Test Strategy document** - This mentions detail aspects of test team, responsibility matrix and rights/responsibility of test manager and test engineer.

- **Traceability Matrix document** - This is SDLC document, which is related to requirement gathering process. As new requirements come, they are added to this matrix. These matrices help testers know the source of requirement. They can be traced forward and backward.

## While Being Tested

The following documents may be required while testing is started and is being done:

- **Test Case document** - This document contains list of tests required to be conducted. It includes Unit test plan, Integration test plan, System test plan and Acceptance test plan.
- **Test description** - This document is a detailed description of all test cases and procedures to execute them.
- **Test case report** - This document contains test case report as a result of the test.
- **Test logs** - This document contains test logs for every test case report.

## After Testing

The following documents may be generated after testing :

- **Test summary** - This test summary is collective analysis of all test reports and logs. It summarizes and concludes if the software is ready to be launched. The software is released under version control system if it is ready to launch.

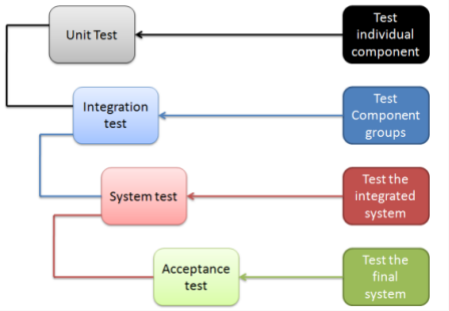
# Testing vs. Quality Control, Quality Assurance and Audit

We need to understand that software testing is different from software quality assurance, software quality control and software auditing.

- **Software quality assurance** - These are software development process monitoring means, by which it is assured that all the measures are taken as per the standards of organization. This monitoring is done to make sure that proper software development methods were followed.
- **Software quality control** - This is a system to maintain the quality of software product. It may include functional and non-functional aspects of software product, which enhance the goodwill of the organization. This system makes sure that the customer is receiving quality product for their requirement and the product certified as 'fit for use'.
- **Software audit** - This is a review of procedure used by the organization to develop the software. A team of auditors, independent of development team examines the software process, procedure, requirements and other aspects of SDLC. The purpose of software audit is to check that software and its development process, both conform standards, rules and regulations.

# Levels of testing

**SOFTWARE TESTING LEVELS** are the different stages of the software development lifecycle where testing is conducted. There are four levels of software testing: Unit >> Integration >> System >> Acceptance.

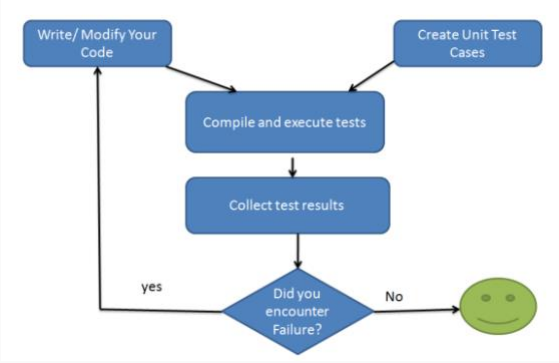



**Overview of Software Testing Levels**

## Unit Testing

The smallest independent and testable part of the source code is referred to as a unit. It is the first step in software testing environment and is generally conducted by the developers or their team mates. This form of testing is rarely performed by software testers. In order to perform integration testing it is important to first complete the unit testing for all the units. In order to perform unit testing it is important to have well defined unit test plan and unit test cases.

There are several benefits of unit testing. First of all, you get the confidence for going ahead with the integration testing only when you are sure that all units are working correctly. When you start unit testing in parallel to development it may look like a slow process as many defects are uncovered during this stage and several changes are made to the code. However, with time the code is refined and number of defects begins to reduce. So, the foundation of the software is strong and in the later stages the software development is carried out at a much faster pace thereby saving a lot of time.




**Unit Testing**

If unit testing is carried out properly then it would also result in a lot of cost saving as the cost of fixing a defect in the final stages of software development are much higher than fixing them in the initial stages.

Unit testing is carried on the smallest testable component of the project so the number of test cases and test data are less, and it is not always possible to check all the scenarios for functional and information flow of software application. So, there are many test cases that can be tested only after the unit has been merged with other units to form a bigger component.

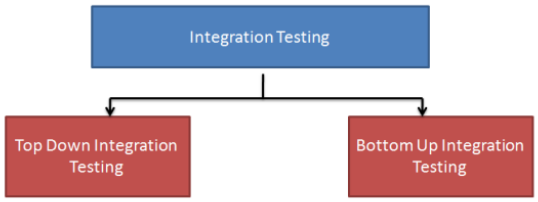
## Integration Testing

Once the unit testing phase is over, it is time to move on to integration testing. During integration testing the tester checks how one or more units interact with each other and produce output for various scenarios. This form of testing is carried out a software testing engineer.

In this form of testing a lot of defects related to functional, requirement and performance levels are uncovered. Unit testing confirms that various units are able to perform as per the requirement individually but integration testing confirms whether these independent units are able to perform as per expectations when integrated together. Integration testing can be broadly classified into:

1. Big bang
2. Top down and
3. Bottom up approach

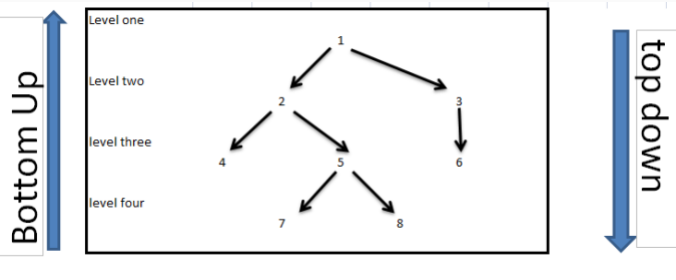
As the name suggests, big bang form of testing all the modules are combined to form a complete system and then tested for bugs.




**Integration Testing**

Top down is systematic approach where the top level modules are first tested and then one by one the sub modules are added and tested. The Bottom up approach is just the opposite of top down. In this case the lower most modules are tested first and step by step the higher

level modules are added and tested. Generally the bottom up approach is followed first in software testing followed by top-down testing.

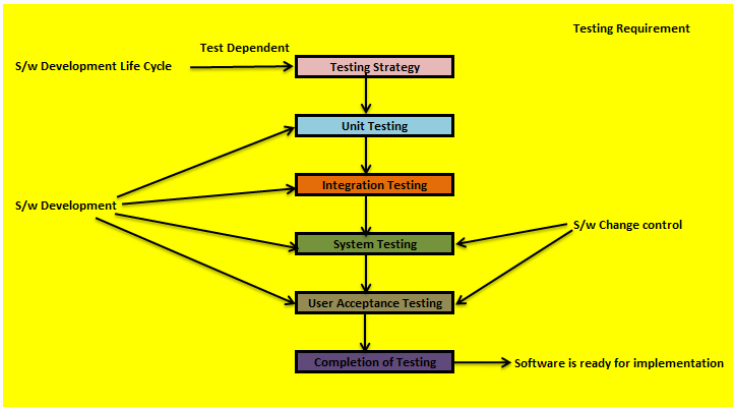



**Top down and Bottom up Integration Testing**

In Top down approach integration testing stubs can be used as handles in case of modules or sub programs that are not available or not ready. These are dummy modules used at low levels. In similar way in case of bottom up approach is a main program is not available then calling program referred to as driver can be used as a replacement to complete testing process.

## System Testing

Once the integration testing phase gets successfully completed it is time to move on to system testing where the system as a whole with all components well integrated is ready for further testing. This is where the software is not only tested for performance but also for adherence to quality standards. As the system is tested as a whole to see if it is in compliance with the functional and technical specifications and the quality standards defined by the organization, it is important that this form of testing is carried out by a highly skilled testing team. For this form of testing it is very important to create a scenario similar to the real time scenario where the system will be deployed.



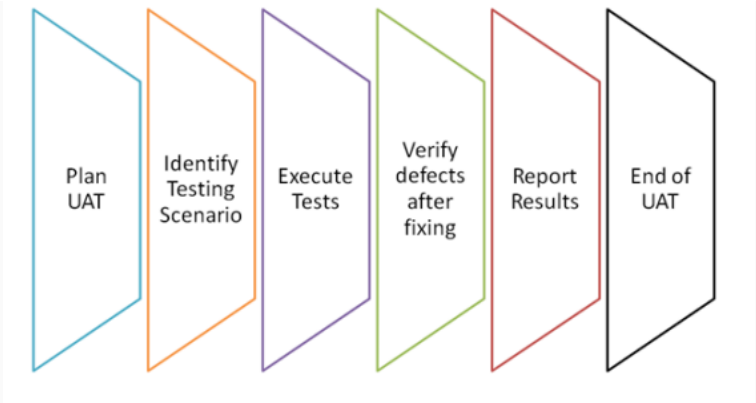

**System Testing**

System testing is purely black box testing. The system is checked as per the requirement specifications. The testing is carried out from the user's point of view. This type of testing is carried out to check the behavior of the application, software design and expectation of the end user. System testing validates and verifies both Application architecture and business requirements of the client.

## Acceptance Testing

Once the system has been thoroughly tested via unit, integration and system testing it is time for the quality assurance team to come and have a look at the system and test it for quality with the help of predefined test scenarios and test cases. The software is tested for accuracy. The acceptance testing looks at the system from various angles: right from cosmetic looks to internal functioning. This form of test is very crucial because there are legal as well as contractual requirements associated with the software for it to be accepted by the client. Acceptance testing can be of following types:

1. User Acceptance testing: This form of testing is carried out by the actual user before the software is accepted. It can be performed at the user's site or in the software organization where the software was developed.
2. Operation Acceptance testing: this form of testing is done to ensure that all the processes and procedures are in place so that the system can be used easily and also be maintained easily.
3. Contract and regulation acceptance testing: is carried out to ensure that the software is in line with all the necessary government, legal and safety standards.
4. Alpha testing: is carried to ensure that the product is of good quality and also to prepare the system for beta testing. This form of testing is performed towards the end of software development where the system can be tested as a whole. This can be a long process as the testers are looking into quality as well as engineering aspects of the developed software. This form of testing is carried out by test engineers and other team members and the product is tested from the point of view of a customer.
5. Beta testing: Once the alpha testing is over, beta testing follows in order to improve the quality of the product and see that the product is as per the requirement of the customer. This form of testing is done a couple of days or weeks before the launch of the product. Beta testing can take a few days but may not take as much time as alpha testing as chances of defect detection are pretty low during this time. It is carried in the real world scenario with the people who are actually going to use the product.

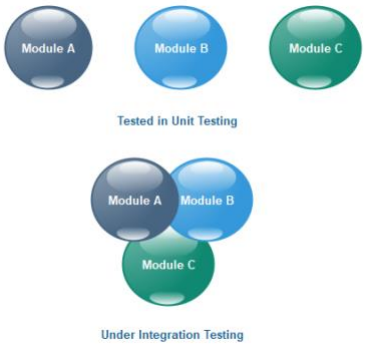



**Acceptance Testing**

# Integration testing

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.




## Reason Behind Integration Testing

Although all modules of software application already tested in unit testing, errors still exist due to the following reasons:

1. Each module is designed by individual software developer whose programming logic may differ from developers of other modules so; integration testing becomes essential to determine the working of software modules.
2. To check the interaction of software modules with the database whether it is an erroneous or not.
3. Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing becomes mandatory.
4. Incompatibility between modules of software could create errors.
5. To test hardware's compatibility with software.
6. If exception handling is inadequate between modules, it can create bugs.

# Integration Testing Techniques

Any testing technique (Blackbox, Whitebox, and Greybox) can be used for Integration Testing; some are listed below:

## Black Box Testing

- State Transition technique
- Decision Table Technique
- Boundary Value Analysis
- All-pairs Testing
- Cause and Effect Graph
- Equivalence Partitioning
- Error Guessing

## White Box Testing

- Data flow testing
- Control Flow Testing
- Branch Coverage Testing
- Decision Coverage Testing

# Methodologies of Integration Testing

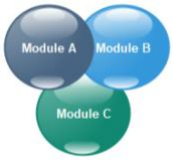
There are two basic methods for Integration Testing:

1. Big Bang
2. Incremental.

## Big Bang Method

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.




### *Advantages:*

- It is convenient for small size software systems.

### *Disadvantages:*

- Identification of defect is difficult.
- Small modules missed easily.
- Time provided for testing is very less.

## Incremental Approach

In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.

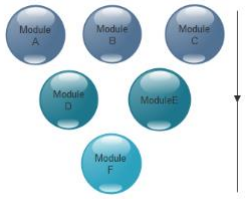
Incremental Approach is carried out by further methods:

- Top Down
- Bottom Up

- Hybrid Testing

## Top-Down Method

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first.



Top Down Method

**Advantages:**

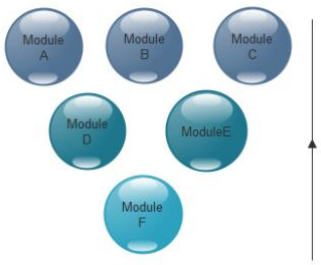
- Identification of defect is difficult.
- An early prototype is possible.

**Disadvantages:**

- Due to the high number of stubs, it gets quite complicated.
- Lower level modules are tested inadequately.
- Critical Modules are tested first so that fewer chances of defects.

## Bottom Up Method

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect.



Bottom Up Method

**Advantages:**

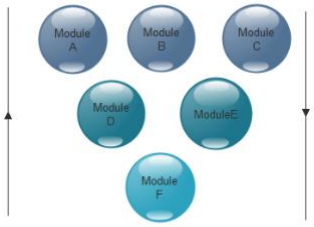
- Identification of defect is easy.
- Do not need to wait for the development of all the modules as It saves time.

**Disadvantages:**

- Critical modules are tested last due to which the defects can occur.
- There is no possibility of an early prototype.

## Hybrid Testing Method

In this approach, both **Top-Down** and **Bottom-Up** approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously. There is less possibility of occurrence of defect because each module interface is tested.



Hybrid Method

**Advantages:**

- The hybrid method provides features of both Bottom Up and Top Down methods.
- It is most time reducing method.
- It provides complete testing of all modules.

**Disadvantages:**

- This method needs a higher level of concentration as the process carried out in both directions simultaneously.
- Complicated method.

## Guidelines for Integration Testing

- First, determine the test case strategy through which executable test cases can be prepared according to test data.
- Examine the structure and architecture of the application and identify the crucial modules to test them first.
- Design test cases to verify each interface in detail.
- Choose input data for test case execution. Input data plays a significant role in testing.
- Fix defects and retest.

# Test case Specification

***Test Case Specification*** document described detailed summary of ***what scenarios will be tested, how they will be tested, how often they will be tested***, and so on and so forth, for a given feature. It specifies the purpose of a specific test, identifies the required inputs and expected results, provides step-by-step procedures for executing the test, and outlines the pass/fail criteria for determining acceptance.
*Test Case Specification* has to be done separately for each unit. Based on the approach specified in the test plan, the feature to be tested for each unit must be determined. The overall approach stated in the plan is refined into specific test techniques that should be followed and into the criteria to be used for evaluation. Based on these the test cases are specified for the testing unit.
However, a ***Test Plan is a collection of all Test Specifications*** for a given area. The Test Plan contains a high-level overview of what is tested for the given feature area.

Reason for Test Case Specification:

There are two basic reasons test cases are specified before they are used for testing:

1. Testing has severe limitations and the effectiveness of testing depends heavily on the exact nature of the test case. Even for a given criterion the exact nature of the test cases affects the effectiveness of testing.
2. Constructing a good ***Test Case*** that will reveal errors in programs is a very creative activity and depends on the tester. It is important to ensure that the set of test cases used is of high quality. This is the primary reason for having the test case specification in the form of a document.
   The Test Case Specification is developed in the Development Phase by the organization responsible for the formal testing of the application.

## What is Test Case Specification Identifiers?

The way to uniquely identify a test case is as follows:

- **Test Case Objectives:** *Purpose of the test*
- **Test Items:** *Items (e.g., requirement specifications, design specifications, code, etc.) required to run a particular test case. This should be provided in "Notes" or "Attachment" feature. It describes the features and conditions required for testing.*
- **Input Specifications:** *Description of what is required (step-by-step) to execute the test case (e.g., input files, values that must be entered into a field, etc.). This should be provided in "Action" field.*
- **Output Specifications:** *Description of what the system should look like after the test case is run. This should be provided in the "Expected Results" field.*
- **Environmental Needs***: Description of any special environmental needs. This includes system architectures, Hardware & Software tools, records or files, interfaces, etc.*

To sum up, ***Test Case Specification*** defines the exact set up and inputs for one Test Case

### What is Test case?

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

### Typical Test Case Parameters:

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters

- Actual Result
- Environment Information
- Comments

### Example:

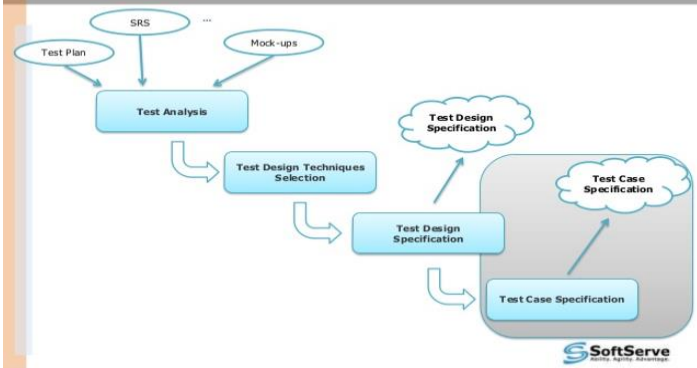
Let us say that we need to check an input field that can accept maximum of 10 characters.

While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while the second case is a FAIL.

| Scenario | Test Step | Expected Result | Actual Outcome |
|---|---|---|---|
| Verify that the input field that can accept maximum of 10 characters | Login to application and key in 10 characters | Application should be able to accept all 10 characters. | Application accepts all 10 characters. |
| Verify that the input field that can accept maximum of 11 characters | Login to application and key in 11 characters | Application should NOT accept all 11 characters. | Application accepts all 10 characters. |

If the expected result doesn't match with the actual result, then we log a defect. The defect goes through the defect life cycle and the testers address the same after fix.

## Trainings' Content



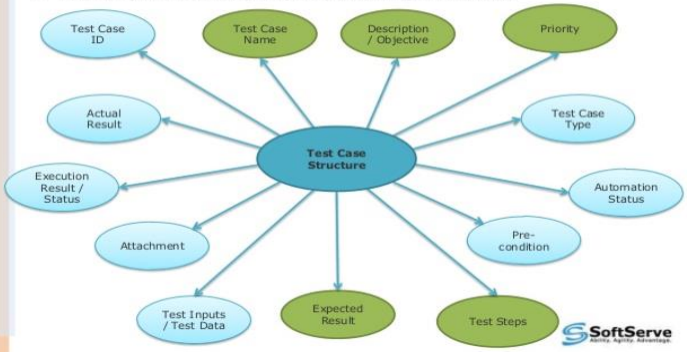

## What is Test Case?

- **Test case is:**
  - a set of input values, execution preconditions, expected results and execution, post conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement
  - documentation specifying inputs, predicted results, and a set of execution conditions for a test item

*according to Standard Glossary of Software Engineering (IEEE 610)*

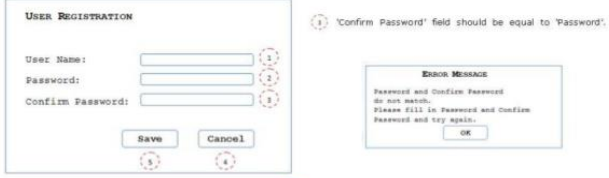
## Test Case Structure

- Depending on the particular project Test Case might consists of:




## Test Case Example

Requirements




Test Design

| Requirement | Test Name | Description |
|---|---|---|
| 'Confirm Password' field validation | Error dialog on saving user account with unequal password and confirm password | This test verifies that error dialog appears while save action if: 1.'Confirm Password' field is blank 2.password and confirm password do not match |

Test Case

## Test Case Example

| Test Case ID: 2 | Test Case Name: Error dialog on saving user account with unequal password and confirm password | Status: Pass |
|---|---|---|
| Test Type: Functional | Author: <First and Last Names> | Creation Date: 10/17/2010 |
| Automation: Recommended | Priority: Medium | Disposition: Reviewed - Completed |

**Objective:**
This test case verifies that error dialog appears while save action if 1)'Confirm Password' field is blank; 2) password and confirm password do not match.

**Pre-Conditions and Setup:**
Administrator user is logged to the system and 'User Registration' page is opened. If not, then run:
#1 Test Function: Open 'User Registration' page as Administrator user

The user "TestUserA" does not exist in the system.

| # | Test Steps | Expected Result |
|---|---|---|
| 1 | Set 1# value to 'User Name' field | |
| 2 | Set 2# value to 'Password' field | |
| 3 | Set 3# value to 'Confirm Password' field | |
| 4 | Click 'Save' button on the page | Error message appears in dialog window: "Password and Confirm Password do not match. Please fill in Password and Confirm Password and try again." |
| 5 | Click 'OK' button on the dialog window | |

**Test Data:**
1# "TestUserA", "TestUserA"
2# "Password1", "Password1"
3# <Blank>, "password2"

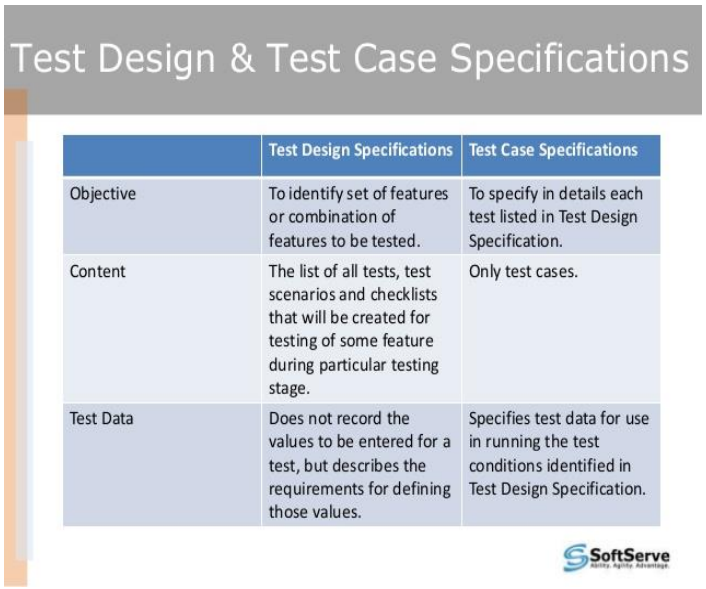
**Attachment(s):**
None

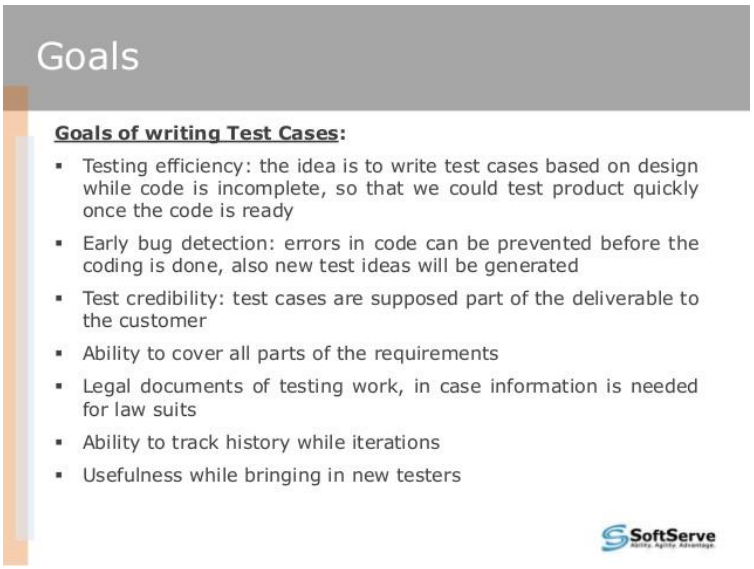
## Test Case Specification

- **Test Case Specification** – a document specifying a set of test cases (objective, inputs, test actions, expected results, and execution preconditions) for a test item (After IEEE-829)

- **According to IEEE 829 Test Case Specification consists of:**

1. Test case specification identifier
2. Test items
3. Input and Output specifications
4. Environmental needs
5. Special procedural requirements
6. Inter-case dependencies




# Reliability Assessment

**RELIABILITY TESTING** is a software testing type, that checks whether the software can perform a failure-free operation for a specified period of time in a particular environment.

Reliability means "yielding the same," in other terms, the word "reliable" means something is dependable and that it will give the same outcome every time. The same is true for Reliability testing. Reliability testing in software assures that the product is fault free and is reliable for its intended purpose.

## Reliability Testing Example

The probability that a PC in a store is up and running for eight hours without crashing is 99%; this is referred as reliability.

Reliability Testing can be categorized into three segments,

- Modeling
- Measurement
- Improvement

The following formula is for calculating the probability of failure.

```
Probability = Number of failing cases/ Total number of cases under consideration
```

## Factors Influencing Software Reliability

1. The number of faults presents in the software
2. The way users operate the system

- Reliability Testing is one of the key to better software quality. This testing helps discover many problems in the software design and functionality.
- The main purpose of reliability testing is to check whether the software meets the requirement of customer's reliability.
- Reliability testing will be performed at several levels. Complex systems will be tested at unit,assembly,subsystem and system levels.

## Why to do Reliability Testing

Reliability testing is done to test the software performance under the given conditions.

The objective behind performing reliability testing are,

1. To find the structure of repeating failures.
2. To find the number of failures occurring is the specified amount of time.
3. To discover the main cause of failure
4. To conduct Performance Testing of various modules of software application after fixing defect

After the release of the product too,we can minimize the possibility of occurrence of defects and thereby improve the software reliability. Some of the tools useful for this are- Trend Analysis,Orthogonal Defect Classification and formal methods, etc..

## Types of reliability Testing

Software reliability testing includes Feature Testing, Load Testing and Regression Testing

**Feature Testing:-**

Featured Testing check the feature provided by the software and is conducted in the following steps:-

- Each operation in the software is executed at least once.
- Interaction between the two operations is reduced.
- Each operation have to be checked for its proper execution.

**Load Testing:-**

Usually, the software will perform better at the beginning of the process and after that, it will start degrading. Load Testing is conducted to check the performance of the software under maximum work load.

**Regression Test:-**

Regression testing is mainly used to check whether any new bugs have been introduced because of the fixing of previous bugs. Regression Testing is conducted after every change or updation of the software features and their functionalities.

# How to do Reliability Testing

Reliability Testing is costly compared to other types of testing. So Proper planning and management is required while doing reliability testing. This includes testing process to be implemented, data for test environment, test schedule, test points, etc.

To begin with reliability testing, tester has to keep following things,

- Establish reliability goals
- Develop operational profile
- Plan and execute tests
- Use test results to drive decisions

As we discussed earlier, there are three categories in which we can perform the Reliability Testing,-**Modeling, Measurement and Improvement**.

The key parameters involved in Reliability Testing are:-

- Probability of failure-free operation
- Length of time of failure-free operation
- The environment in which it is executed

## Step 1) Modeling

Software Modeling Technique can be divided into two subcategories:

**1. Prediction Modeling**

**2. Estimation Modeling**

- Meaningful results can be obtained by applying suitable models.
- Assumptions and abstractions can be made to simplify the problems and no single model will suitable for all the situations.

The major differences of two models are:-

| Issues | Prediction Models | Estimation Models |
|---|---|---|
| **Data Reference** | It uses historical data | It uses current data from the software development. |
| **When used in Development Cycle** | It will be usually created before the development or testing phases. | It will be usually used at the later stage of Software Development Life Cycle. |
| **Time Frame** | It will predict the reliability in the future. | It will predict the reliability either for the present time or in the future time. |

## Step 2) Measurement

Software reliability cannot be measured directly and hence, other related factors are considered in order to estimate the software reliability. The current practices of Software Reliability Measurement are divided into four categories:-

1. **Product Metrics:-**

Product metrics are the combination of 4 types of metrics:

- **Software size**: - Line of Code (LOC) is an intuitive initial approach for measuring the size of the software. Only the source code is counted in this metric, and the comments and other non-executable statements will not be counted.
- **Function point Metric**:- Function Pont Metric is the method for measuring the functionality of the Software Development. It will consider the count of inputs, outputs, master files, etc. It measures the functionality delivered to the user and is independent of the programming language.
- **Complexity**:- It is directly related to software reliability, so representing complexity is important. Complexity-oriented metric is a method of determining the complexity of a program's control structure, by simplifying the code into a graphical representation.

- **Test Coverage Metrics**:- It is a way of estimating fault and reliability by performing the complete test of software products. Software reliability means it is the function of determining that the system has been completely verified and tested.

2. **Project Management Metrics**

- Researchers have realized that good management can result in the better products.
- A good management can achieve higher reliability by using better development process,risk management process,configuration management process, etc.

3. **Process Metrics**

The quality of the product is directly related to the process. The process metrics can be used to estimate, monitor and improve the reliability and quality of software.

4. **Fault and Failure Metrics**

Fault and Failure Metrics are mainly used to check whether the system is completely failure-free. Both the types of faults found out during the testing process (i.e. before delivery) as well as the failure reported by users after delivery are collected, summarized and analyzed to achieve this goal.

Software reliability is measured in terms of **mean time between failures (MTBF)**. MTBF consists of

- Mean to failure (MTTF): It is the difference of time between two consecutive failures
- Mean time to repair (MTTR): It is the time required to fix the failure.

```
MTBF = MTTF + MTTR
```

Reliability for good software is a number between **0 and 1.**

Reliability increases when errors or bugs from the program are removed.

## Step 3) Improvement

Improvement completely depends upon the problems occurred in the application or system, or else the characteristics of the software. According to the complexity of the software module,the way of improvement will also differ. Two main constraints time and budget, which will limit the efforts are put into the software reliability improvement.

# Example Methods for Reliability Testing

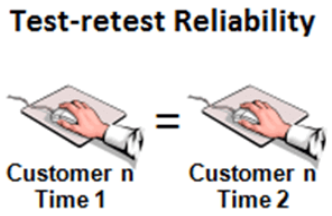
Testing for reliability is about exercising an application so that failures are discovered and removed before the system is deployed.

There are mainly three approaches used for Reliability Testing

- Test-Retest Reliability
- Parallel Forms Reliability
- Decision Consistency

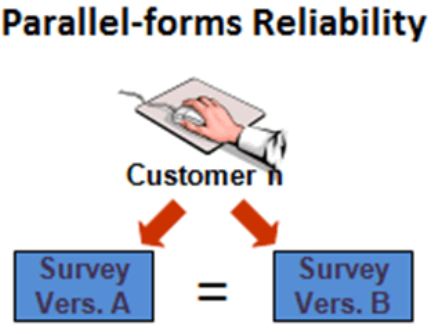
Below we tried to explain all these with an example.

## Test-Retest Reliability




To estimate test-retest reliability, a single group of examinees will perform testing process only a few days or weeks apart. The time should be short enough so that the examinees skills in the area can be assessed. The relationship between the examinee's scores from two different administrations is estimated, through statistical correlation. This type of reliability demonstrates the extent to which a test is able to produce stable, consistent scores across time.

## Parallel Forms Reliability




Many exams have multiple formats of question papers, this parallel forms of exam provide Security. Parallel forms reliability is estimated by administrating both forms of the exam to the same group of examinees. The examinees scores on the two test forms are correlated in order to

determine how similarly the two test forms functions. This reliability estimate is a measure of how consistent examinees scores can be expected to across test forms.

## Decision Consistency

After doing Test-Retest Reliability and Parallel Form Reliability, we will get a result of examinees either pass or fail. It is the reliability of this classification decision that is estimated in decision consistency reliability.

**Importance of Reliability Testing**

A thorough assessment of reliability is required to improve the performance of software product and process. Testing software reliability will help the software managers and practitioners to a great extent.

To check the reliability of the software via testing:-

1. A large number of test cases should be executed for an extended period of time to find out how long the software will execute without failure.
2. The test cases distribution should match the actual or planned operational profile of the software. The more often a function of the software is executed, the greater the percentage of test cases that should be allocated to that function or subset.

## Reliability Testing Tools

Some of the Tools used for Software Reliability are

1. WEIBULL++:- Reliability Life Data Analysis

2. RGA:- Reliability Growth Analysis

3. RCM:-Reliability Centered Maintenance

# Validation & Verification Metrics

Verification and Validation is the process of investigating that a software system satisfies specifications and standards and it fulfills the required purpose. **Barry Boehm** described verification and validation as the following:

> **Verification:** Are we building the product right?
> **Validation:** Are we building the right product?

**Verification:**
Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have.
Verification is **Static Testing**.

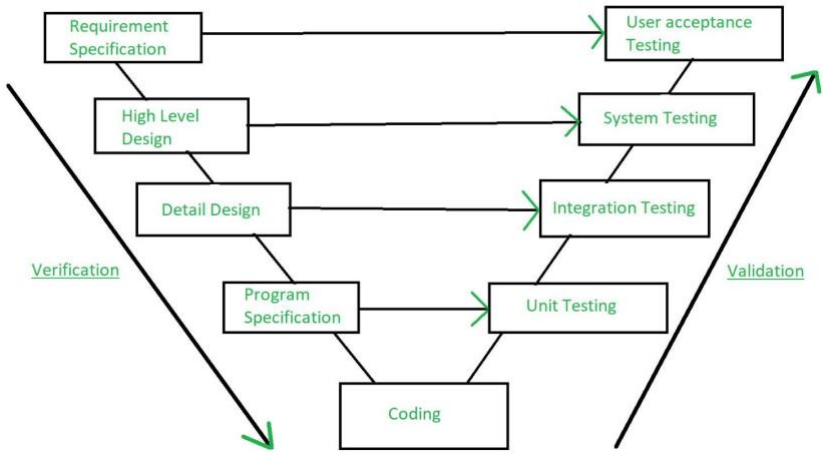
Activities involved in verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

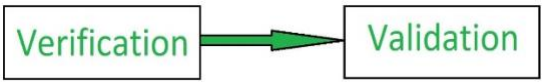
**Validation:**
Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product.
Validation is the **Dynamic Testing**.

Activities involved in validation:

1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing




**Note:** Verification is followed by Validation.



# Difference between verification and validation

| Verification | Validation |
| --- | --- |
| • The verifying process includes checking documents, design, code, and program | • It is a dynamic mechanism of testing and validating the actual product |
| • It does ***not*** involve executing the code | • It always involves executing the code |
| • Verification uses methods like reviews, walkthroughs, inspections, and desk-checking etc. | • It uses methods like Black Box Testing, White Box Testing, and non-functional testing |
| • Whether the software conforms to specification is checked | • It checks whether the software meets the requirements and expectations of a customer |
| • It finds bugs early in the development cycle | • It can find bugs that the verification process can not catch |
| • Target is application and software architecture, specification, complete design, high level, and database design etc. | • Target is an actual product |
| • QA team does verification and make sure that the software is as per the requirement in the SRS document. | • With the involvement of testing team validation is executed on software code. |
| • It comes before validation | • It comes after verification |

# Test Monitoring and Test Control

Test Monitoring and Test Control is basically a management activity. Test Monitoring is a process of and evaluating providing feedback on the "currently in progress" testing phase. Test Control is an activity of guiding and taking corrective action based on some metrics or information to improve efficiency and quality.

**Test Monitoring activity includes:**

1. Providing feedback to the team and the other concerned stakeholders about the progress of the testing efforts.
2. Broadcasting the results of testing performed, to the associated members.
3. Finding and tracking the Test Metrics.
4. Planning and Estimation, for deciding the future course of action, based on the metrics calculated.

Point 1 and 2 basically talk about Test Reporting, which is an important part of Test Monitoring. Reports should be precise and should avoid "long stories". It is important here to understand that the content of the report differs for every stakeholder.

**Points 3 and 4 talks about the metrics. Following metrics can be used for Test Monitoring:**

1. Test Coverage Metric
2. Test Execution Metrics (Number of test cases pass, fail, blocked, on hold)
3. Defect Metrics
4. Requirement Traceability Metrics
5. Miscellaneous metrics like level of confidence of testers, date milestones, cost, schedule and turnaround time.

**Test Control involves guiding and taking corrective measures activity, based on the results of Test Monitoring. Test Control examples include:**

1. Prioritizing the Testing efforts
2. Revisiting the Test schedules and Dates
3. Reorganizing the Test environment
4. Re prioritizing the Test cases/Conditions

Test Monitoring and Control go hand in hand. Being primarily a manager's activity, a Test Analyst contributes towards this activity by gathering and calculating the metrics which will be eventually used for monitoring and control.

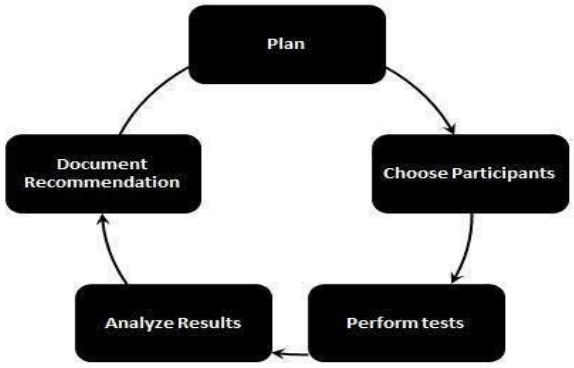
# Usability Testing

Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

- Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.
- The measure of increase in user productivity if any.
- Assessment of a user's attitude towards using the software.

It is also called User Experience(UX) Testing. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

### Usability Testing Process:




## Why do Usability Testing

Aesthetics and design are important. How well a product looks usually determines how well it works.

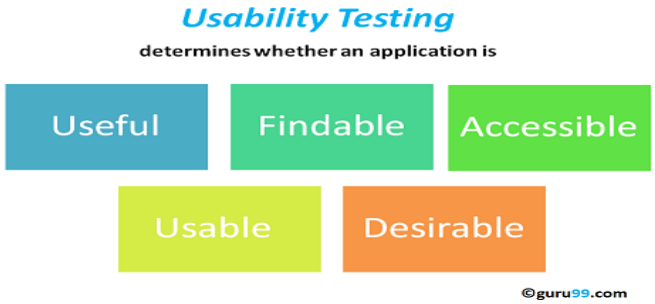
There are many software applications/websites, which miserably fail, once launched, due to following reasons -

- Where do I click next?
- Which page needs to be navigated?
- Which Icon or Jargon represents what?
- Error messages are not consistent or effectively displayed

- Session time not sufficient.

Software Engineering, Usability Testing identifies usability errors in the system early in the development cycle and can save a product from failure.

## Example Usability Testing Test Cases




The goal of this testing is to satisfy users and it mainly concentrates on the following parameters of a system:

The effectiveness of the system

- Is the system is easy to learn?
- Is the system useful and adds value to the target audience?
- Are Content, Color, Icons, Images used are aesthetically pleasing?

Efficiency

- Little navigation should be required to reach the desired screen or webpage, and scrollbars should be used infrequently.
- Uniformity in the format of screen/pages in your application/website.
- Option to search within your software application or website.

Accuracy

- No outdated or incorrect data like contact information/address should be present.
- No broken links should be present.

User Friendliness

- Controls used should be self-explanatory and must not require training to operate
- Help should be provided for the users to understand the application/website
- Alignment with the above goals helps in effective usability testing

# How to do Usability Testing: Complete Process

Usability testing process consists of the following phases


Planning:- During this phase the goals of usability test are determined. Having volunteers sit in front of your application and recording their actions is not a goal. You need to determine critical functionalities and objectives of the system. You need to assign tasks to your testers, which exercise these critical functionalities. During this phase, the usability testing method, number & demographics of usability testers, test report formats are also determined

Recruiting: During this phase, you recruit the desired number of testers as per your usability test plan. Finding testers who match your demographic (age, sex etc.) and professional ( education, job etc.) profile can take time.

Usability Testing: During this phase, usability tests are actually executed.

Data Analysis: Data from usability tests is thoroughly analyzed to derive meaningful inferences and give actionable recommendations to improve the overall usability of your product.

Reporting: Findings of the usability test is shared with all concerned stakeholders which can include designer, developer, client, and CEO

# Methods of Usability Testing: 2 Techniques

There are two methods available to do usability testing -

- Laboratory Usability Testing
- Remote Usability Testing

Laboratory Usability Testing:. This testing is conducted in a separate lab room in presence of the observers. The testers are assigned tasks to execute. The role of the observer is to monitor the behavior of the testers and report the outcome of testing. The observer remains silent during the course of testing. In this testing, both observers and testers are present in a same physical location.

Remote Usability Testing: Under this testing observers and testers are remotely located. Testers access the System Under Test, remotely and perform assigned tasks. Tester's voice , screen activity , testers facial expressions are recorded by an automated software. Observers analyze this data and report findings of the test. Example of such a software - http://silverbackapp.com/

## How many users do you need ?

Research (Virzi, 1992 and Neilsen Landauer, 1993) indicates that 5 users are enough to uncover 80% of usability problems. Some researchers suggest other numbers.

The truth is , the actual number of the user required depends on the complexity of the given application and your usability goals. Increase in usability participants results into increased cost , planning , participant management and data analysis.

But as a general guideline, if you on a small budget and interested in DIY usability testing 5 is a good number to start with. If budget is not a constraint its best consult experienced professionals to determine the number of users.

## UX Testing Checklist

The primary goal of this testing is to find crucial usability problems before the product is launched. Following things have to be considered to make a testing success:

- Start the UX testing during the early stage of design and development
- It's a good practice to conduct usability testing on your competitor's product before you begin development. This will help you determine usability standards for your target audience
- Select the appropriate users to test the system(Can be experts/non-experts users/50-50 of Experts and Non-Experts users)
- Use a bandwidth shaper . For instance , your target audience has poor network connectivity , limit network bandwidth to say 56 Kbps for your usability testers.
- Testers need to concentrate on critical & frequently used functionalities of the system.
- Assign a single observer to each tester. This helps observer to accurately note tester's behavior. If an observer is assigned to multiple testers, results may be compromised
- Educate Designers and Developers that this testing outcomes is not a sign of failure but it's a sign of Improvement

## Advantages

- It helps uncover usability issues before the product is marketed.
- It helps improve end-user satisfaction
- It makes your system highly effective and efficient
- It helps gather true feedback from your target audience who actually use your system during a usability test. You do not need to rely on "opinions" from random people.

## Disadvantages

- Cost is a major consideration in usability testing. It takes lots of resources to set up a Usability Test Lab. Recruiting and management of usability testers can also be expensive

However, these costs pay themselves up in form of higher customer satisfaction, retention and repeat business. Usability testing is therefore highly recommended.

# User Interface Testing

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].

### GUI Testing - Characteristics:

- GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
- During execution, the values of the properties of each objects of a GUI define the GUI state.
- It has capabilities to exercise GUI events like key press/mouse click.
- Able to provide inputs to the GUI Objects.
- To check the GUI representations to see if they are consistent with the expected ones.
- It strongly depends on the used technology.

### GUI Testing - Approaches:

- **Manual Based -** Based on the domain and application knowledge of the tester.
- **Capture and Replay -** Based on capture and replay of user actions.
- **Model-based testing -** Based on the execution of user sessions based on a GUI model. Various GUI models are briefly discussed below.

### Model Based Testing - In Brief:

- **Event-based model -** Based on all events of the GUI need to be executed at least once.
- **State-based model -** "all states" of the GUI are to be exercised at least once.
- **Domain model -** Based on the application domain and its functionality.

### GUI Testing Checklist:

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

## GUI Automation Tools

Following are some of the open source GUI automation tools in the market:

| Product | Licensed Under | URL |
|---|---|---|
| AutoHotkey | GPL | http://www.autohotkey.com/ |
| Selenium | Apache | http://docs.seleniumhq.org/ |
| Sikuli | MIT | http://sikuli.org |
| Robot Framework | Apache | www.robotframework.org |
| watir | BSD | http://www.watir.com/ |
| Dojo Toolkit | BSD | http://dojotoolkit.org/ |

Following are some of the Commercial GUI automation tools in the market.

| Product | Vendor | URL |
|---|---|---|
| AutoIT | AutoIT | http://www.autoitscript.com/site/autoit/ |
| EggPlant | TestPlant | www.testplant.com |
| QTP | Hp | http://www8.hp.com/us/en/software-solutions/ |
| Rational Functional Tester | IBM | http://www-03.ibm.com/software/products/us/en/functional |
| Infragistics | Infragistics | www.infragistics.com |
| iMacros | iOpus | http://www.iopus.com/iMacros/ |

| | | |
|---|---|---|
| CodedUI | Microsoft | http://www.microsoft.com/visualstudio/ |
| Sikuli | Micro Focus International | http://www.microfocus.com/ |

## Need of GUI Testing

Now the basic concept of GUI testing is clear. The few questions that will strike in your mind will be

- Why do GUI testing?
- Is it really needed?
- Does testing of functionally and logic of Application is not more than enough?? Then why to waste time on UI testing.

To get the answer to think as a user, not as a tester. A user doesn't have any knowledge about XYZ software/Application. It is the UI of the Application which decides that a user is going to use the Application further or not.

A normal User first observes the design and looks of the Application/Software and how easy it is for him to understand the UI. If a user is not comfortable with the Interface or find Application complex to understand he would never going to use that Application Again. That's why, GUI is a matter for concern, and proper testing should be carried out in order to make sure that GUI is free of Bugs.

## What do you Check-in GUI Testing?

The following checklist will ensure detailed GUI Testing in Software Testing.

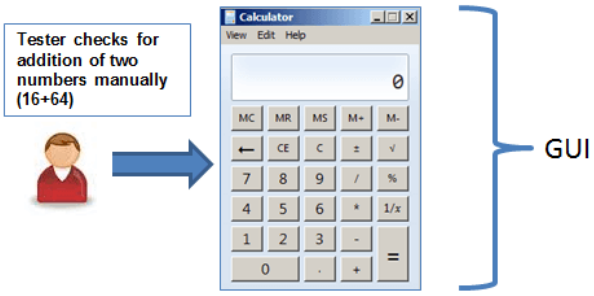
- Check all the GUI elements for size, position, width, length, and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in an application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

# GUI Testing Techniques / Methods

GUI testing can be done in three ways:

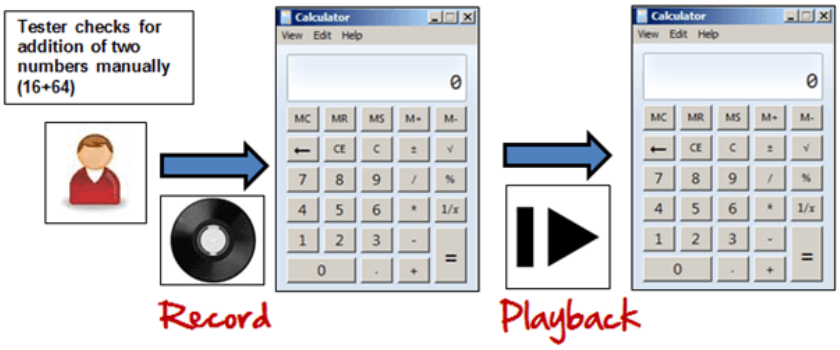
## Manual Based Testing

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in the business requirements document.

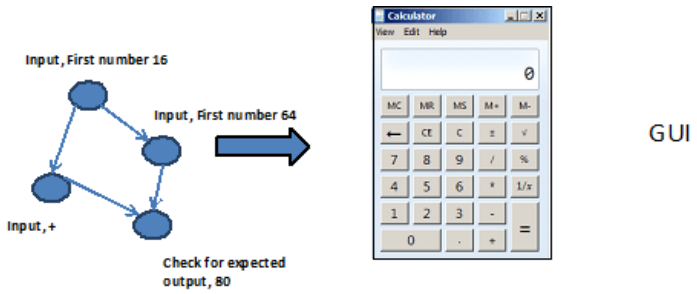



## Record and Replay

GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured by the automation tool. During playback, the recorded test steps are executed on the Application Under Test. Example of such tools - QTP.




## Model Based Testing




Model Based Testing

A model is a graphical description of a system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements. The following needs to be considered for this model based testing:

- Build the model
- Determine Inputs for the model
- Calculate the expected output for the model
- Run the tests
- Compare the actual output with the expected output
- A decision on further action on the model

Some of the modeling techniques from which test cases can be derived:

- Charts - Depicts the state of a system and checks the state after some input.
- Decision Tables - Tables used to determine results for each input applied

Model based testing is an evolving technique for generating test cases from the requirements. Its main advantage, compared to above two methods, is that it can determine undesirable states that your GUI can attain.

## Example GUI Testing Test Cases

GUI Testing basically involves

- Testing the size, position, width, height of the elements.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen.
- Testing of the font whether it is readable or not.
- Testing of the screen in different resolutions with the help of zooming in and zooming out like 640 x 480, 600x800, etc.
- Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
- Testing the colors of the fonts.
- Testing the colors of the error messages, warning messages.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.

- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.

# Follows Standards and Guidelines

## Nielsen and Molich's 10 User Interface Design Guidelines

Jakob Nielsen, a renowned web usability consultant and partner in the Nielsen Norman Group, and Rolf Molich, another prominent usability expert, established a list of ten user interface design guidelines in the 1990s. Note that there is considerable overlap between Nielsen and Molich's heuristics and Ben Shneiderman's 'eight golden rules'. These 10 rules of thumb further iterate upon Shneiderman's eight golden rules 4 years after Shneiderman's initial publication.

- **Visibility of system status**. Users should always be informed of system operations with easy to understand and highly visible status displayed on the screen within a reasonable amount of time.
- **Match between system and the real world**. Designers should endeavor to mirror the language and concepts users would find in the real world based on who their target users are. Presenting information in logical order and piggybacking on user's expectations derived from their real-world experiences will reduce cognitive strain and make systems easier to use.
- **User control and freedom**. Offer users a digital space where backward steps are possible, including undoing and redoing previous actions.
- **Consistency and standards**. Interface designers should ensure that both the graphic elements and terminology are maintained across similar platforms. For example, an icon that represents one category or concept should not represent a different concept when used on a different screen.
- **Error prevention**. Whenever possible, design systems so that potential errors are kept to a minimum. Users do not like being called upon to detect and remedy problems, which may on occasion be beyond their level of expertise. Eliminating or flagging actions that may result in errors are two possible means of achieving error prevention.
- **Recognition rather than recall.** Minimize cognitive load by maintaining task-relevant information within the display while users explore the interface. Human attention is limited and we are only capable of maintaining around five items in our short-term memory at one time. Due to the limitations of short-term memory, designers should ensure users can simply employ recognition instead of recalling information across parts of the dialogue. Recognizing something is always easier than recall because recognition involves perceiving cues that help us reach into our vast memory and allowing relevant information to surface. For example, we often find the format of multiple choice questions easier than short answer questions on a test because it only requires us to recognize the answer rather than recall it from our memory.
- **Flexibility and efficiency of use**. With increased use comes the demand for less interactions that allow faster navigation. This can be achieved by using abbreviations, function keys, hidden commands and macro facilities. Users should be able to customize or tailor the interface to suit their needs so that frequent actions can be achieved through more convenient means.
- **Aesthetic and minimalist design**. Keep clutter to a minimum. All unnecessary information competes for the user's limited attentional resources, which could inhibit user's memory retrieval of relevant information. Therefore, the display must be reduced to only the necessary components for the current tasks, whilst providing clearly visible and unambiguous means of navigating to other content.

- **Help users recognize, diagnose and recover from errors**. Designers should assume users are unable to understand technical terminology, therefore, error messages should almost always be expressed in plain language to ensure nothing gets lost in translation.
- **Help and documentation**. Ideally, we want users to navigate the system without having to resort to documentation. However, depending on the type of solution, documentation may be necessary. When users require help, ensure it is easily located, specific to the task at hand and worded in a way that will guide them through the necessary steps towards a solution to the issue they are facing.

Google Inc., the multibillion-dollar technology company, certainly produces designs that reflect the above heuristics. Jon Wiley, the head designer of Google Search in 2012 once said:

“When I think of design and creating great user experiences, I generally think of it in terms of three things: usability, utility and desirability.”

Nielsen and Molich’s 10 user interface guidelines cover these three key components of user experience quite nicely, which means you can likely improve the user experience of your designs by following these guidelines!

# 📄 Accessibility Testing

Accessibility Testing is defined as a type of Software Testing performed to ensure that the application being tested is usable by people with disabilities like hearing, color blindness, old age and other disadvantaged groups. It is a subset of Usability Testing.

People with disabilities use assistive technology which helps them in operating a software product. Examples of such software are:

- **Speech RecognitionSoftware -** It will convert the spoken word to text , which serves as input to the computer.
- **Screen reader software** - Used to read out the text that is displayed on the screen
- **Screen Magnification Software**- Used to enlarge the monitor and make reading easy for vision-impaired users.
- **Special keyboard** made for the users for easy typing who have motor control difficulties

## Why Accessibility Testing?

**Reason 1**: Cater to market for Disabled People.

About 20% of the population has disability issues.

- 1 in 10 people have a sever disability

- 1 in 2 people over 65 have reduced capabilities

Disabilities include blindness, deaf, handicapped, or any disorders in the body.

A software product can cater to this big market, if it's made disabled friendly. Accessibility issues in software can be resolved if Accessibility Testing is made part of normal software testing life cycle.

**Reason 2**: Abide by Accessibility Legislations

Government agencies all over the world have come out with legalizations, which requires that IT products to be accessible by disabled people.

Following are the legal acts by various governments -

- United States: Americans with Disabilities Act - 1990
- United Kingdom: Disability Discrimination Act - 1995
- Australia: Disability Discrimination Act - 1992
- Ireland : Disability Act of 2005

Accessibility Testing is important to ensure legal compliance.

**Reason 3**: Avoid Potential Law Suits

In the past,Fortune 500 companies have been sued because their products were not disabled friendly. Here a few prominent cases

- National Federation for the Blind (NFB) vs Amazon (2007)
- Sexton and NFB vs Target (2007)
- NFB Vs AOL settlement (1999)

It's best to create products which support disabled and avoid potential lawsuits.

## Which Disabilities to Support?

Application must support people with disabilities like -

| Type of Disability | Disability Description |
|---|---|
| **Vision Disability** | • Complete Blindness or Color Blindness or Poor Vision |

| | |
|---|---|
| | • Visual problems like visual strobe and flashing effect problems |
| **Physical Disability** | • Not able to use the mouse or keyboard with one hand.<br>• Poor motor skills like hand movements and muscle slowness |
| **Cognitive disability** | • Learning Difficulties or Poor Memory or not able to understand more complex scenarios |
| **Literacy Disability** | • Reading Problems |
| **Hearing Disability** | • Auditory problems like deafness and hearing impairments<br>• Cannot able to hear or not able to hear clearly |

## How to do Accessibility Testing?

Accessibility Testing can be performed in 2 ways, and they are:

1. Manual
2. Automated

Following are the point's needs to be checked for application to be used by all users. This checklist is used for signing off accessibility testing.

1. Whether an application provides keyboard equivalents for all mouse operations and windows?
2. Whether instructions are provided as a part of user documentation or manual? Is it easy to understand and operate the application using the documentation?
3. Whether tabs are ordered logically to ensure smooth navigation?
4. Whether shortcut keys are provided for menus?
5. Whether application supports all operating systems?
6. Whether response time of each screen or page is clearly mentioned so that End Users know how long to wait?
7. Whether all labels are written correctly in the application?
8. Whether color of the application is flexible for all users?

9. Whether images or icons are used appropriately, so it's easily understood by the end users?
10. Whether an application has audio alerts?
11. Whether a user is able to adjust audio or video controls?
12. Whether a user can override default fonts for printing and text displays?
13. Whether user can adjust or disable flashing, rotating or moving displays?
14. Check to ensure that color-coding is never used as the only means of conveying information or indicating an action
15. Whether highlighting is viewable with inverted colors? Testing of color in the application by changing the contrast ratio
16. Whether audio and video related content are properly heard by the disability people ? Test all multimedia pages with no speakers in websites
17. Whether training is provided for users with disabilities that will enable them to become familiar with the software or application?

Accessibility testing may be challenging for testers because they are unfamiliar with disabilities. It is better to work with disabled people who have specific needs to understand their challenges.

There are different way of Testing the Accessibility depending upon the Disability. We will learn all them one by one.

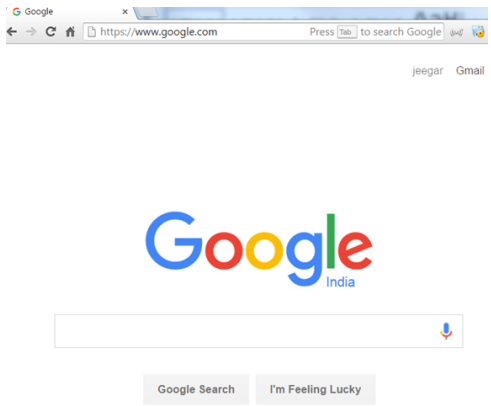
## 1) Vision Disability

OK now let us assume I don't have vision ability. I am completely blind, and I wanted to access XYZ Website. In that case, what is the option???? Cannot I access the XYZ website? What the option do I have? There is one-word option which is termed as **SCREENREADER.** Yeah, you got it right. SCREENREADER. Now, what is this Screen reader? It is a Software which is used for narrating the content on the web. Basically, what is on your website whether it is content, Link, Radio Button, Images, Video, etc. A screen reader will narrate each and everything for me. There are numerous Screen Reader available. I have worked with jaws.

Basically, when you start jaws or any screen reader and then go to the website, then it will narrate you the complete content. For Ex: I have started jaws, and started the browser JAWS will announce that Mozilla Firefox starts page, now if I go to address bar then JAWS will announce that **ADDRESS BAR** and then type **www.google.com** on address bar, jaws will going to explain somewhat like this:-

```
Address Bar,w,w,w,period,g,o,o,g,l,e,period,c,o,m. Also, when the page loads completely jaws will
again announce Google.Com Home page.
```

Now if I go to Google Search, then JAWS will announce that Google search. So it would be easy for a blind person to recognize things in an easy manner.




The point I want to explain here a screen reader will narrate word by word if you enter something or in the text box**.** Similarly, if there is link it will pronounce it as a link, for Button it will pronounce it as a button. So that a Blind person can easily Identify things.

Now If a website is poorly designed and developed, then it might be possible (it generally happens) that jaws would not be able to narrate correct content which in turn result for inaccessibility for Blind Person.(Say if jaws are narrating a link as a content, then a blind user would never able to know that it's a link and if that would be a crucial one for that website then ????).In that case, it would be a result into a high loss for Website Business.

## 2) Visual Impairment

There are two categories which I want to be mentioned under visual impairment.

The first one is Color Blindness. Color Blindness means not completely blind but not able to view some specific color properly. Red and Blue are the common colors which people not able to see properly if they do have color blindness. So basically, if I do have a color blindness of red color and I want to use website which is 80% in red then???Would I be comfortable on that website? The answer is No.

So a website should be designed such that a person with color blindness does not have any problem to access that. Take a simple example of a button which is in Red. To make it accessible if it is outlined with Black. Then it is easy to access. Normally Black and white are considered as universal.

## 3) POOR VISION DISABILITY

The second thing is a person having poor vision (not clear vision) or having different eyesight problem (there are many eye problem related to the retina, etc.) for accessing any site.

1) In such cases, the best thing to do is avoid small text. Because it would be a great advantage for poorly vision people.

2) Also, people with vision problem would like to zoom text of website to make it comfortable for them. So a website should be designed in such a manner that if enlarging it, its layout is not breakable when zooming the text. Otherwise, it won't be a good impression for them.

### 4) Other Disability

In Accessibility Testing for Disabled audience one very major point to consider is Accessing the Website without the use of the mouse. A person should be able to complete access the website the links, buttons, radio buttons, checkboxes, pop-ups, dropdown, all the controls should be completely accessible and operable through the keyboard.

**For Example**: If I am right handed paralyzed, and I am not comfortable with a mouse or say I don't want to use a mouse then what? In that case, if I am not able to access link or checkboxes on site via keyboard then???? So a website should be completely accessible with Keyboard.

```
Alternative Text should be there for Images, Audio, Video so that screen reader reads them and wil
l narrate them so that a blind person can easily recognize what the  image, audio, the video is al
l about. In addition, to it, keyboard shortcuts should be there to easily access website and navig
ation should be available with the keyboard.
```

Also, the focus should be completely visible. When we are pressing tab, then the user should be able to see where the control is moving. With visible focus, it becomes very easy for a user having poor vision or color blindness to identify the flow of a site and also an ease of access.

**User with Hearing Disability (Deaf or hard to listen):** The last ones are a person having a disability of Hearing. A deaf person can access the website as he is what able to see the content on the website. But when it comes to audio and video they face difficulties. So in that case, for any Video and Audio, there should be Alt text. Alt text means Alternative text. Suppose there is any Video about how to Book an airline ticket. In that case, the text should be there so that a deaf person can read that and get the idea what the video is all about.

## Accessibility Testing Tools:

To make your website more acceptable and user-friendly, it is crucial that it is easily accessible. There are various tools which can check the accessibility of the website. Some of these popular tools are listed below-

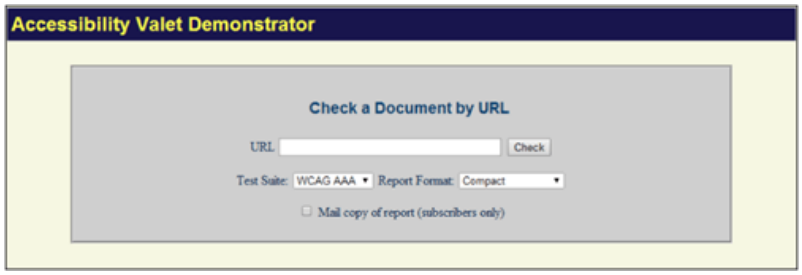
## 1) Wave

Wave is a free web accessibility tool created by WEBAIM. It is used to validate the web page manually for various aspects of accessibility. This tool can be used to check the intranet, password protected, dynamically generated, or sensitive web pages. Major functions of Web Accessibility Toolbar includes identifying components of a webpage, providing access to alternative view of page content and facilitating the use of third party online applications. It ensures 100% private and secure accessibility reporting

## 2) TAW

TAW is the online tool for determining accessibility of your web. This tool analyzes the web site in accordance with W3C web accessibility guidelines and shows accessibilities issues. Web accessibility test issues are categorized into priority 1, priority 2 and priority 3. The interesting feature of TAW is the ability to generate subsets of WCAG 1.0 to test against. In TAW tool, you can either choose to test a single page or multiple pages by "spider" a site. TAW also enable us to define additional checks via the "User Checking's" dialog box

## 3) Accessibility Valet




It is a tool that allows you to check web pages against WCAG (Web Content Accessibility Guidelines) compliance. All the HTML reporting options show your markup in a normalized form highlighting deprecated, bogus and valid mark as well as elements that are misplaced. This tool offers various features like

- In-dept reports for developers
- Executive summary for QA and Management
- Meta-data for the semantic web and WWW
- Automatic cleanup and Html to XHtml conversion
- Scripting tools

### 4) Accessibility Developer Tools

It is a Chrome extension. It does and accessibility audit. The results of the audit show accessibility rules that are violated by the Page Under Test. The extension has high reviews and is frequently updated

### 5) Quick Accessibility Page Tester

Since there are some excellent accessibility toolbars, Quick Page Accessibility Tester is a bookmark that you can click to get a quick analysis of the web page. It will figure out various issues with your page, warns about possible issues and highlight areas on the page which might benefit from ARIA (Accessible Rich Internet Applications).

There are various tools available in the market to perform web accessibility testing given below:

### 6) aDesigner

This is a tool developed by IBM which simulates the experience of visually impaired individuals so that the designer can better understand the needs of disabled people and develop applications accordingly.

### 7) WebAnywhere

This is a browser based tool that works similarly to the screen readers like Jaws. It assists the readers how to read the web page.

### 8) Web accessibility toolbar

WAT is an extension of Internet explorer or Opera that offers web page designers with useful features in the analysis of web page. One best feature is GreyScale feature which helps to find low contrast spots in the design.

## Myths of Accessibility Testing:

Following are the Myths of Accessibility Testing:

**Myth:** Creating Accessible website is expensive

**Fact:** It is not expensive. Take the timeout to think about accessibility issues in the design stage itself along with basic testing .This will save money as well as rework.

**Myth:** Changing inaccessible websites to accessible website is time consuming and expensive

It is not necessary to incorporate all the changes at one time. Work on basic needs which are most necessary for disabled users.

**Myth:** Accessibility is plain and boring

**Accessibility doesn't mean text only page**

You can make web pages attractive, but it should be designed in such a way that it can be accessible by all users. Also as per W3C web content accessibility guidelines - it strongly discourage the use of text only pages.

**Myth:** Accessibility if for the Blind and Disabled

**Fact** Following accessibility guidelines improves the overall usability of the software, which helps regular users as well.

# Website Testing

**WEB TESTING**, or website testing is checking your web application or website for potential bugs before its made live and is accessible to general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked.

## How to test Web Application

In Software Engineering, the following testing types/technique may be performed depending on your web testing requirements.

## 1. Functionality Testing:

This is used to check if your product is as per the specifications you intended for it as well as the functional requirements you charted out for it in your developmental documentation. Web based Testing Activities includes:

**Test** all **links** in your webpages are working correctly and make sure there are no broken links. Links to be checked will include -

- Outgoing links
- Internal links
- Anchor Links
- MailTo Links

**Test Forms** are working as expected. This will include-

- Scripting checks on the form are working as expected. For example- if a user does not fill a mandatory field in a form an error message is shown.
- Check default values are being populated
- Once submitted, the data in the forms is submitted to a live database or is linked to a working email address
- Forms are optimally formatted for better readability

**Test Cookies** are working as expected. Cookies are small files used by websites to primarily remember active user sessions so you do not need to log in every time you visit a website. Cookie Testing will include

- Testing cookies (sessions) are deleted either when cache is cleared or when they reach their expiry.

- Delete cookies (sessions) and test that login credentials are asked for when you next visit the site.

**Test HTML and CSS** to ensure that search engines can crawl your site easily. This will include

- Checking for Syntax Errors
- Readable Color Schemas
- Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

**Test business workflow**- This will include

- Testing your end - to - end workflow/ business scenarios which takes the user through a series of webpages to complete.
- Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

**Tools that can be used**: **QTP** , IBM Rational , **Selenium**

## 2. Usability testing:

Usability Testing has now become a vital part of any web based project. It can be **carried out by testers** like you **or a small focus group** similar to the target audience of the web application.

**Test** the site **Navigation**:

- Menus, buttons or Links to different pages on your site should be easily visible and consistent on all webpages

**Test** the **Content**:

- Content should be legible with no spelling or grammatical errors.
- Images if present should contain an "alt" text

**Tools that can be used**: Chalkmark, Clicktale, Clixpy and Feedback Army

## 3.Interface Testing:

Three areas to be tested here are - Application, Web and Database Server

- **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
- **Web Server**: Test Web server is handling all application requests without any service denial.
- **Database Server:** Make sure queries sent to the database give expected results.

**Test system response** when **connection between the three layers** (Application, Web and Database) **cannot be established** and appropriate message is shown to the end user.

**Tools that can be used**: AlertFox, Ranorex

## 4. Database Testing:

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include-

- Test if any errors are shown while executing queries
- Data Integrity is maintained while creating, updating or deleting data in database.
- Check response time of queries and fine tune them if necessary.
- Test data retrieved from your database is shown accurately in your web application

**Tools that can be used**: **QTP**, **Selenium**

## 5. Compatibility testing.

Compatibility tests ensures that your web application displays correctly across different devices. This would include-

**Browser Compatibility Test**: Same website in different browsers will display differently. You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for Mobile Browser Compatibility.

The rendering of web elements like buttons, text fields etc. changes with change in **Operating System**. Make sure your website works fine for various combination of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox, Internet Explorer, Safari etc.

**Tools that can be used**: NetMechanic

## 6. Performance Testing:

This will ensure your site works under all loads. Software Testing activities will include but not limited to -

- Website application response times at different connection speeds
- Load test your web application to determine its behavior under normal and peak loads
- Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
- Test if a crash occurs due to peak load, how does the site recover from such an event
- Make sure optimization techniques like gzip compression, browser and server side cache enabled to reduce load times

**Tools that can be used**: **Loadrunner**, **JMeter**

## 7. Security testing:

Security Testing is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include-

- Test unauthorized access to secure pages should not be permitted
- Restricted files should not be downloadable without appropriate access
- Check sessions are automatically killed after prolonged user inactivity
- On use of SSL certificates, website should re-direct to encrypted SSL pages.

**Tools that can be used**: Babel Enterprise, BFBTester and CROSS

## 8. Crowd Testing:

You will select a large number of people (crowd) to execute tests which otherwise would have been executed a select group of people in the company. Crowdsourced testing is an interesting and upcoming concept and helps unravel many a unnoticed defects.

**Tools that can be used**: People like you and me !!!. And yes , loads of them!

This concludes that  It includes almost all testing types applicable to your web application.

As a Web-tester its important to note that web testing is quite an arduous process and you are bound to come across many obstacles. One of the major problems you will face is of course **deadline pressure**. Everything is always needed yesterday! The number of times the **code will need changing** is also taxing. Make sure you **plan your work** and know clearly

what is expected of you. Its best **define all the tasks** involved in your web testing and then **create a work chart for accurate estimates and planning**.

# Web page Fundamentals

Websites that are not well designed tend to perform poorly and have sub-optimal Google Analytics metrics (e.g. high bounce rates, low time on site, low pages per visit and low conversions). So what makes good web design? Below we explore the top 10 web design principles that will make your website aesthetically pleasing, easy to use, engaging, and effective.

1. **Purpose**

Good web design always caters to the needs of the user. Are your web visitors looking for information, entertainment, some type of interaction, or to transact with your business? Each page of your website needs to have a clear purpose, and to fulfill a specific need for your website users in the most effective way possible.

**2. Communication**

People on the web tend to want information quickly, so it is important to communicate clearly, and make your information easy to read and digest. Some effective tactics to include in your web design include: organising information using headlines and sub headlines, using bullet points instead of long windy sentences, and cutting the waffle.

**3. Typefaces**

In general, Sans Serif fonts such as Arial and Verdana are easier to read online (Sans Serif fonts are contemporary looking fonts without decorative finishes). The ideal font size for reading easily online is 16px and stick to a maximum of 3 typefaces in a maximum of 3 point sizes to keep your design streamlined.

**4. Colours**

A well thought out colour palette can go a long way to enhance the user experience. Complementary colours create balance and harmony. Using contrasting colours for the text and background will make reading easier on the eye. Vibrant colours create emotion and should be used sparingly (e.g. for buttons and call to actions). Last but not least, white space/ negative space is very effective at giving your website a modern and uncluttered look.

**5. Images**

A picture can speak a thousand words, and choosing the right images for your website can help with brand positioning and connecting with your target audience. If you don’t have high quality professional photos on hand, consider purchasing stock photos to lift the look of your website. Also consider using infographics, videos and graphics as these can be much more effective at communicating than even the most well written piece of text.

### 6. Navigation

Navigation is about how easy it is for people to take action and move around your website. Some tactics for effective navigation include a logical page hierarchy, using bread crumbs, designing clickable buttons, and following the 'three click rule' which means users will be able to find the information they are looking for within three clicks.

### 7. Grid based layouts

Placing content randomly on your web page can end up with a haphazard appearance that is messy. Grid based layouts arrange content into sections, columns and boxes that line up and feel balanced, which leads to a better looking website design.

### 8. "F" Pattern design

Eye tracking studies have identified that people scan computer screens in an "F" pattern. Most of what people see is in the top and left of the screen and the right side of the screen is rarely seen. Rather than trying to force the viewer's visual flow, effectively designed websites will work with a reader's natural behaviour and display information in order of importance (left to right, and top to bottom).

### 9. Load time

Everybody hates a website that takes ages to load.  Tips to make page load times more effective include optimising image sizes (size and scale), combining code into a central CSS or JavaScript file (this reduces HTTP requests) and minify HTML, CSS, JavaScript (compressed to speed up their load time).

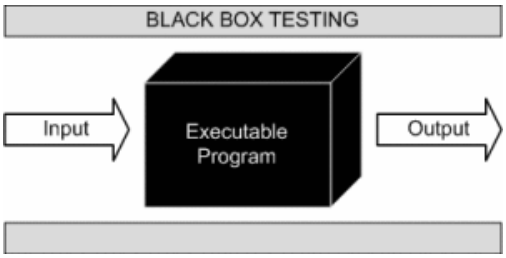
### 10: Mobile friendly

It is now commonplace to access websites from multiple devices with multiple screen sizes, so it is important to consider if your website is mobile friendly. If your website is not mobile friendly, you can either rebuild it in a responsive layout (this means your website will adjust to different screen widths) or you can build a dedicated mobile site (a separate website optimised specifically for mobile users).

It is easy to create a beautiful and functional website, simply by keeping these design elements in mind. Have you got a website design that needs reviewing or optimising? Or perhaps, you are planning a website and you are looking to get the design right from the ground up. Either way, these principles of effective web design can help your website be more engaging, useful, and memorable for visitors.

# Black Box Testing

**BLACK BOX TESTING**, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.




This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

Definition by ISTQB

- **black box testing:** Testing, either functional or non-functional, without reference to the internal structure of the component or system.
- **black box test design technique:** Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

### Example

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

## How to do BlackBox Testing

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

## Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones -

- **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** - Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

## Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

- For Functional/ Regression Tests you can use - QTP, Selenium
- For Non-Functional Tests, you can use - LoadRunner, Jmeter

## Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

Or

Black box testing can be done in following ways:

**1. Syntax Driven Testing –** This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers,language that can be represented by context free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

**2. Equivalence partitioning –** It is often seen that many type of inputs work similarly so instead of giving all of them separately we can group them together and test only one input of each group. The idea is to partition the input domain of the system into a number of equivalence classes such

that each member of class works in a similar way, i.e., if a test case in one class results in some error, other members of class would also result into same error.

The technique involves two steps:

1. **Identification of equivalence class –** Partition any input domain into minimum two sets: **valid values** and **invalid values**. For example, if the valid range is 0 to 100 then select one valid input like 49 and one invalid like 104.
2. **Generating test cases –**

   (i) To each valid and invalid class of input assign unique identification number.
   (ii) Write test case covering all valid and invalid test case considering that no two invalid inputs mask each other.

   To calculate the square root of a number, the equivalence classes will be:
   **(a) Valid inputs:**

   - Whole number which is a perfect square- output will be an integer.
   - Whole number which is not a perfect square- output will be decimal number.
   - Positive decimals

   **(b) Invalid inputs:**

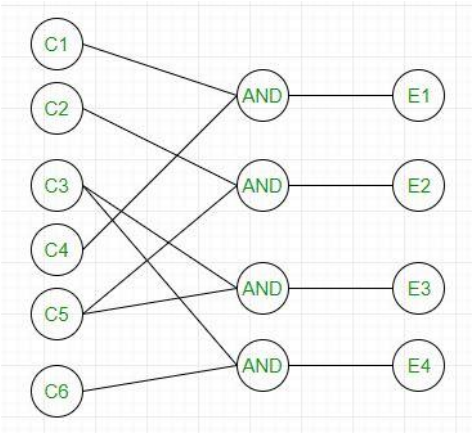
   - Negative numbers(integer or decimal).
   - Characters other that numbers like “a”,”!”,”;”,etc.

**3. Boundary value analysis –** Boundaries are very good places for errors to occur. Hence if test cases are designed for boundary values of input domain then the efficiency of testing improves and probability of finding errors also increase. For example – If valid range is 10 to 100 then test for 10,100 also apart from valid and invalid inputs.

**4. Cause effect Graphing –** This technique establishes relationship between logical input called causes with corresponding actions called effect. The causes and effects are represented using Boolean graphs. The following steps are followed:

1. Identify inputs (causes) and outputs (effect).
2. Develop cause effect graph.
3. Transform the graph into decision table.
4. Convert decision table rules to test cases.

For example, in the following cause effect graph:




It can be converted into decision table like:

| | | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| CAUSES | C1 | 1 | 0 | 0 | 0 |
| | C2 | 0 | 1 | 0 | 0 |
| | C3 | 0 | 0 | 1 | 1 |
| | C4 | 1 | 0 | 0 | 0 |
| | C5 | 0 | 1 | 1 | 0 |
| | C6 | 0 | 0 | 0 | 1 |
| EFFECTS | E1 | x | - | - | - |
| | E2 | - | x | - | - |
| | E3 | - | - | x | - |
| | E4 | - | - | - | x |

Each column corresponds to a rule which will become a test case for testing. So there will be 4 test cases.

**5. Requirement based testing –** It includes validating the requirements given in SRS of software system.

**6. Compatibility testing –** The test case result not only depend on product but also infrastructure for delivering functionality. When the infrastructure parameters are changed it is still expected to work properly. Some parameters that generally affect compatibility of software are:

1. Processor (Pentium 3,Pentium 4) and number of processors.
2. Architecture and characteristic of machine (32 bit or 64 bit).
3. Back-end components such as database servers.
4. Operating System (Windows, Linux, etc).

## Advantages

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.
- Tester need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.

- Test cases can be designed as soon as the specifications are complete.

### Disadvantages

- Only a small number of possible inputs can be tested and many program paths will be left untested.
- Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- Tests can be redundant if the software designer/developer has already run a test case.
- Ever wondered why a soothsayer closes the eyes when foretelling events? So is almost the case in Black Box Testing.

## White Box Testing

**WHITE BOX TESTING** (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

It is one of two parts of the **Box Testing** approach to software testing. Its counterpart, **Blackbox testing**, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

### Example

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

## Levels Applicable To

White Box Testing method is applicable to the following levels of software testing:

- Unit Testing: For testing paths within a unit.
- Integration Testing: For testing paths between units.
- System Testing: For testing paths between subsystems.

However, it is mainly applied to Unit Testing.

# What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

# How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include *Manual Testing*, trial, and error testing and the use of testing tools as we will explain further on in this article.

## WhiteBox Testing Example

Consider the following piece of code

```
Printme (int a, int b) {                        ------------  Printme is a function
    int result = a+ b;
    If (result> 0)
      Print ("Positive", result)
    Else
      Print ("Negative", result)
    }                                          -----------   End of the source code
```

The goal of WhiteBox testing is to verify all the decision branches, loops, statements in the code.

To exercise the statements in the above code, WhiteBox test cases would be

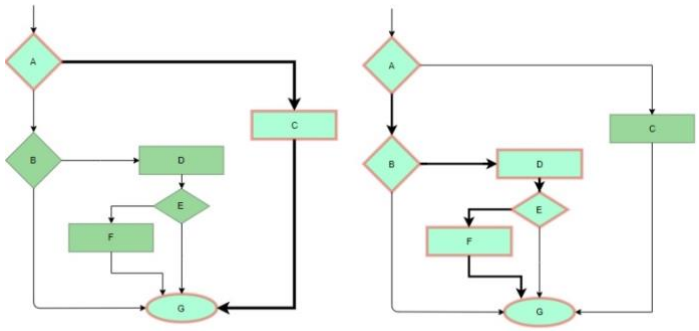
- A = 1, B = 1
- A = -1, B = -3

## White Box Testing Techniques

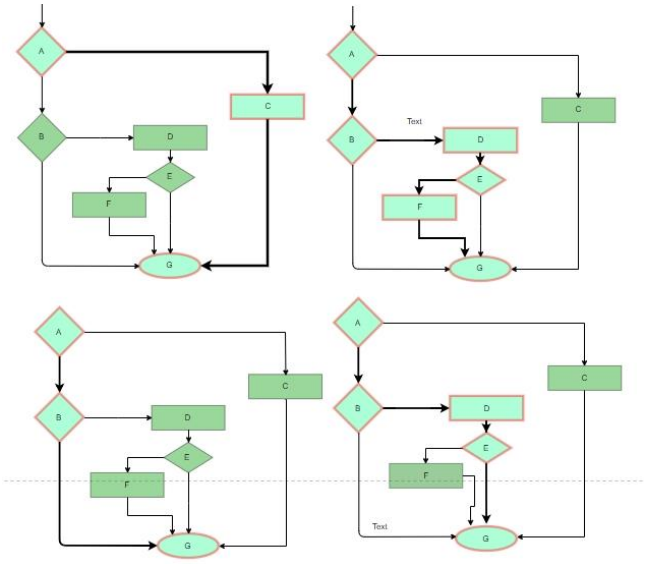
A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a *Test Case* suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product

There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques

**Statement Coverage**:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.




**Branch Coverage** - This technique checks every possible path (if-else and other conditional loops) of a software application.




Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.

## Types of White Box Testing

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below --

- *Unit Testing:* It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a

single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.

- *Testing for Memory Leaks*: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Apart from above, a few testing types are part of both black box and white box testing. They are listed as below

- White Box **Penetration Testing**: In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats
- White Box Mutation Testing: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

## White Box Testing Tools

Below is a list of top white box testing tools.

- Parasoft Jtest
- EclEmma
- NUnit
- PyUnit
- HTMLUnit
- CppUnit

### Advantages

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

### Disadvantages

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available.

# Gray Box Testing

GRAY BOX TESTING is a software testing method which is a combination of Black Box Testing method and White Box Testing method. In Black Box Testing, the internal structure of the item being tested is unknown to the tester and in White Box Testing the internal structure is known. In Gray Box Testing, the internal structure is partially known. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.

Gray Box Testing is named so because the software program, in the eyes of the tester is like a gray/semi-transparent box; inside which one can partially see.

Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known


In Software Engineering, Gray Box Testing gives the ability to test both sides of an application, presentation layer as well as the code part. It is primarily useful in Integration Testing and Penetration Testing.

## Example

An example of Gray Box Testing would be when the codes for two units/modules are studied (White Box Testing method) for designing test cases and actual tests are conducted using the exposed interfaces (Black Box Testing method).

## Levels Applicable To

Though Gray Box Testing method may be used in other levels of testing, it is primarily used in Integration Testing

# Why Gray Box Testing

Gray Box Testing is performed for the following reason,

- It provides combined benefits of both black box testing and white box testing both

- It combines the input of developers as well as testers and improves overall product quality
- It reduces the overhead of long process of testing functional and non-functional types
- It gives enough free time for a developer to fix defects
- Testing is done from the user point of view rather than a designer point of view

## Gray Box Testing Strategy

To perform Gray box testing, it is not necessary that the tester has the access to the source code. A test is designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

To perform Gray box Testing-

- It applies a straightforward technique of black box testing
- It is based on requirement test case generation, as such, it presets all the conditions before the program is tested by assertion method.

Techniques used for Grey box Testing are-

- *Matrix Testing:* This testing technique involves defining all the variables that exist in their programs.
- *Regression Testing*: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.
- *Orthogonal Array Testing or OAT*: It provides maximum code coverage with minimum test cases.
- *Pattern Testing:* This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Usually, Grey box methodology uses automated software testing tools to conduct the testing. Stubs and module drivers are created to relieve tester to manually generate the code.

Steps to perform Grey box Testing are:

- *Step 1*: Identify inputs
- *Step 2*: Identify the outputs
- *Step 3*: Identify the major paths
- *Step 4*: Identify Subfunctions
- *Step 5*: Develop inputs for Subfunctions

- *Step 6*: Develop outputs for Subfunctions
- *Step 7*: Execute test case for Subfunctions
- *Step 8*: Verify the correct result for Subfunctions
- *Step 9*: Repeat steps 4 & 8 for other Subfunctions
- *Step 10*: Repeat steps 7 & 8 for other Subfunctions

The test cases for grey box testing may include, GUI related, Security related, Database related, Browser related, Operational system related, etc.

## Gray Box Testing Challenges

- When a component under test encounter a failure of some kind may lead to abortion of the ongoing operation
- When test executes in full but the content of the result is incorrect

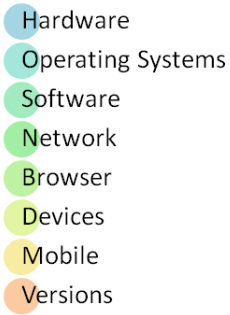
# Configuration & compatibility testing

**Compatibility** is nothing but the capability of existing or living together. In normal life, Oil is not compatible with water, but milk can be easily combined with water.

**Compatibility Testing** is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or Mobile devices.

Compatibility Testing is a type of Non-functional testing

## Types of Compatibility Tests

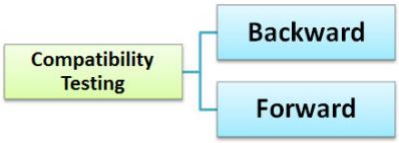



Let's look into compatibility testing types

- Hardware: It checks software to be compatible with different hardware configurations.
- Operating Systems: It checks your software to be compatible with different Operating Systems like Windows, Unix, Mac OS etc.

- Software: It checks your developed software to be compatible with other software. For example, MS Word application should be compatible with other software like MS Outlook, MS Excel, VBA etc.
- Network: Evaluation of performance of a system in a network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application in different networks with all parameters mentioned earlier.
- Browser: It checks the compatibility of your website with different browsers like Firefox, Google Chrome, Internet Explorer etc.
- Devices: It checks compatibility of your software with different devices like USB port Devices, Printers and Scanners, Other media devices and Blue tooth.
- Mobile: Checking your software is compatible with mobile platforms like Android, iOS etc.
- Versions of the software: It is verifying your software application to be compatible with different versions of the software. For instance checking your Microsoft Word to be compatible with Windows 7, Windows 7 SP1, Windows 7 SP2, Windows 7 SP3.

There are two types of version checking




Backward compatibility Testing is to verify the behavior of the developed hardware/software with the older versions of the hardware/software.

Forward compatibility Testing is to verify the behavior of the developed hardware/software with the newer versions of the hardware/software.

## Tools for Compatibility Testing

- BrowserStack - Browser Compatibility Testing: This tool helps a Software engineer to check application in different browsers.
- Virtual Desktops - Operating System Compatibility: This is used to run the applications in multiple operating systems as virtual machines. n Number of systems can be connected and compare the results.

## How to do Compatibility Testing

- The initial phase of compatibility testing is to define the set of environments or platforms the application is expected to work on.
- The tester should have enough knowledge of the platforms/software/hardware to understand the expected application behavior under different configurations.
- The environment needs to be set-up for testing with different platforms, devices, networks to check whether your application runs well under different configurations.
- Report the bugs. Fix the defects. Re-test to confirm Defect fixing.

## Configuration Testing

Configuration testing is defined as a software testing type, that checks an application with multiple combinations of software and hardware to find out the optimal configurations that the system can work without any flaws or bugs.

As discussed above, Configuration Testing is a software testing where the application under test has to be tested using multiple combinations of Software and Hardware.

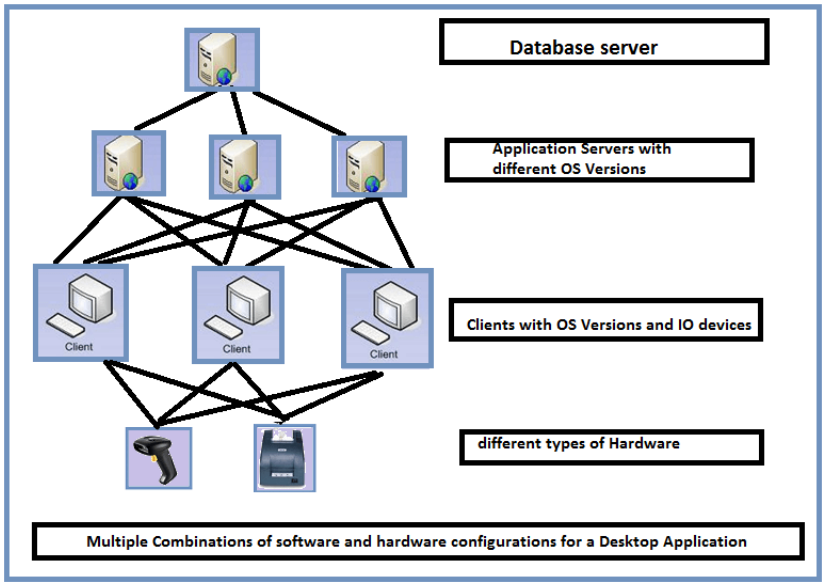
Let's understand this with an example of a Desktop Application:

Generally, Desktop applications will be of 2 tier or 3 tier, here we will consider a 3 tier Desktop application which is developed using Asp.Net and consists of Client, Business Logic Server and Database Server where each component supports below-mentioned platforms.

- Client Platform - Windows XP, Window7 OS, windows 8 OS , etc
- Server Platform - Windows Server 2008 R2,Windows Server 2008 R2, Windows Server 2012R2
- Database –SQL Sever 2008, SQL Server 2008R2, SQL Server 2012, etc.

A tester has to test the Combination of Client, Server and Database with combinations of the above-mentioned platforms and database versions to ensure that the application is functioning properly and does not fail.

Configuration testing is not only restricted to Software but also applicable for Hardware which is why it is also referred as a Hardware configuration testing, where we test different hardware devices like Printers, Scanners, Web cams, etc. that support the application under test.




## Pre-requisites for Configuration Testing

For any project before starting with the config test, we have to follow some pre-requisites

- Creation of matrix which consists of various combinations of software and hardware configurations
- Prioritizing the configurations as its difficult to test all the configurations
- Testing every configuration based on prioritization.

## Objectives of Configuration Testing

The objectives of configuration Testing is to

- Validating the application to determine if it fulfills the configurability requirements
- Manually causing failures which help in identifying the defects that are not efficiently found during testing (Ex: changing the regional settings of the system like Time Zone, Language, Date time formats, etc.)
- Determine an optimal configuration of the application under test.
- Analyzing the system performance by adding or modifying the hardware resources like Load Balancers, increase or decrease in memory size, connecting various printer models, etc.
- Analyzing system Efficiency based on the prioritization, how efficiently the tests were performed with the resources available to achieve the optimal system configuration.
- Verification of the system in a geographically distributed Environment to verify how effectively the system performs.

For Ex: Server at a different location and clients at a different location, the system should work fine irrespective of the system settings.

- Verifying how easily the bugs are reproducible irrespective of the configuration changes.
- Ensuring how traceable the application items are by properly documenting and maintaining the versions which are easily identifiable.
- Verifying how manageable the application items are throughout the software development life cycle.

## How to do Configuration Testing

In this section, we will discuss the strategy that needs to be followed for configuration testing types and there are two types of configuration testing as mentioned below

- Software Configuration Testing
- Hardware Configuration Testing

Software Configuration Testing

Software configuration testing is testing the Application under test with multiple OS, different software updates, etc. Software Configuration testing is very time consuming as it takes time to install and uninstall different software's that is used for the testing.

One of the approaches that is followed to test the software configuration is to test on Virtual Machines. Virtual Machine is an Environment that is installed on software and acts like a Physical Hardware and users will have the same feel as of a Physical Machine. Virtual Machines simulates real-time configurations.

Instead of Installing and uninstalling the software in multiple physical machines which is time-consuming, it's always better to install the application/software in the virtual machine and continue testing. This process can be performed by having multiple virtual machines, which simplifies the job of a tester

Software configuration testing can typically begin when

- Configurability requirements to be tested are specified
- Test Environment is ready
- Testing Team is well trained in configuration testing
- Build released is unit and Integration test passed

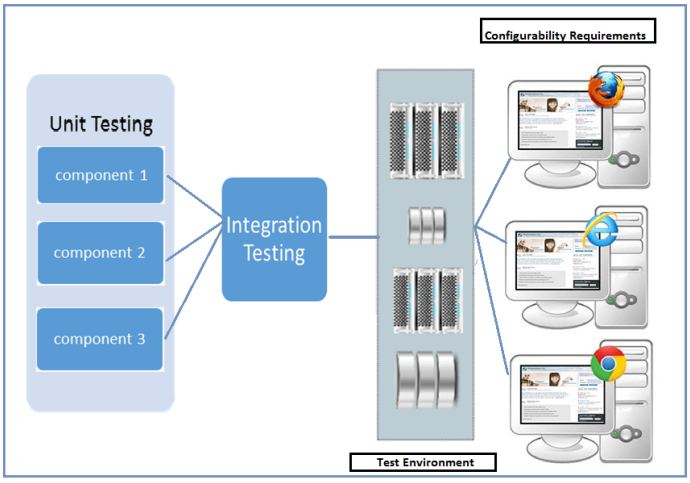
Typical Test Strategy that is followed to test the software configuration test is to run the functional test suite across multiple software configurations to verify if the application under test is working as desired without any flaws or errors.

Another strategy is to ensure the system is working fine by manually failing the test cases and verifying for the efficiency.

Example:

Say there is a Banking Application, which has to be tested for its compatibility across multiple browsers when the application is hosted in an environment where all the prerequisites are present it might pass the unit and Integration Testing in the test lab.

But if the same application is installed in a client place and the machines are missing some software's updates or the versions on which the application is dependent directly or indirectly there is a chance that the application might fail. To avoid this kind of situation, it's always suggested to fail the tests manually by removing some of the configurability requirements and then proceed with the testing.




Hardware Configuration Testing

Hardware configuration testing is generally performed in labs, where we find physical machines with different hardware attached to them.

Whenever a build is released, the software has to be installed in all the physical machines where the hardware is attached, and the test suite has to be run on each machine to ensure that the application is working fine.

To perform the above task a significant amount of effort is required to install the software on each machine, attach the hardware and manually running or even to automate the above said process and running the test suite.

Also, while performing hardware configuration test, we specify the type of hardware to be tested, and there are a lot of computer hardware and peripherals which make it quite impossible to run all of them. So it becomes the duty of the tester to analyze what hardware is mostly used by users and try to make the testing based on the prioritization.

## Sample Test Cases

Consider a Banking Scenario to test for the hardware compatibility. A Banking Application that is connected to Note Counting Machine has to be tested with different models like Rolex, Strob, Maxsell, StoK, etc.

Let's take some sample test cases to test the Note Counting Machine

- Verifying the connection of application with Rolex model when the prerequisites are NOT installed
- Verifying the connection of application with Rolex model when the prerequisites are installed
- Verify if the system is counting the notes correctly
- Verify if the system is counting the notes incorrectly
- Verifying the tampered notes
- Verifying the response times
- Verifying if the fake notes are detected and so on

The above test cases are for one model, and the same has to be tested with all the models available in the market by setting them up in a test lab which is difficult. Hence, it is advisable to outsource the hardware configuration testing to organizations which specializes them.

# 📄 Automation Testing

**AUTOMATION TESTING** means using an automation tool to execute your test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Test Automation demands considerable investments of money and resources.

Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool, it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation.

The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

Automated software testing is important due to the following reasons:

- Manual Testing of all workflows, all fields, all negative scenarios is time and money consuming
- It is difficult to test for multilingual sites manually
- Automation does not require Human intervention. You can run automated test unattended (overnight)
- Automation increases the speed of test execution
- Automation helps increase Test Coverage
- Manual Testing can become boring and hence error-prone.

Test cases to be automated can be selected using the following criterion to increase the automation ROI

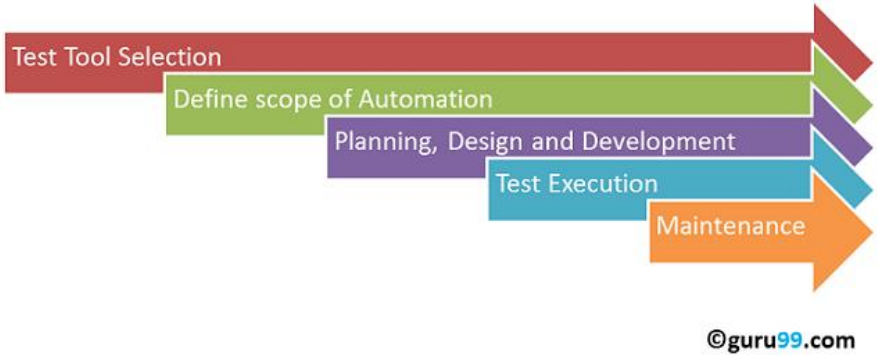
- High Risk - Business Critical test cases
- Test cases that are repeatedly executed
- Test Cases that are very tedious or difficult to perform manually
- Test Cases which are time-consuming

The following category of test cases are not suitable for automation:

- Test Cases that are newly designed and not executed manually at least once
- Test Cases for which the requirements are frequently changing
- Test cases which are executed on an ad-hoc basis.

## Automated Testing Process:

Following steps are followed in an Automation Process



## Test tool selection

Test Tool selection largely depends on the technology the Application Under Test is built on. For instance, QTP does not support Informatica. So QTP cannot be used for testing Informatica applications. **It's a good idea to conduct a Proof of Concept of Tool on AUT.**

## Define the scope of Automation

The scope of automation is the area of your Application Under Test which will be automated. Following points help determine scope:

- The features that are important for the business
- Scenarios which have **a large amount of data**
- **Common functionalities** across applications
- Technical feasibility
- The extent to which business components are reused
- **The complexity** of test cases
- Ability to use the same test cases for cross-browser testing

## Planning, Design, and Development

During this phase, you create an Automation strategy & plan, which contains the following details-

- Automation tools selected
- Framework design and its features
- In-Scope and Out-of-scope items of automation
- Automation testbed preparation
- Schedule and Timeline of scripting and execution
- Deliverables of Automation Testing

## Test Execution

Automation Scripts are executed during this phase. The scripts need input test data before there are set to run. Once executed they provide detailed test reports.

Execution can be performed using the automation tool directly or through the Test Management tool which will invoke the automation tool.

Example: Quality center is the Test Management tool which in turn it will invoke QTP for execution of automation scripts. Scripts can be executed in a single machine or a group of machines. The execution can be done during the night, to save time.

### Maintenance

As new functionalities are added to the System Under Test with successive cycles, Automation Scripts need to be added, reviewed and maintained for each release cycle. **Maintenance becomes necessary to improve the effectiveness of Automation Scripts.**

## Framework for Automation

A framework is set of automation guidelines which help in

- Maintaining consistency of Testing
- Improves test structuring
- Minimum usage of code
- Less Maintenance of code
- Improve re-usability
- Non Technical testers can be involved in code
- The training period of using the tool can be reduced
- Involves Data wherever appropriate

There are four types of frameworks used in automation software testing:

1. Data Driven Automation Framework
2. Keyword Driven Automation Framework
3. Modular Automation Framework
4. Hybrid Automation Framework

## Automation Tool Best Practices

To get maximum ROI of automation, observe the following

- The scope of Automation needs to be determined in detail before the start of the project. This sets expectations from Automation right.
- Select the right automation tool: A tool must not be selected based on its popularity, but it's fit to the automation requirements.
- Choose an appropriate framework

- Scripting Standards- Standards have to be followed while writing the scripts for Automation. Some of them are-
    - Create uniform scripts, comments, and indentation of the code
    - Adequate Exception handling - How error is handled on system failure or unexpected behavior of the application.
    - User-defined messages should be coded or standardized for Error Logging for testers to understand.
- Measure metrics- Success of automation cannot be determined by comparing the manual effort with the automation effort but by also capturing the following metrics.
    - Percent of defects found
    - The time required for automation testing for each and every release cycle
    - Minimal Time is taken for release
    - Customer Satisfaction Index
    - Productivity improvement

The above guidelines if observed can greatly help in making your automation successful.

## Benefits of Automation Testing

Following are benefits of automated testing:

- 70% faster than the manual testing
- Wider test coverage of application features
- Reliable in results
- Ensure Consistency
- Saves Time and Cost
- Improves accuracy
- Human Intervention is not required while execution
- Increases Efficiency
- Better speed in executing tests
- Re-usable test scripts
- Test Frequently and thoroughly
- More cycle of execution can be achieved through automation
- Early time to market

## Different types of software testing that can be automated

- Smoke Testing
- Unit Testing
- Integration Testing
- Functional Testing

- Keyword Testing
- Regression Testing
- Data Driven Testing
- Black Box Testing

## How to Choose an Automation Tool?

Selecting the right tool can be a tricky task. Following criterion will help you select the best tool for your requirement-

- Environment Support
- Ease of use
- Testing of Database
- Object identification
- Image Testing
- Error Recovery Testing
- Object Mapping
- Scripting Language Used
- Support for various types of test - including functional, test management, mobile, etc...
- Support for multiple testing frameworks
- Easy to debug the automation software scripts
- Ability to recognize objects in any environment
- Extensive test reports and results
- Minimize training cost of selected tools

Tool selection is one of biggest challenges to be tackled before going for automation. First, Identify the requirements, explore various tools and its capabilities, set the expectation from the tool and go for a Proof Of Concept.

## Automation Testing Tools

There are tons of Functional and Regression Testing Tools available in the market. Here are best tools certified by our experts

### 1. Ranorex Studio

Ranorex Studio is an all-in-one tool for automating functional UI tests, regression tests, data-driven tests and much more. Ranorex Studio includes an easy to use click-and-go interface to automate tests for web, desktop, and mobile applications.

**Features:**

- Functional UI and end-to-end testing on desktop, web, and mobile
- Cross-browser testing
- SAP, ERP, Delphi and legacy applications.
- iOS and Android
- Run tests locally or remotely, in parallel or distribute on a Selenium Grid
- Robust reporting

## 2. mabl

mabl delivers scriptless end-to-end test automation, integrated with your delivery pipeline, so you can focus on improving your app.

**Features:**

- Proprietary machine learning models automatically identify and surface application issues
- Tests are automatically repaired when UI changes
- Automated regression insights on every build

## 3. Selenium

It is a software testing tool used for Regression Testing. It is an open source testing tool that provides playback and recording facility for Regression Testing. The Selenium IDE only supports Mozilla Firefox web browser.

- It provides the provision to export recorded script in other languages like Java, Ruby, RSpec, Python, C#, etc
- It can be used with frameworks like JUnit and TestNG
- It can execute multiple tests at a time
- Autocomplete for Selenium commands that are common
- Walkthrough tests
- Identifies the element using id, name, X-path, etc.
- Store tests as Ruby Script, HTML, and any other format
- It provides an option to assert the title for every page
- It supports selenium user-extensions.js file
- It allows to insert comments in the middle of the script for better understanding and debugging

## 4. QTP (MicroFocus UFT)

QTP is widely used for functional and regression testing, it addresses every major software application and environment. To simplify test creation and maintenance, it uses the concept of keyword driven testing. It allows the tester to build test cases directly from the application.

- It is easier to use for a non-technical person to adapt to and create working test cases
- It fix defects faster by thoroughly documenting and replicating defects for developer
- Collapse test creation and test documentation at a single site
- Parameterization is easy than WinRunner
- QTP supports .NET development environment
- It has better object identification mechanism
- It can enhance existing QTP scripts without "Application Under Test" is available, by using the active screen

## 5. Rational Functional Tester

It is an Object-Oriented automated Functional Testing tool that is capable of performing automated functional, regression, data-driven testing and GUI testing. The main features of this tool are

- It supports a wide range of protocols and applications like Java, HTML, NET, Windows, SAP, Visual Basic, etc.
- It can record and replay the actions on demand
- It integrates well with source control management tools such as Rational Clear Case and Rational Team Concert integration
- It allows developers to create keyword associated script so that it can be re-used
- Eclipse Java Developer Toolkit editor facilitates the team to code test scripts in Java with Eclipse
- It supports custom controls through proxy SDK (Java/.Net)
- It supports version control to enable parallel development of test scripts and concurrent usage by geographically distributed team

## 6. WATIR

It is an open source testing software for regression testing. It enables you to write tests that are easy to read and maintain. Watir supports only internet explorer on windows while Watir webdriver supports Chrome, Firefox, IE, Opera, etc.

- It supports multiple browsers on different platforms
- Rather than using proprietary vendor script, it uses a fully-featured modern scripting language Ruby
- It supports your web app regardless of what it is developed in

## 7. SilkTest

Silk Test is designed for doing functional and regression testing. For e-business application, silk test is the leading functional testing product. It is a product of Segue Software takeover by Borland in 2006. It is an object-oriented language just like C++. It uses the concept of an object, classes, and inheritance. Its main feature includes

- It consists of all the source script files
- It converts the script commands into GUI commands. On the same machine, commands can be run on a remote or host machine
- To identify the movement of the mouse along with keystrokes, Silktest can be executed. It can avail both playback and record method or descriptive programming methods to get the dialogs
- It identifies all controls and windows of the application under test as objects and determines all of the attributes and properties of each window

# MODULE-4

# Software Project Management

A **project** is a group of tasks that need to complete to reach a clear result. A project also defines as a set of inputs and outputs which are required to achieve a goal. Projects can vary from simple to difficult and can be operated by one person or a hundred.

Projects usually described and approved by a project manager or team executive. They go beyond their expectations and objects, and it's up to the team to handle logistics and complete the project on time. For good project development, some teams split the project into specific tasks so they can manage responsibility and utilize team strengths.

**Software project management** is an art and discipline of planning and supervising software projects. It is a sub-discipline of software project management in which software projects planned, implemented, monitored and controlled.

It is a procedure of managing, allocating and timing resources to develop computer software that fulfills requirements.

In software Project Management, the client and the developers need to know the length, period and cost of the project.

### Need of Software Project Management:

Software is an non-physical product. Software development is a new stream in business and there is very little experience in building software products. Most of the software products are made to fit client's requirements. The most important is that the basic technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. Such type of business and environmental constraints increase risk in software development hence it is essential to manage software projects efficiently.

It is necessary for an organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. Hence in order, software project management is necessary to incorporate user requirements along with budget and time constraints.

## Prerequisite of software project management

There are three needs for software project management. These are:

- Time
- Cost
- Quality

It is an essential part of the software organization to deliver a quality product, keeping the cost within the clients budget and deliver the project as per schedule. There are various factors, both external and internal, which may impact this triple factor. Any of three-factor can severely affect the other two.

# Project Manager

A project manager is a character who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project. A project manager represents an essential role in the achievement of the projects.

A project manager is a character who is responsible for giving decisions, both large and small projects. The project manager is used to manage the risk and minimize uncertainty. Every decision the project manager makes must directly profit their project.

## Role of a Project Manager:

*1. Leader*

A project manager must lead his team and should provide them direction to make them understand what is expected from all of them.

*2. Medium:*

The Project manager is a medium between his clients and his team. He must coordinate and transfer all the appropriate information from the clients to his team and report to the senior management.

*3. Mentor:*

He should be there to guide his team at each step and make sure that the team has an attachment. He provides a recommendation to his team and points them in the right direction.

## Responsibilities of a Project Manager:

- Managing risks and issues.
- Create the project team and assigns tasks to several team members.
- Activity planning and sequencing.
- Monitoring and reporting progress.
- Modifies the project plan to deal with the situation.

Software Project Management consists of several different type of managements:

- Conflict Management:

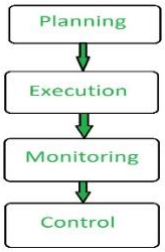
  Conflict management is the process to restrict the negative features of conflict while increasing the positive features of conflict. The goal of conflict management is to improve learning and

group results including efficacy or performance in an organizational setting. Properly managed conflict can enhance group results.

- Risk Management:
  Risk management is the analysis and identification of risks that is followed by synchronized and economical implementation of resources to minimize, operate and control the possibility or effect of unfortunate events or to maximize the realization of opportunities.
- Requirement Management:
  It is the process of analyzing, prioritizing, tracing and documenting on requirements and then supervising change and communicating to pertinent stakeholders. It is a continuous process during a project.
- Change Management:
  Change management is a systematic approach for dealing with the transition or transformation of an organization's goals, processes or technologies. The purpose of change management is to execute strategies for effecting change, controlling change and helping people to adapt to change.
- Software Configuration Management:
  Software configuration management is the process of controlling and tracing changes in the software, part of the larger cross-disciplinary field of configuration management. Software configuration management include revision control and the inauguration of baselines.

- Release Management:
  Release Management is the task of planning, controlling and scheduling the build in deploying releases. Release management ensures that organization delivers new and enhanced services required by the customer, while protecting the integrity of existing services.

Aspects of Software Project Management:




Advantages of Software Project Management:

- It helps in planning of software development.
- Implementation of software development is made easy.
- Monitoring and controlling are aspects of software project management.
- It overall manages to save time and cost for software development.

# Project Scheduling

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

- Identify all the functions required to complete the project.
- Break down large functions into small activities.
- Determine the dependency among various activities.
- Establish the most likely size for the time duration required to complete the activities.
- Allocate resources to activities.
- Plan the beginning and ending dates for different activities.
- Determine the critical path. A critical way is the group of activities that decide the duration of the project.

The first method in scheduling a software plan involves identifying all the functions required to complete the project. A good judgment of the intricacies of the project and the development process helps the supervisor to identify the critical role of the project effectively. Next, the large functions are broken down into a valid set of small activities which would be assigned to various engineers. The work breakdown structure formalism supports the manager to breakdown the function systematically after the project manager has broken down the purpose and constructs the work breakdown structure; he has to find the dependency among the activities. Dependency among the various activities determines the order in which the various events would be carried out.

If an activity A necessary the results of another activity B, then activity A must be scheduled after activity B. In general, the function dependencies describe a partial ordering among functions, i.e., each service may precede a subset of other functions, but some functions might not have any precedence ordering describe between them (called concurrent function). The dependency among the activities is defined in the pattern of an activity network.

Once the activity network representation has been processed out, resources are allocated to every activity. Resource allocation is usually done using a Gantt chart. After resource allocation is completed, a PERT chart representation is developed. The PERT chart representation is useful for program monitoring and control.

For task scheduling, the project plan needs to decompose the project functions into a set of activities. The time frame when every activity is to be performed is to be determined. The end of every action is called a milestone. The project manager tracks the function of a project by audit the timely completion of the milestones. If he examines that the milestones start getting delayed, then he has to handle the activities carefully so that the complete deadline can still be met.

Scheduling techniques help to align the timeline, the scope and your resources.

A **schedule** has to fit a specified timeframe and use available resources with the right skills. Given many uncertainties, variables and a possibility that resource availability or a project scope may change, it's hard to create a schedule that will last.

And after all, it is you who's going to be held accountable.

Here's how and when to use scheduling techniques applicable for IT projects, so you can prepare a reliable agenda.

### Why is scheduling important?

A schedule is your project's timetable, consisting of sequenced activities and milestones that have to be delivered under a given deadline.

Having a project plan you know exactly what should be delivered in what order. Resource allocation helps you find and assign the right employees. Then, a schedule tells you exactly when all of that should happen.

With the right scheduling techniques you can also adjust some activities and tasks in case of a project running late or if any changes to the scope occur.

### Scheduling techniques you should know

#### 1. Mathematical analysis

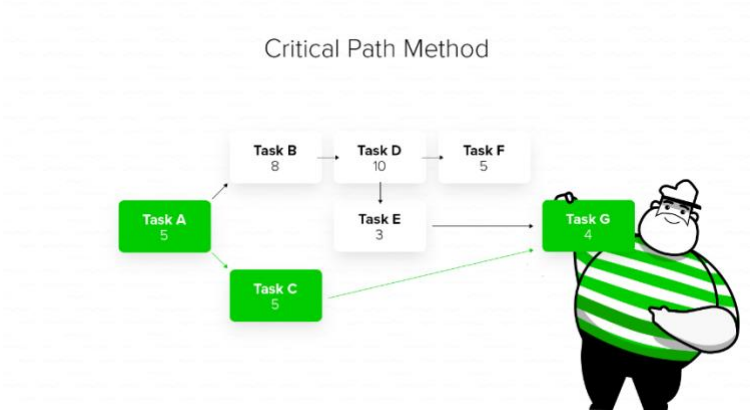
The first two techniques used by project managers are Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT). You can use these methods to calculate the assumed start and finish dates, based on the known scope of the project.

**Critical Path Method**

Let's take a look at the CPM, first. Think of the critical path as of your project's tree diagram. The Critical Path Method helps you uncover the longest possible timeline for the project, as well as the shortest one. With the CPM you'll also be able to mark critical tasks and the ones that may float. Changes in the latter's delivery won't affect the schedule.

To use the CPM, you need to know your project's scope and list all of the tasks necessary for its completion. Next, estimate how long each task may take. After that, you should also note all dependencies between the tasks. This way you know which ones can be done separately, and which ones require previous tasks to be completed. Lastly, add milestones and deliverables to your project.

A critical path helps to visualize the project flow and calculate its duration when all dependencies and deliverables are known. This technique may not be so handy if there are many uncertainties in the project.




**Program Evaluation and Review Technique**

Similar to a critical path, PERT is a way to visualize tasks' flow in the project and estimate the timeline based on their assumed duration. This technique also illustrates dependencies between tasks.

To schedule a project using PERT, just like in CPM you will need to define tasks and their order first, based on your project's milestones. Using a network framework similar to CPM,  consisting of these tasks, you can estimate different timelines for a project depending on the level of confidence:

- Optimistic timing
- Most-likely timing
- Pessimistic timing

Although it looks very similar to CPM, PERT uses weighted average duration rather than estimates to calculate possible timeframes.

A disadvantage of this approach is also a need to know the tasks and dependencies between them in order to fully benefit from this technique.

#### 2. Duration compression

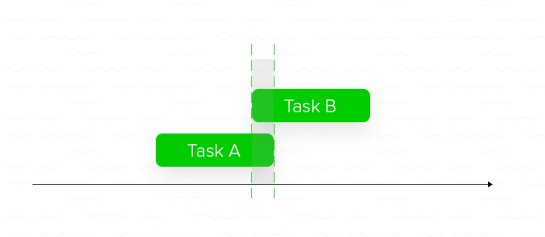
Duration compression is a way to shorten a schedule. It may be of use if the project is going late and you have to find a way to adjust a schedule without changing the scope of the project. There are two techniques that you may apply: fast tracking and crashing.

##### Fast tracking

As you already know what a critical path is, here's another way to use it. Fast tracking helps you to find tasks that could be done simultaneously or be partially overlapped to speed up the project's delivery. For that, take a look at the critical path to decide which activities could be fast-tracked.

Say you'd started a new project and have already gathered all requirements. At this point you could start with the design phase, and only start with software development if design is ready.

But if you need to fast track both processes, you can start software development after first, the most important designs are ready. Then, continue to prepare latter views while programmers implement materials they've already received.



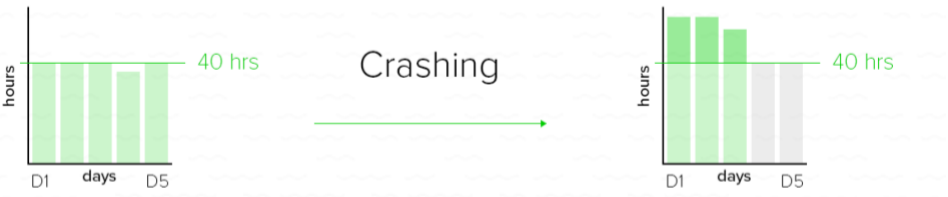

Although fast tracking may seem appealing, remember about the risks, too. As you will have to manage several important activities at once, it's easier to make costly mistakes or sacrifice quality.

**Crashing**

Another compression technique is crashing, which is about adding extra resources to finish the project on time. It is a tricky one though, as you need to have spare resources you can use.

Plus, not all tasks can be done faster by adding more team members. According to the Brook's Law, "adding more human resources to a late software project makes it later", which is connected to the general law of diminishing returns. The reason for that is additional communication that is needed to introduce new team members to a project and a limited divisibility of tasks.

Another way to use the crashing technique is to add time (e.g. paid overtime), but it has to still fit a deadline. The con of this approach is raising the cost of the project, though.



### Simulation

In the simulation technique you use a different set of activities to calculate possible durations of the project. It's especially handy when there are many uncertainties and variables. Using simulation you can create a schedule based on assumptions, so you can use even if you don't know all of the actions or if the scope may change.

One of the models you can use is a Monte Carlo Simulation. It takes different assumptions and possible outcomes into account, resulting in forecasted duration. The advantage of this approach is that it takes risks and uncertainties into account, so even if a scope changes or additional tasks occur, you can adjust your schedule.

### Resource-leveling heuristics

Resource leveling is adjusting a schedule and resources to cut the time of delivery or to *avoid under- or overutilization of resources*. You can use it to adjust a single activity in a project.

To adjust a schedule using resource leveling, divide or merge activities according to the resources' availability, so there are no under- or overutilized team members.


The use of this technique is vastly debated in project management community, as it may increase the project's cost and time. If you want to apply this technique, you should take the downsides of it into consideration.

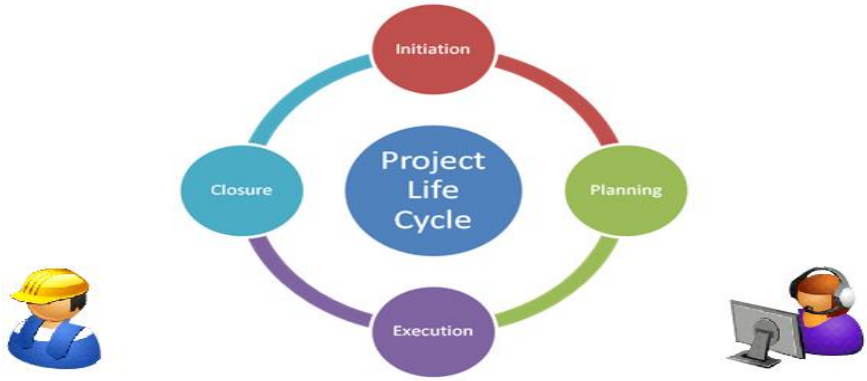
# Project Staffing

**Project staffing - Flexible Staffing Strategy / Solutions**

Companies face multiple challenges in this global economy. To remain competitive in this demanding market, they are constantly adopting latest cutting edge technologies so as to improve services and increase their bottom-line. This technology drive has created huge demand for highly talented and skilled resources for implementing mission critical projects. To improve the profits, these technology initiatives and related projects demand optimum resourcing / staffing strategies.

IT Managers face tough staffing issues with tight deadlines for deliverables. They have insufficient staff and most of them may not be able to take up the development using latest technologies without proper training. This calls for a staffing plan/strategy to support the flexible demands of the projects during the entire life cycle using outsourced services. ***The purpose of the staffing plan is to ensure that the project has sufficient resources with the right skills based on project schedule for successful completion.***

When developed as a strategy, flexible project staffing can cut cost, maintain quality and help accomplish specific goals. Better Minds Consulting has the expertise and capabilities to support mission critical project staffing needs. Our best practices approach starts off with an assessment of our client project needs, processes, and technologies. Our team, after careful evaluation, recommends a staffing plan to support the end-to end needs of the project implementation. Our staffing solutions offer flexibility, scalability, right mix and competitive pricing.



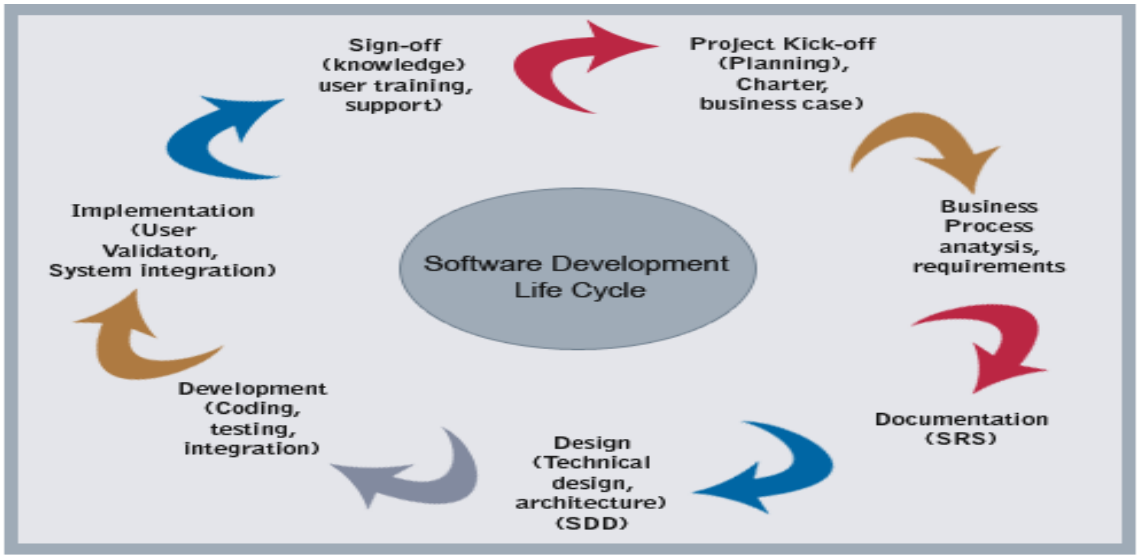

**Project Staffing Plan**

Primary task of our team is to build an effective project staffing plan to identify the following:

1. Skills Matrix - Required skills for each task in the project structure
2. Estimate man hours for each task
3. Resource matrix – required resources (internal and outsourced) - Training of in-house team, hiring, supplemental staffing using IT staffing vendors, etc. Planned hiring of new technical staff can help in transition. Critical staff has to be identified and managed carefully.

4. Resource Deployment Matrix - Resource start date and sign off date (optimum utilization) – Gradual increase / decrease of resources, rotation and re-deployment based on project delivery schedules / milestones. Right mix will ensure both internal staff and outsourced resources are deployed in a blended mode to avoid unexpected departure or absence of outside consultants.
5. Resource / Risk Management – Manage / monitor resources periodically and avoid project risks due to resource issues. Absence of skilled resources when required will lead to project delays and cost escalation. Project staffing needs careful planning as it involves lead time to source, screen, and successfully deploy the right technical and management resources. As part of risk and continuity management, full time internal staff must be allowed to grow during the implementation cycle so that knowledge gained is not lost.
6. A check list with all the above factors will help the program managers / project managers / HR to plan the internal and outsourcing staffing requirements, manage critical staff, transition planning, knowledge transfer and deliver projects on time within budget.

Better Minds Consulting provides end-to-end staffing services for software projects from **"kick-off to sign-off", also called SDLC (Software Development Life Cycle) as shown below.**




Our project staffing strategy ensures a seamless integration with the client team in a shared responsibility approach to ensure project success. Periodic meetings from Kick-off to sign-off helps in project coordination, managing T&M and meeting the client expectations. Timely monitoring leads to better visibility of resource utilization and cost control. Better minds specializes in providing technology and engineering consultants who are ready to roll with little or no learning curve. Our consultants have strong industry experience, and academic background. Our professionals constantly update their technology and domain skills to keep ahead.

Personnel Planning deals with staffing. Staffing deals with the appoint personnel for the position that is identified by the organizational structure.

It involves:

- Defining requirement for personnel

- Recruiting (identifying, interviewing, and selecting candidates)
- Compensating
- Developing and promoting agent

For personnel planning and scheduling, it is helpful to have efforts and schedule size for the subsystems and necessary component in the system.

At planning time, when the system method has not been completed, the planner can only think to know about the large subsystems in the system and possibly the major modules in these subsystems.

Once the project plan is estimated, and the effort and schedule of various phases and functions are known, staff requirements can be achieved.

From the cost and overall duration of the projects, the average staff size for the projects can be determined by dividing the total efforts (in person-months) by the whole project duration (in months).

Typically the staff required for the project is small during requirement and design, the maximum during implementation and testing, and drops again during the last stage of integration and testing.

Using the COCOMO model, average staff requirement for various phases can be calculated as the effort and schedule for each method are known.

When the schedule and average staff level for every action are well-known, the overall personnel allocation for the project can be planned.

This plan will indicate how many people will be required for different activities at different times for the duration of the project.

The total effort for each month and the total effort for each step can easily be calculated from this plan.

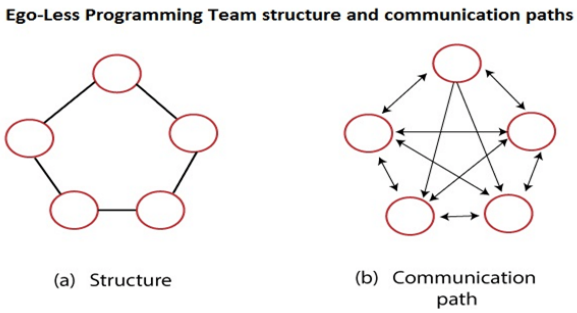
## Team Structure

Team structure addresses the issue of arrangement of the individual project teams. There are some possible methods in which the different project teams can be organized. There are primarily three formal team structures: **chief programmer, Ego-less or democratic, and the mixed team organizations** even several other variations to these structures are possible. Problems of various complexities and sizes often need different team structures for the chief solution.

## Ego-Less or Democratic Teams

Ego-Less teams subsist of a team of fewer programmers. The objective of the group is set by consensus, and input from each member is taken for significant decisions. Group leadership revolves among the group members. Due to its nature, egoless teams are consistently known as democratic teams.

The structure allows input from all representatives, which can lead to better decisions in various problems. This suggests that this method is well suited for long-term research-type projects that do not have time constraints.




# Chief Programmer Team

A chief-programmer team, in contrast to the ego-less team, has a hierarchy. It consists of a chief-programmer, who has a backup programmer, a program librarian, and some programmers.

The chief programmer is essential for all major technical decisions of the project.

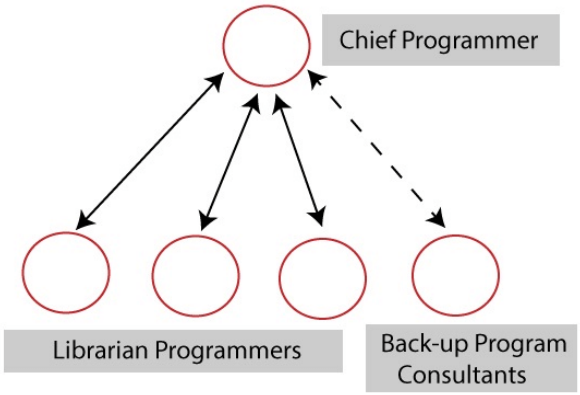
He does most of the designs, and he assigns coding of the different part of the design to the programmers.

The backup programmer uses the chief programmer makes technical decisions, and takes over the chief programmer if the chief programmer drops sick or leaves.

The program librarian is vital for maintaining the documentation and other communication-related work.

This structure considerably reduces interpersonal communication. The communication paths, as shown in fig:

Chief Programmer Team structure and communication paths




# Controlled Decentralized Team

## (Hierarchical Team Structure)

A third team structure known as the controlled decentralized team tries to combine the strength of the democratic and chief programmer teams.

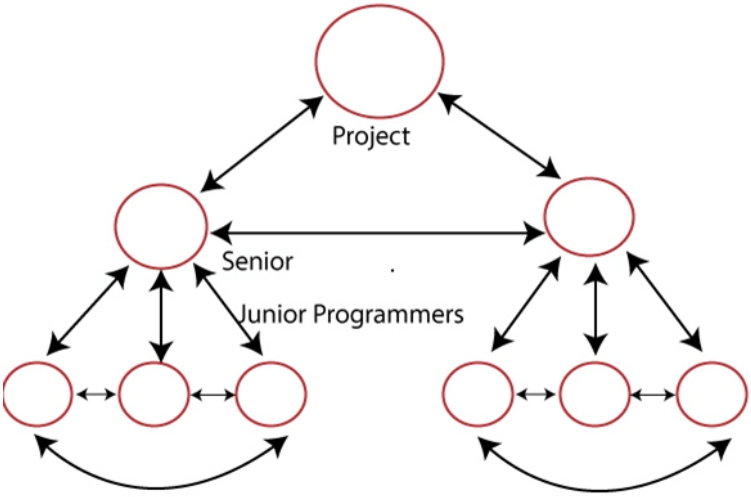
It consists of project leaders who have a class of senior programmers under him, while under every senior programmer is a group of a junior programmer.

The group of a senior programmer and his junior programmers behave like an ego-less team, but communication among different groups occurs only through the senior programmers of the group.

The senior programmer also communicates with the project leader.

Such a team has fewer communication paths than a democratic team but more paths compared to a chief programmer team.

This structure works best for large projects that are reasonably straightforward. It is not well suited for simple projects or research-type projects.

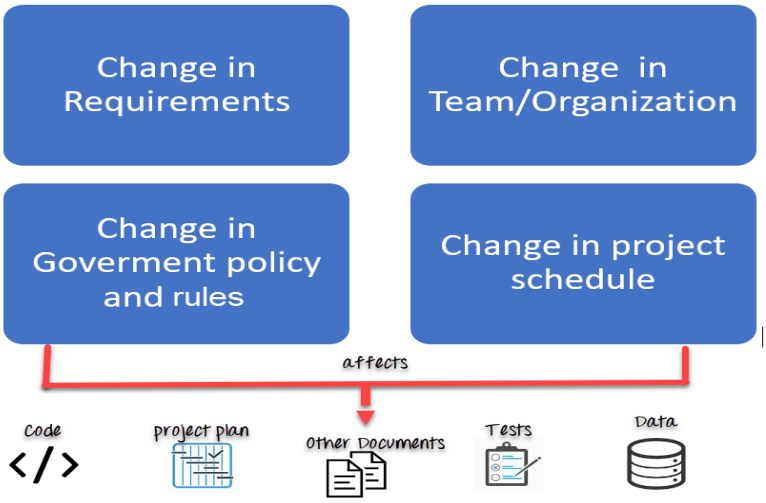


# Software Configuration Management

**Software Configuration Management** is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle. It is abbreviated as the SCM process in software engineering. The primary goal is to increase productivity with minimal mistakes.

The primary reasons for Implementing Software Configuration Management System are**:**

- There are multiple people working on software which is continually updating
- It may be a case where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently
- Changes in user requirement, policy, budget, schedule need to be accommodated.
- Software should able to run on various machines and Operating Systems
- Helps to develop coordination among stakeholders
- SCM process is also beneficial to control the costs involved in making changes to a system




Any change in the software configuration Items will affect the final product. Therefore, changes to configuration items need to be controlled and managed.

## Tasks in SCM process

1. Configuration Identification
2. Baselines
3. Change Control
4. Configuration Status Accounting
5. Configuration Audits and Reviews

## Configuration Identification:

Configuration identification is a method of determining the scope of the software system. With the help of this step, you can manage or control something even if you don't know what it is. It is a description that contains the CSCI type (Computer Software Configuration Item), a project identifier and version information.

**Activities during this process:**

- Identification of configuration Items like source code modules, test case, and requirements specification.
- Identification of each CSCI in the SCM repository, by using an object-oriented approach
- The process starts with basic objects which are grouped into aggregate objects. Details of what, why, when and by whom changes in the test are made
- Every object has its own features that identify its name that is explicit to all other objects
- List of resources required such as the document, the file, tools, etc.

Example:

Instead of naming a File login.php its should be named login_v1.2.php where v1.2 stands for the version number of the file

Instead of naming folder "Code" it should be named "Code_D" where D represents code should be backed up daily.

## Baseline:

A baseline is a formally accepted version of a software configuration item. It is designated and fixed at a specific time while conducting the SCM process. It can only be changed through formal change control procedures.

**Activities during this process:**

- Facilitate construction of various versions of an application
- Defining and determining mechanisms for managing various versions of these work products
- The functional baseline corresponds to the reviewed system requirements
- Widely used baselines include functional, developmental, and product baselines

In simple words, baseline means ready for release.

## Change Control:

Change control is a procedural method which ensures quality and consistency when changes are made in the configuration object. In this step, the change request is submitted to software configuration manager.

Activities during this process:

- Control ad-hoc change to build stable software development environment. Changes are committed to the repository
- The request will be checked based on the technical merit, possible side effects and overall impact on other configuration objects.
- It manages changes and making configuration items available during the software lifecycle

## Configuration Status Accounting:

Configuration status accounting tracks each release during the SCM process. This stage involves tracking what each version has and the changes that lead to this version.

**Activities during this process:**

- Keeps a record of all the changes made to the previous baseline to reach a new baseline
- Identify all items to define the software configuration
- Monitor status of change requests
- Complete listing of all changes since the last baseline
- Allows tracking of progress to next baseline
- Allows to check previous releases/versions to be extracted for testing

## Configuration Audits and Reviews:

Software Configuration audits verify that all the software product satisfies the baseline needs. It ensures that what is built is what is delivered.

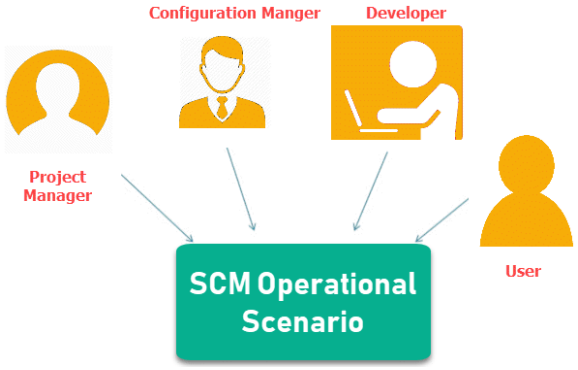
**Activities during this process:**

- Configuration auditing is conducted by auditors by checking that defined processes are being followed and ensuring that the SCM goals are satisfied.
- To verify compliance with configuration control standards. auditing and reporting the changes made
- SCM audits also ensure that traceability is maintained during the process.
- Ensures that changes made to a baseline comply with the configuration status reports

- Validation of completeness and consistency

## Participant of SCM process:

Following are the key participants in SCM




**1. Configuration Manager**

- Configuration Manager is the head who is Responsible for identifying configuration items.
- CM ensures team follows the SCM process
- He/She needs to approve or reject change requests

**2. Developer**

- The developer needs to change the code as per standard development activities or change requests. He is responsible for maintaining configuration of code.
- The developer should check the changes and resolves conflicts

**3. Auditor**

- The auditor is responsible for SCM audits and reviews.
- Need to ensure the consistency and completeness of release.

**4. Project Manager:**

- Ensure that the product is developed within a certain time frame
- Monitors the progress of development and recognizes issues in the SCM process
- Generate reports about the status of the software system
- Make sure that processes and policies are followed for creating, changing, and testing

**5. User**

The end user should understand the key SCM terms to ensure he has the latest version of the software

## Software Configuration Management Plan

The SCMP (Software Configuration management planning) process planning begins at the early phases of a project. The outcome of the planning phase is the SCM plan which might be stretched or revised during the project.

- The SCMP can follow a public standard like the IEEE 828 or organization specific standard
- It defines the types of documents to be management and a document naming. Example Test_v1
- SCMP defines the person who will be responsible for the entire SCM process and creation of baselines.
- Fix policies for version management & change control
- Define tools which can be used during the SCM process
- Configuration management database for recording configuration information.

## Software Configuration Management Tools

Any Change management software should have the following 3 Key features:

**Concurrency Management:**

When two or more tasks are happening at the same time, it is known as concurrent operation. Concurrency in context to SCM means that the same file being edited by multiple persons at the same time.

If concurrency is not managed correctly with SCM tools, then it may create many pressing issues.

**Version Control:**

SCM uses archiving method or saves every change made to file. With the help of archiving or save feature, it is possible to roll back to the previous version in case of issues.

**Synchronization:**

Users can checkout more than one files or an entire copy of the repository. The user then works on the needed file and checks in the changes back to the repository.They can synchronize their local copy to stay updated with the changes made by other team members.

Following are popular tools

**1. Git:** Git is a free and open source tool which helps version control. It is designed to handle all types of projects with speed and efficiency.

**2. Team Foundation Server:** Team Foundation is a group of tools and technologies that enable the team to collaborate and coordinate for building a product.

**3. Ansible**: It is an open source Software configuration management tool. Apart from configuration management it also offers application deployment & task automation.

# Software Quality Assurance

**Quality** defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability.

**There are two kinds of Quality:**


**Quality of Design:** Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

**Quality of conformance:** Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.

**Software Quality:** Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

**Quality Control:** Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

**Quality Assurance:** Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.

**Quality Assurance** focuses on how the engineering and management activity will be done?

As anyone is interested in the quality of the final product, it should be assured that we are building the right product.

It can be assured only when we do inspection & review of intermediate products, if there are any bugs, then it is debugged. This quality can be enhanced.

## Importance of Quality

We would expect the quality to be a concern of all producers of goods and services. However, the distinctive characteristics of software and in particular its intangibility and complexity, make special demands.

**Increasing criticality of software:** The final customer or user is naturally concerned about the general quality of software, especially its reliability. This is increasing in the case as organizations become more dependent on their computer systems and software is used more and more in safety-critical areas. For example, to control aircraft.

**The intangibility of software:** This makes it challenging to know that a particular task in a project has been completed satisfactorily. The results of these tasks can be made tangible by demanding that the developers produce 'deliverables' that can be examined for quality.

**Accumulating errors during software development:** As computer system development is made up of several steps where the output from one level is input to the next, the errors in the earlier ?deliverables? will be added to those in the later stages leading to accumulated determinable effects. In general the later in a project that an error is found, the more expensive it will be to fix. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly challenging to control.

**Software Quality Assurance (SQA)** is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of a software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of an Umbrella activity that is applied throughout the software process.

**Software Quality Assurance have:**

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

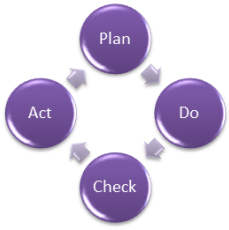
### Major Software Quality Assurance Activities:

1. **SQA Management Plan:**
   Make a plan how you will carry out the sqa through out the project. Think which set of software engineering activities are the best for project.check level of sqa team skills.
2. **Set The Check Points:**
   SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.
3. **Multi testing Strategy:**
   Do not depend on single testing approach. When you have lot of testing approaches available use them.
4. **Measure Change Impact:**
   The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.
5. **Manage Good Relations:**
   In the working environment managing the good relation with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don't play politics.

## How to do Quality Assurance: Complete Process

Quality assurance has a defined cycle called PDCA cycle or Deming cycle. The phases of this cycle are:

- Plan
- Do
- Check
- Act




These above steps are repeated to ensure that processes followed in the organization are evaluated and improved on a periodic basis. Let's look into the above steps in detail -

- Plan - Organization should plan and establish the process related objectives and determine the processes that are required to deliver a high-Quality end product.
- Do - Development and testing of Processes and also "do" changes in the processes

- Check - Monitoring of processes, modify the processes, and check whether it meets the predetermined objectives
- Act - Implement actions that are necessary to achieve improvements in the processes

An organization must use Quality Assurance to ensure that the product is designed and implemented with correct procedures. This helps reduce problems and errors, in the final product.

## Best practices for Quality Assurance:

- Create a Robust Testing Environment
- Select release criteria carefully
- Apply automated testing to high-risk areas to save money. It helps to fasten the entire process.
- Allocate Time Appropriately for each process
- It is important to prioritize bugs fixes based on software usage
- Form dedicated security and performance testing team
- Simulate customer accounts similar to a production environment

## Quality Assurance Functions:

There are 5 primary Quality Assurance Functions:

1. **Technology transfer:** This function involves getting a product design document as well as trial and error data and its evaluation. The documents are distributed, checked and approved
2. **Validation:** Here validation master plan for the entire system is prepared. Approval of test criteria for validating product and process is set. Resource planning for execution of a validation plan is done.
3. **Documentation:** This function controls the distribution and archiving of documents. Any change in a document is made by adopting the proper change control procedure. Approval of all types of documents.
4. **Assuring Quality of products**
5. **Quality improvement plans**

### Benefits of Software Quality Assurance (SQA):

1. SQA produce high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for long time.
5. High quality commercial software increase market share of company.

6. Improving the process of creating software.
7. Improves the quality of the software.

**Disadvantage of SQA:**
There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

# Project Monitoring

Project monitoring is an integral part of the project management. It provides understanding of the progress of the project so that appropriate corrective actions can be taken when the performance deviates significantly from the planned path. It consists of regular systematic collection and analysis of information to track the progress of the project implementation against pre-set targets and objectives. It is an important management tool which, if used properly, provides continuous feedback on the project implementation progress as well assists in the identification of potential successes and constraints to facilitate timely decisions.

Effective monitoring of the project is a critical element of good project management. It supports informed and timely decision making by the management and provides accountability for achieving results. It is a key part of project cycle management. It is to be built into the project at the planning stage. It is not an 'add on' tool which can be used during mid-way of the project implementation. On the other hand, it is to be woven throughout the project.

Project monitoring clarifies project objectives, links activities and their resources to objectives, translates objectives into performance indicators and sets targets, routinely collects data on these indicators, compares actual results with targets, and reports progress to the management and alerts the management about the problems which frequently gets cropped up during the implementation of the project. It provides information to the management whether the project is proceeding as per schedule relative to the targets or there is time over run in the project implementation. It also focuses, in particular, on the efficiency and the use of resources during the project implementation. It provides support to the management in its efforts to complete the project in time and within the budget.

Project monitoring activities take place in parallel with the project execution activities so that, while the project work is being executed, the project is being monitored by implementing the appropriate level of oversight for the purpose of the mid-way corrective actions. High quality monitoring of the project progress encourages timely decision making, ensures project accountability, and provides a robust foundation for successful completion of the project. It is through the continuous monitoring of

project performance that the management has an opportunity to learn about what is working well, what is lagging behind, and what challenges are arising.

**Definition of project monitoring**

Project monitoring is defined as a process which is performed to track the progress of project execution so that potential problems can be identified well in time for the taking of the corrective actions for the purpose of controlling the execution of the project. It is a continuing function during project implementation which provides management of an ongoing development intervention with indications of the extent of progress and achievement of objectives and progress in the use of allocated funds. It provides information and ensures the use of such information by management to assess project effects – both intentional and unintentional – and their impact. It aims at determining whether or not the intended project goals and objectives are being on the track.

Project monitoring can also be defined as the ongoing process by which management gets regular feedback on the progress being made towards achieving the goals and objectives of the project. It focuses on reviewing of progress against achieving of goals. In other words, monitoring is not only concerned with the taking of the actions but is also concerned with making the progress towards achievement of the results. In the more limited approach, monitoring can focus on tracking project with regards to the use of the resources. In the broader approach, monitoring also involves tracking strategies and actions being taken by management, and figuring out what new strategies and actions need to be taken to ensure progress towards the project objectives.

**General aspects of project monitoring**

Project monitoring provides opportunities at regular predetermined points to validate the logical progress of the project, its activities and their implementation and to make adjustments as needed. Good planning and designs alone do not ensure results in the execution of the project. Progress towards achieving results needs to be monitored. Equally, no amount of good monitoring alone can correct poor project designs, plans and results. Information from monitoring needs to be used to encourage improvements or reinforce plans. Information from systematic monitoring also provides critical input for project evaluation. It is very difficult to evaluate project progress which is not well designed and which is not systematically monitored.

Project monitoring provides records of activities and results, and signals problems to be remedied along the way. It is normally descriptive in nature and does not explain why a particular problem has arisen, or why a particular outcome has occurred or failed to occur. It helps the management for the fine-tuning of the implementation activities, reorientation of the project implementation by making appropriate changes in future planning. Without effective monitoring it is normally impossible to judge if work is going in the right direction, whether progress and success can be claimed and how further efforts can be improved.

**Project monitoring activities**

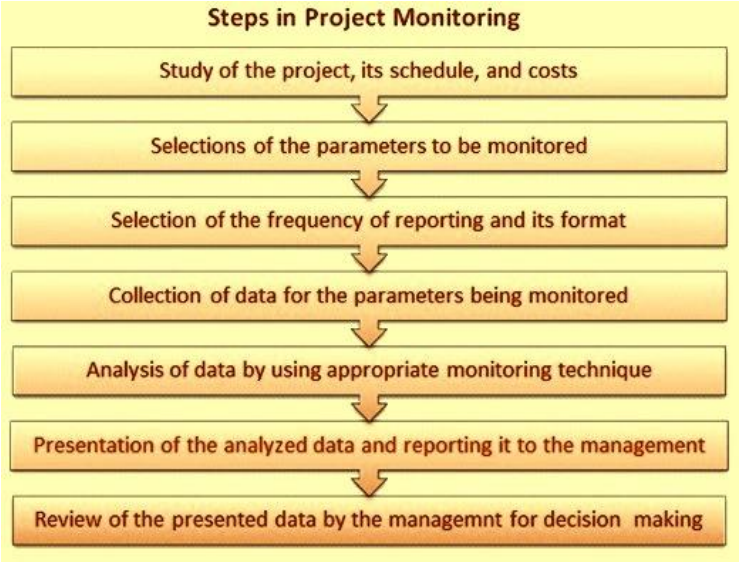
Project monitoring activities involve (i) the monitoring of actual project progress as compared to the planned project progress and the collection of key progress metrics such as risks, issues, changes and dependencies, and (ii) the reporting of project status, costs and outputs and other relevant information, at a summary level, to the management. The format and timing of project monitoring and reporting varies in each organization and also depends upon such items as the size, duration, risk and complexity of the project.

Project monitoring is carried out (i) measuring progress of project activities against established schedules and indicators of success, (ii) identifying factors affecting the progress of project activities, (iii) measuring the response of the decision taken on the project activities and its effect on the progress of project implementation, and (iv) to minimize the risks of project failure.

Timing and method of project monitoring are significant aspects of the project management. Important steps in project monitoring (Fig 1) include the following.

- Study of the project, its schedule, and costs.
- Selection of the parameters to be monitored.
- Selection of the frequency of reporting and its format.
- Collection of data for the parameters being monitored.
- Analysis of the data by using appropriate monitoring technique.
- Presentation of the analyzed data and reporting it to the management.
- Review of the presented data by the management for decision making.

Integrity and accuracy of the data and its proper analysis is very important in the project monitoring since it helps in taking of proper decisions for the project.




**Project monitoring process**

Process of the project monitoring is required to fulfill certain criteria. It is to be relevant towards meeting the needs of the project. It is to be efficient so that it facilitates the progress of the project. It is to be effective so that it helps the management to take the right decisions regarding the project. It has to impact the project in the positive manner by being responsive to the mid-way changes in the project. And above all it is to be sustainable and is to meet the project requirements during the life cycle of the project.

Each project is unique and differs from other project. It is therefore necessary the process of the project monitoring is suitably designed and is suitably aligned to the requirements necessary for the project implementation. However, the process of the project monitoring is to be simple, quick in providing the information for corrective action, cost-effective, flexible, accurate, comprehensive, relevant, and accessible. Project monitoring process is expected to lead to learning. It is to be transparent and consists of sharing of the information up and down.

One of the greatest weaknesses of project monitoring is the lack of effective and timely communication of information to the users. The employees monitoring the project frequently invest too much time and resources in gathering data which they frequently fail to interpret and present in a

form that which conveys the meaning of the progress made. The project monitoring is effective only if the collected data is properly analyzed and presented timely in a concise manner to the management for decision making.

The importance of communication in process of project monitoring is equally critical. Communication in project monitoring process acts as a lubricant facilitating and speeding up the project movement for the achievement the stated goals and objectives of the project.

There are various processes and tools which are normally used to assist the project monitoring. The process of project monitoring generally involve obtaining, analyzing and reporting of monitoring data. Specific processes and tools used for monitoring can vary from project to project and also to meet the requirements of the monitoring, but there are some overall best practices, which are summarized below.

- Monitoring of the project is to be well-focused to meet the needs of the project, the specific audiences and uses. It is to be sufficient and limited to what is necessary.
- Monitoring is to be systematic, based upon predetermined indicators and assumptions.
- Monitoring is also to look for unanticipated changes occurring in the project and its context, including any changes in project assumptions/risks. This information is useful for adjusting project implementation plans.
- Monitoring needs to be timely, so that it informs timely the progress made towards project implementation.
- As far as possible, monitoring is to participatory with the involvement of key players. This helps in the understanding and ownership of the analyzed data of the project monitoring.
- Monitoring information is not only meant for the project management but need to be shared with all the key players of the project.

Some of the most widely used tools for project monitoring, and their limitations include the following.

- Verbal communication – This is probably the most effective mode of communication. Among its advantages is that it is quick, and its presentation can be adapted to concerns and questions of the audience. However, this type of tool to communicate monitoring information can lead to misunderstandings and sometimes denial of information.

- Written communication – This is probably the most reliable mode of communication. Among its advantages is that it provides clear data and removes the possibility of any misunderstandings. However, it requires time and generally frequent follow up to receive monitoring data through written mode of communication.
- Meetings – The very nature of project management makes it inevitable that certain meetings are convened to communicate and share project information. Collection of data for the project monitoring can even require meetings with different players involved with project implementation. However, meetings to be effective as tools for project monitoring are to be focused for sharing and interchanging information, clarifying, stimulating, and seeking the best solutions regarding project performance.
- Reports – The monitoring reports are important since they are an essential part of project monitoring. Activities undertaken, inputs supplied, funds disbursed etc. have to be recorded and accounted for in the reports. However, reports are only effective if they are submitted to the right people at the right time to facilitate corrective decision making. Further the reports are to be concise and to the point so that they can get the needed attention.
- Diary notes – While many of the people involved in project monitoring do not use this mode of recording information, it remains an important option. It is essential to record key decisions, which may have been made at formal or informal meetings. However while taking diary notes, it is to be ensured that the date, time, place and the names of the people present are included when the decision are being taken.

However, it is often the experience from many projects that there are certain problems faced during the project monitoring. These problems include the following.

- Many of the reports gather dust in offices without being effectively used.
- Sometimes the wrong information is collected and analyzed, which may not be useful in decision-making.
- Some departments or units are not serous in providing the information necessary for the project monitoring. These departments are casual in providing the information.
- Many a times there are no feedback on the reports presented to the management and hence the people involved in monitoring of the project do not get motivation for the preparation of serious reports.

The process of the project monitoring is built up on the project design. It is to help the project management in making and the checking of the work plans, and other management tools used during the project implementation. The project monitoring process is to help the management to assess the

quality/capacity of existing project implementing arrangements with a view to providing support where required.

Project monitoring provides information enabling management to assess implementation progress and make timely decisions. It is concerned with verifying that project activities are being undertaken, services are being delivered, and the project is leading to the desired behaviour changes described in the project designs and plans. It is normally an internal organizational activity. It can be externally led though in that case active participation of project employees is needed. It is an essential part of good day-to-day practice for the project management and need to be integrated within the structure of the project management. It is important for making decisions on overall project direction. The activity of the project monitoring takes place during the implementation phase of the project and generally focuses on the implementation progress of the project.

Project monitoring needs a sound project plan which allows an assessment of performance to be made (by comparing actual achievements against plan). In turn, project monitoring fulfills the need to regularly review, update and improve the project implementation plan as per project requirements. Project monitoring requires that appropriate and useful indicators, targets and baseline information requirements are established. A clear project work plan which specifies the sequencing and timing of key activities/tasks facilitates effective project monitoring. Project monitoring helps the management in assessing the implementing agency’s capacity to manage the work of the project assigned to it, and thereby making judgement whether support activities and resources are needed for the project to progress within its plan and budget. Project monitoring also needs management to provide necessary incentives to monitor, so that monitoring is not simply undertaken as a bureaucratic requirement. Project management need to play playing an enhanced role as dialogue partner and thereby supporting the progressive development of effective process for the project monitoring.

The collected data during the project monitoring is normally analyzed and presented to the management with the active use of several tools. These tools include (i) critical path analysis (CPA), (ii) variance analysis, (iii) programme evaluation and review technique (PERT) charts, (iv) Gantt or bar charts, (v) work breakdown structure (WBS) technique, (vi) earned value analysis, (vii) critical ratios, (viii) logical framework matrix, (ix) benchmarking techniques, and (x) specialized information programmes, etc. The progress is usually monitored against certain milestones planned for the project execution. The analyzed data of the project monitoring constitutes critical part of the project management information system (PMIS).

Inadequate resources lead to poor quality of project monitoring. For ensuring effective and quality monitoring, it is critical to set aside adequate financial and human resources at the planning stage. The required financial and human resources for monitoring are to be considered within the overall costs of the project and not as additional costs.

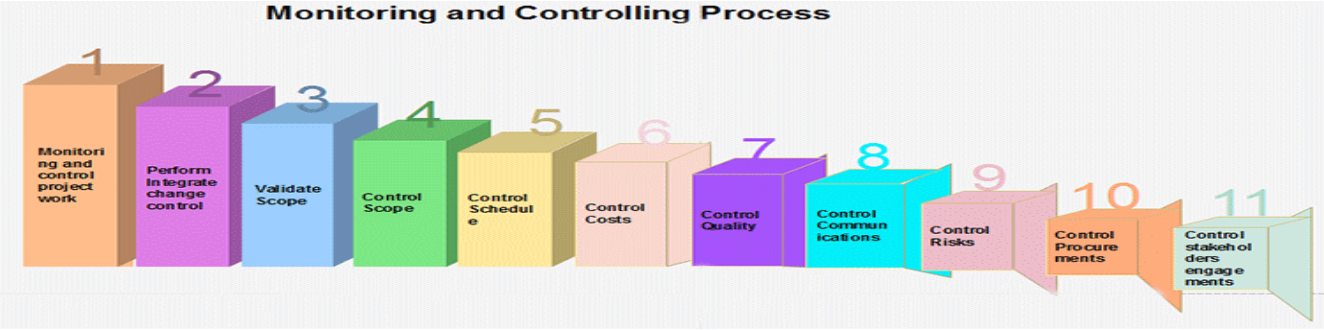
Project monitoring serves several purposes. It brings into focus the problems in the implementation of the projects which are brewing up and which need solutions for the smooth progress of the project. In the absence of effective monitoring, it is very difficult to know whether the intended results are being achieved as planned, what corrective action are needed to ensure achievement of the intended results during project execution, and whether initiatives are making positive contributions towards the project execution. Project monitoring always relate to pre-identified results in the development plan of the project. It is driven by the need to account for the achievement of intended results and provide a fact base to inform corrective decision making. It is an essential management tool to support the project and helps in the management commitment to accountability for results and effective utilization of resources entrusted to it.

Project monitoring is part of the project and project management not an addition to it. It is not to be regarded as merely a management or reporting requirement. Rather, it is to be regarded as an opportunity to (i) engage all the agencies participating in the project so that they feel ownership of results being achieved and are motivated to sustain them, (ii) demonstrate achievement of development results, how they benefit the project and all the agencies involved in addressing the operational challenges faced by them, and (iii) nurture an inclusive and purposeful monitoring culture to make the implementation and management effective and interesting as well as to ease gathering of data and evidence objectively to back achievements and make decisions.

## Monitering & control

Monitoring and Controlling are processes needed to track, review, and regulate the progress and performance of the project. It also identifies any areas where changes to the project management method are required and initiates the required changes.

The Monitoring & Controlling process group includes eleven processes, which are:




1. **Monitor and control project work:** The generic step under which all other monitoring and controlling activities fall under.
2. **Perform integrated change control:** The functions involved in making changes to the project plan. When changes to the schedule, cost, or any other area of the project management plan are necessary, the program is changed and re-approved by the project sponsor.
3. **Validate scope:** The activities involved with gaining approval of the project's deliverables.
4. **Control scope:** Ensuring that the scope of the project does not change and that unauthorized activities are not performed as part of the plan (scope creep).
5. **Control schedule:** The functions involved with ensuring the project work is performed according to the schedule, and that project deadlines are met.
6. **Control costs:** The tasks involved with ensuring the project costs stay within the approved budget.
7. **Control quality:** Ensuring that the quality of the project?s deliverables is to the standard defined in the project management plan.
8. **Control communications:** Providing for the communication needs of each project stakeholder.
9. **Control Risks:** Safeguarding the project from unexpected events that negatively impact the project's budget, schedule, stakeholder needs, or any other project success criteria.
10. **Control procurements:** Ensuring the project's subcontractors and vendors meet the project goals.
11. **Control stakeholder engagement:** The tasks involved with ensuring that all of the project's stakeholders are left satisfied with the project work.

# Software Testing

Software Testing is a process of evaluating the functionality of a software application to find any software bugs. It checks whether the developed software met the specified requirements and identifies any defect in the software in order to produce a quality product. It is basically executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

It is also stated as the process of verifying and validating a software product. It checks whether the software product:

- 
  - Meets the business and technical requirements that guided its design and development
  - Works as per the requirement
  - Can be implemented with the same characteristics

Now let's move ahead and know more about the importance of testing.

## Why do we need Software Testing?

The software application's success rate controls the growth of our business. It plays an important role for the development of software applications and products.

We need software testing for the following reasons:

1. **Cost Effective** – Testing has many benefits and one of the most important ones is cost-effectiveness. Testing our project on time can save money in the long run. Software development consists of many stages and if bugs are caught in the earlier stages it costs much less to fix them.
2. **Security** – This is the most sensitive and vulnerable part of software testing. Users are always looking for trusted products that they can rely on. It helps in removing problems and risks beforehand.
3. **Product Quality** – In order to make your product vision come to life, it has to work as planned. It is important to follow the product requirements because it helps you get the required end results.
4. **Customer Satisfaction** – The ultimate goal for a product owner is to give the best customer satisfaction. Software should be tested in order to bring the best user experience possible. Being the best product in this saturated market will help you gain trustworthy clients which will have great long-term effects.

Now let's move ahead and have a look at some of the principles of Software Testing.

## Principles of Software Testing

Testing of software is exceptionally imaginative and an intellectual task for testers to perform. Testing of software or applications consist of some principles that play a significant role for a software tester while testing the project.

The Principles of Software Testing are as follows :

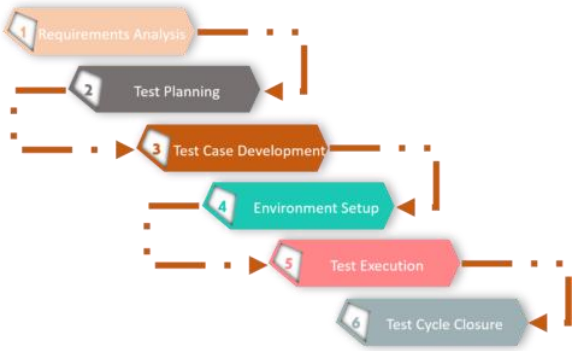
1. **Software testing can help in detecting bugs**: Testing any software or project can help in revealing a few or some defects that may or may not be detected by developers. However, testing of software

alone cannot confirm that your developed project or software is error free. Hence, it's essential to devise test cases and find out as many defects as possible.

2. **Testing with effectiveness is impossible**: Until your project or application under test has a straightforward structure having limited input, it won't be likely or achievable to check and test all feasible sets of data, modules, and scenarios.
3. **Early testing**: The earlier you will begin to test your project or software the better you will find to utilize your existing time.
4. **Defect in clustering**: At the time of testing, you can observe that majority of the defects or bugs reported are because of a small number of modules inside your software or system.
5. **Software testing is context-dependent**: Various methods, procedures, and kinds of testing are there which defines the type and characteristics of the application. For example, an application related to health device needs more testing and doctor based feedbacks than a game or small software.
6. **Error free or Bug-free software is a myth**: Just because when a tester tested an application and didn't detect any defects in that project, doesn't indicate or imply that your software is ready for shipping.

Now that we know about software testing and the principles of it, let's move ahead and have a look at the life cycle of software testing.

## Software Testing Life Cycle (STLC)




It is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. It defines a series of activities conducted to perform Software Testing. It also identifies what test activities to carry out and when to accomplish those test activities. In STLC process, each activity is carried out in a planned and systematic way and each phase has different goals and deliverable.

The different phases of Software testing life cycle are:

- **Requirement Analysis** – Requirement Analysis is the first step involved in Software testing life cycle. In this step, Quality Assurance (QA) team understands the requirement in terms of what we will testing & figure out the testable requirements.
- **Test Planning** – Test Planning is most important phase of Software testing life cycle where all testing strategy is defined. This phase is also called as **Test Strategy** phase. In this phase,Test Manager is involved to determine the effort and cost estimates for entire project. It defines the objective & scope of the project.
- **Test Case Development** – The Test case development begins once the test planning phase is completed. This is the phase of STLC where testing team notes the detailed test cases. Along with test cases, testing team also prepares the test data for testing. Once the test cases are ready then these test cases are reviewed by peer members or QA lead.

- **Test Environment Setup** – Setting up the test environment is vital part of the Software Testing Life Cycle. A testing environment is a setup of software and hardware for the testing teams to execute test cases. It supports test execution with hardware, software and network configured.
- **Test Execution** – The next phase in Software Testing Life Cycle is Test Execution. Test execution is the process of executing the code and comparing the expected and actual results. When test execution begins, the test analysts start executing the test scripts based on test strategy allowed in the project.
- **Test Cycle Closure** – The final phase of the Software Testing Life Cycle is Test Cycle Closure. It involves calling out the testing team member meeting & evaluating cycle completion criteria based on Test coverage, Quality, Cost, Time, Critical Business Objectives, and Software.

## Types of Testing

Testing is an integral part of any successful software project. The type of testing depends on various factors, including project requirements, budget, timeline, expertise, and suitability. Software testing is a huge domain but it can be broadly categorized into two areas such as :

- **Manual Testing** – Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automation tools. It means the application is tested manually by QA testers.

Tests need to be performed manually in every environment, using a different data set and the success or failure rate of every transaction should be recorded.This type of testing requires the tester's knowledge, experience, analytical/logical skills, creativity, and intuition.

Some of the **tools** used for **Manual Testing** are:

1. Stryka
2. Bugzilla
3. Jira
4. Mantis
5. Trac
6. Redmine
7. Fogbuz
8. Lighthouse

- **Automated Testing** – Automation testing is an Automatic technique where the tester writes scripts by own and uses suitable software to test the software. It is basically an automation process of a manual process. Like regression testing, Automation testing also used to test the application from load, performance and stress point of view.

Automated testing allows you to execute repetitive task and regression test without the intervention of manual tester. Even though all processes are performed automatically, automation requires some manual effort to create initial testing scripts.

Some of the **tools** used for **Automated Testing** are :

1. Selenium
2. TestingWhiz
3. Ranorex
4. Sahi
5. Waitir
6. WaitiN
7. Tosca TestSuite

Selenium is the household name when it comes to test automation. It is considered the industry standard for user interface automation testing of Web applications.

### Software Testing Companies

There are many big corporate companies providing testing services along with other core software development services. Let's have a look at some of the big names among the Software Testing companies :

- ScienceSoft
- Capgemini
- Wipro
- Cognizant
- HP
- Infosys
- TCS
- Hexaware

# Organization structures for testing teams

**A goal can be described as (i) a statement of intent, or (ii) a statement of a accomplishment that an individual or an organization wants to achieve.**

A goal statement relates to an area where an individual, group, or organization wants to make improvements. Goals project future states of an organization, a group, or an individual. In an organization there is often a hierarchy of goals. At the top level are general organizational goals. There are intermediate-level goals that may be associated with a particular organizational functional unit. Individual projects have specific goals. These usually reflect organizational goals. There are personal-level goals as well. Each individual in an organization has a set of goals for self-improvement so that he or she can more effectively contribute to the project, functional unit, and organization as a whole.

Goal statements can express expectations in quantitative terms or be more general in nature. For the testing goals below, goals 1 and 2 express what is to be achieved in a more quantitative manner than goals 3 and 4.

**1.** One-hundred percent of testing activities are planned.

**2.** The degree of automation for regression testing is increased from 50% to 80% over the next 3 years.

**3.** Testing activities are performed by a dedicated testing group.

**4.** Testing group members have at least a bachelor-level degree and have taken a formal course in software testing.

In general, quantitative goals are more useful. These are measurable goals, and give an organization, group, or individual the means to evaluate progress toward achieving the goal. In the testing domain, goal statements should provide a high-level vision of what testing is to accomplish in the organization with respect to quality of process and product. In addition to general testing goal statements, lower-level goal statements should be developed for all levels of testing. Goals for the education and training of testing personnel should also be included with testing goal statements. Test plans should express testing goals for each project. These reflect overall organizational testing goals as well as specific goals for the project.

The TMM itself is built on a hierarchy of high-level testing maturity goals and subgoals which support the growth of an effective software testing process and promote high software quality. The TMM can be used by decision-makers in an organization to develop both long- and short term testing goals based on the TMM goal hierarchy.

**A policy can be defined as a high-level statement of principle or course of action that is used to govern a set of activities in an organization.**

Because a policy provides the vision and framework for decision making, it is important to have the policy formally adopted by the organization, documented, and available for all interested parties. An intraorganizational web site is suggested as a location for policy statements. This would allow for updates and visibility within the organization. A policy statement should be formulated by a team or task force consisting of upper management, executive personnel, and technical staff. In the case of testing, a testing policy statement is used to guide the course of testing activities and test process evolution. It should be agreed upon as workable by all concerned.

Testing policy statements reflect, integrate, and support achievement of testing goals. These goals in turn often target increasing software quality and improving customer satisfaction. Test policies also provide high-level guidance as to how testing is to be done in the organization, how its effectiveness will be evaluated, who will be responsible, and what choices of resources are possible. They should be explicit enough to guide decisions on all important testing issues, for example, how to test, what to test, and who will test. Policies are not written in stone, and as an organization grows in maturity its policies will change and mature. The task force should establish documented procedures for policy change. A brief outline of a sample testing policy statement appropriate for a TMM level 2 organization follows.

**Testing Policy : Organization X**

Our organization, the X Corporation, realizes that testing is an important component of the software development process and has a high impact on software quality and the degree of customer satisfaction. To ensure that our testing process is effective and that our software products meet the client‘s requirements we have developed and adopted the following testing policy statement.

**1.** Delivering software of the highest quality is our company goal. The presence of defects has a negative impact on software quality. Defects affect the correctness, reliability, and usability of a software product, thus rendering it unsatisfactory to the client. We define a testing activity as a set of tasks whose purpose is to reveal functional and quality- related defects in a software deliverable. Testing activities include traditional execution of the developing software, as well as reviews of the software deliverables produced at all stages of the life cycle. The aggregation of all testing activities performed in a systematic manner supported by organizational policies, procedures, and standards constitutes the testing process.

**2.** A set of testing standards must be available to all interested parties on an intraorganizational web site. The standards contain descriptions of all test-related documents, prescribed templates, and the methods, tools, and procedures to be used for testing. The standards must specify the types of projects that each of these items is to be associated with.

**3.** In our organization the following apply to all software development/ maintenance projects:

• Execution-based tests must be performed at several levels such as unit , integration, system, and acceptance tests as appropriate for each software product.

• Systematic approaches to test design must be employed that include application of both white and black box testing methods.

• Reviews of all major product deliverables such as requirements and design documents, code, and test plans are executionbase

• Testing must be planned for all projects. Plans must be developed for all levels of executionbased testing as well as for reviews of deliverables.

Test plan templates must be included in organizational standards documents and implemented online. A test plan for a project must be compatible with the project plan for that project. Test plans must be approved by the project manager and technical staff. Acceptance test plans must also be approved by the client.

• Testing activities must be monitored using measurements and milestones to ensure that they are proceeding according to plan.

• Testing activities must be integrated into the software life cycle and carried out in parallel with other development activities. The extended modified V-model as shown in the testing standards document has been adopted to support this goal.

• Defects uncovered during each test must be classified and recorded.

• There must be a training program to ensure that the best testing practices are employed by the testing staff.

**4.** Because testing is an activity that requires special training and an impartial view of the software, it must be carried out by an independent testing group. Communication lines must be established to support cooperation between testers and developers to ensure that the software is reliable, safe, and meets client requirements.

**5.** Testing must be supported by tools, and, test-related measurements must be collected and used to evaluate and improve the testing process and the software product.

**6.** Resources must be provided for continuos test process improvement.

**7.** Clients/developer/tester communication is important, and clients must be involved in acceptance test planning, operational profile development, and usage testing when applicable to the project. Clients must sign off on the acceptance test plan and give approval for all changes in the acceptance test plan.

**8.** A permanent committee consisting of managerial and technical staff must be appointed to be responsible for distribution and maintenance of organizational test policy statements. Whatever the nature of the test policy statement, it should have strong support and continual commitment from management. After the policy statement has been developed, approved, and distributed, a subset of the task force should be appointed to permanently oversee policy implementation and change.

**Debugging Policy : Organization X**

Our organization, the X Corporation, is committed to delivering highquality software to our customers. Effective testing and debugging processes are essential to support this goal. It is our policy to separate testing and debugging, and we consider them as two separate processes. Each has different psychologies, goals, and requirements. The resources, training, and tools needed are different for both. To support the separation of these two processes we have developed individual testing and debugging policies. Our debugging policy is founded on our quality goal to remove all defects from our software that impact on our customers‘ ability to use our software effectively, safely, and economically. To achieve this goal we have developed the following debugging policy statement.

**1.** Testing and debugging are two separate processes. Testing is the process used to detect (reveal) defects. Debugging is the process dedicated to locating the defects, repairing the code, and retesting the software. Defects are anomalies that impact on software functionality as well as on quality attributes such as performance, security, ease of use, correctness, and reliability.

**2.** Since debugging is a timely activity, all project schedules must allow for adequate time to make repairs, and retest the repaired software.

**3.** Debugging tools, and the training necessary to use the tools, must be available to developers to support debugging activities and tasks.

**4.** Developers/testers and SQA staff must define and document a set of defect classes and defect severity levels. These must be must be available to all interested parties on an intraorganizational web site, and applied to all projects.

When failures are observed during testing or in operational software they are documented. A problem, or test incident, report is completed by the developer/tester at testing time and by the users when a failure/ problem is observed in operational software. The problem report is forwarded to the development group. Both testers/developers and SQA staff must communicate and work with users to gain an understanding of the problem. A fix report must be completed by the developer when the defect is repaired and code retested. Standard problem and fix report forms must be available to all interested parties on an intraorganizational web site, and applied to all projects.

**6.** All defects identified for each project must be cataloged according to class and severity level and stored as a part of the project history.

**7.** Measurement such as total number of defects, total number of defects/ KLOC, and time to repair a defect are saved for each project.

**8.** A permanent committee consisting of managerial and technical staff must be appointed to be responsible for distribution and maintenance of organizational debugging policy statements.

# Test Management

Test management, process of managing the tests. A test management is also performed using tools to manage both types of tests, automated and manual, that have been previously specified by a test procedure.

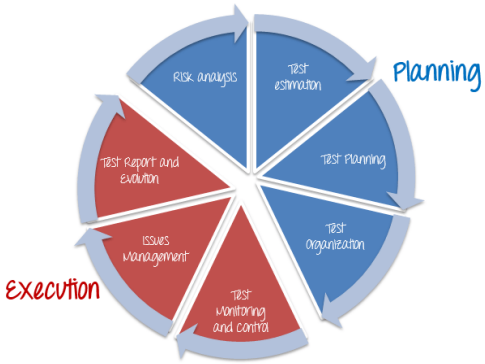
Test management tools allow automatic generation of the requirement test matrix (RTM), which is an indication of functional coverage of the application under test (SUT).

Test Management tool often has multifunctional capabilities such as testware management, test scheduling, the logging of results, test tracking, incident management and test reporting.

### Test Management Responsibilities:

- Test Management has a clear set of roles and responsibilities for improving the quality of the product.
- Test management helps the development and maintenance of product metrics during the course of project.
- Test management enables developers to make sure that there are fewer design or coding faults.

## Test Management Phases




## Test Management Process

There are two main Parts of Test Management Process: -

- Planning
    1. Risk Analysis
    2. Test Estimation
    3. Test Planning
    4. Test Organization
- Execution
    1. Test Monitoring and Control
    2. Issue Management
    3. Test Report and Evaluation

# Planning

## Risk Analysis and Solution

**Risk** is the potential loss (an undesirable outcome, however not necessarily so) resulting from a given action or an activity.

Risk Analysis is the first step which Test Manager should consider before starting any project. Because all projects may contain risks, early risk detection and identification of its solution will help Test Manager to **avoid** potential loss in the future & save on project cost.

## Test Estimation

An estimate is a forecast or prediction. Test Estimation is approximately determining **how long** a task would take to complete. Estimating effort for the test is one of the **major** and **important** tasks in Test Management.

Benefits of correct estimation:

1. Accurate test estimates lead to better planning, execution and monitoring of tasks under a test manager's attention.
2. Allow for more accurate scheduling and help realize results more confidently.

## Test Planning

A Test Plan can be defined as a document describing the **scope**, **approach**, **resources**, and **schedule** of intended Testing activities.

A project may **fail** without a complete Test Plan. Test planning is particularly important in large software system development.

In software testing, a test plan gives **detailed** testing information regarding an upcoming testing effort, including:

- Test Strategy
- Test Objective
- Exit /Suspension Criteria
- Resource Planning
- Test Deliverables

## Test Organization

Now you have a Plan, but how will you stick to the plan and execute it? To answer that question, you have **Test Organization** phase.

Generally speaking, you need to organize an effective Testing Team. You have to assemble a skilled team to run the ever-growing testing engine effectively.

# Execution

## Test Monitoring and Control

What will you do when your project runs **out of resource** or **exceeds** the time schedule? You need to Monitor and Control Test activities to bring it back on schedule.

Test Monitoring and Control is the process of overseeing all the metrics necessary to ensure that the project is running well, on schedule, and not out of budget.

### Monitoring

Monitoring is a process of **collecting**, **recording**, and **reporting** information about the project activity that the project manager and stakeholder needs to know

To Monitor, Test Manager does following activities

- **Define** the project goal, or project performance standard
- **Observe** the project performance, and compare between the actual and the planned performance expectations
- **Record** and **report** any detected problem which happens to the project

### Controlling

Project Controlling is a process of using data from monitoring activity to bring actual performance to planned performance.

In this step, the Test Manager takes action to correct the deviations from the plan. In some cases, the plan has to be **adjusted** according to project situation.

## Issue Management

As mentioned in the beginning of the topics, all projects may have **potential** risk. When the risk happens, it becomes an **issue**.

In the life cycle of any project, there will be always an **unexpected** problems and questions that crop up. For an example:

- The company cuts down your project budget
- Your project team lacks the skills to complete project
- The project schedule is too tight for your team to finish the project at the deadline.

Risk to be avoided while testing:

- **Missing** the deadline
- **Exceed** the project budget
- **Lose** the customer trust

When these issues arise, you have to be ready to deal with them – or they can potentially affect the project's outcome.

## Test Report & Evaluation

The project has already completed. It’s now time for look back what you have done.

The purpose of the Test Evaluation Reports is:

“Test Evaluation Report” describes the results of the Testing in terms of Test coverage and exit criteria. The data used in Test Evaluation are based on the test results data and test result summary.

# Organization Structure

Any operating organization should have its own structure in order to operate efficiently. For an organization, the organizational structure is a hierarchy of people and its functions.

The organizational structure of an organization tells you the character of an organization and the values it believes in. Therefore, when you do business with an organization or getting into a new job in an organization, it is always a great idea to get to know and understand their organizational structure.

Depending on the organizational values and the nature of the business, organizations tend to adopt one of the following structures for management purposes.

Although the organization follows a particular structure, there can be departments and teams following some other organizational structure in exceptional cases.

Sometimes, some organizations may follow a combination of the following organizational structures as well.

## Organizational Structure Types

Following are the types of organizational structures that can be observed in the modern business organizations.

### Bureaucratic Structures

Bureaucratic structures maintain strict hierarchies when it comes to people management. There are three types of bureaucratic structures:

**1 - Pre-bureaucratic structures**

This type of organizations lacks the standards. Usually this type of structure can be observed in small scale, start-up companies. Usually the structure is centralized and there is only one key decision maker.

The communication is done in one-on-one conversations. This type of structures is quite helpful for small organizations due to the fact that the founder has the full control over all the decisions and operations.

**2 - Bureaucratic structures**

These structures have a certain degree of standardization. When the organizations grow complex and large, bureaucratic structures are required for management. These structures are quite suitable for tall organizations.

**3 - Post-bureaucratic Structures**

The organizations that follow post-bureaucratic structures still inherit the strict hierarchies, but open to more modern ideas and methodologies. They follow techniques such as total quality management (TQM), culture management, etc.

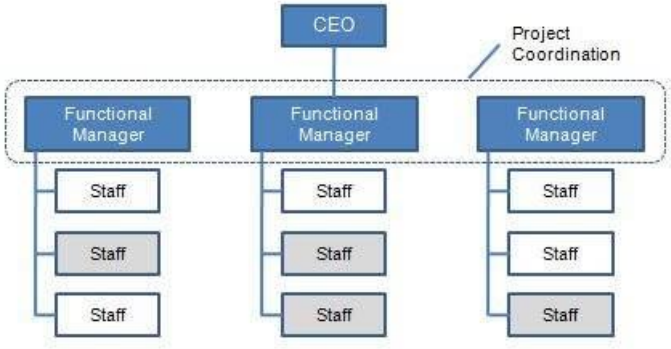
## Functional Structure

The organization is divided into segments based on the functions when managing. This allows the organization to enhance the efficiencies of these functional groups. As an example, take a software company.

Software engineers will only staff the entire software development department. This way, management of this functional group becomes easy and effective.

Functional structures appear to be successful in large organization that produces high volumes of products at low costs. The low cost can be achieved by such companies due to the efficiencies within functional groups.

In addition to such advantages, there can be disadvantage from an organizational perspective if the communication between the functional groups is not effective. In this case, organization may find it difficult to achieve some organizational objectives at the end.

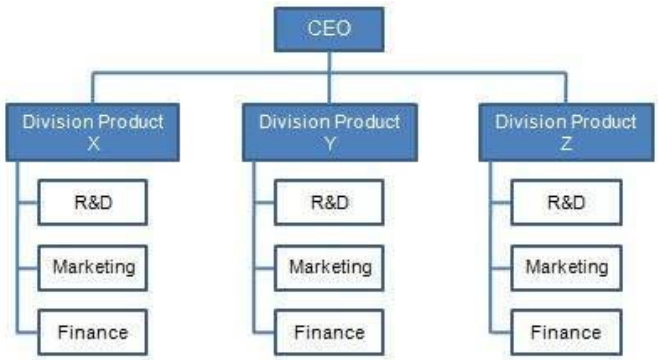



## Divisional Structure

These types of organizations divide the functional areas of the organization to divisions. Each division is equipped with its own resources in order to function independently. There can be many bases to define divisions.

Divisions can be defined based on the geographical basis, products/services basis, or any other measurement.

As an example, take a company such as General Electrics. It can have microwave division, turbine division, etc., and these divisions have their own marketing teams, finance teams, etc. In that sense, each division can be considered as a micro-company with the main organization.



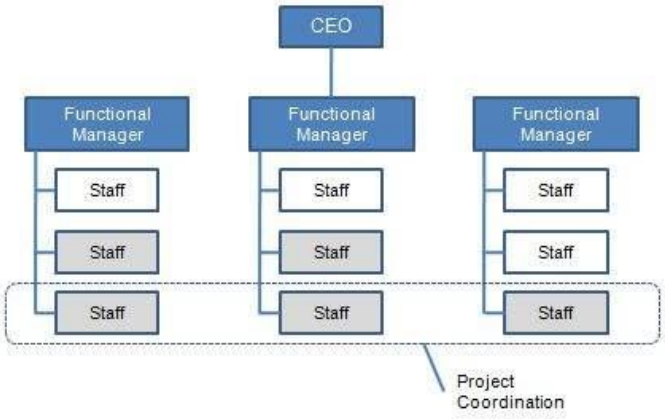
### Matrix Structure

When it comes to matrix structure, the organization places the employees based on the function and the product.

The matrix structure gives the best of the both worlds of functional and divisional structures.

In this type of an organization, the company uses teams to complete tasks. The teams are formed based on the functions they belong to (ex: software engineers) and product they are involved in (ex: Project A).

This way, there are many teams in this organization such as software engineers of project A, software engineers of project B, QA engineers of project A, etc.




## Conclusion

Every organization needs a structure in order to operate systematically. The organizational structures can be used by any organization if the structure fits into the nature and the maturity of the organization.

In most cases, organizations evolve through structures when they progress through and enhance their processes and manpower. One company may start as a pre-bureaucratic company and may evolve up to a matrix organization.

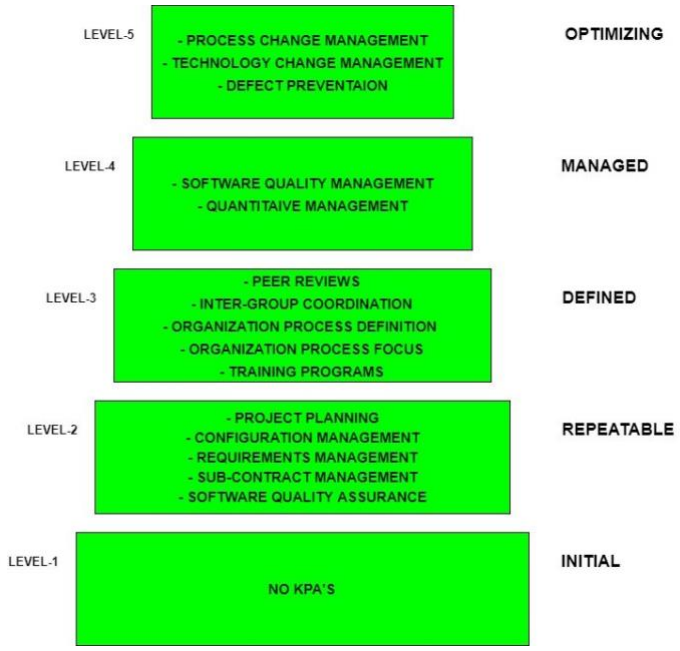
# Capability Maturity Model (CMM)

CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

- It is not a software process model. It is a framework which is used to analyse the approach and techniques followed by any organization to develop a software product.
- It also provides guidelines to further enhance the maturity of those software products.
- It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
- This model describes a strategy that should be followed by moving through 5 different levels.
- Each level of maturity shows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA's).

Key Process Areas (KPA's):
Each of these KPA's defines the basic requirements that should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.




The 5 levels of CMM are as follows:

Level-1: Initial –

- No KPA's defined.
- Processes followed are adhoc and immature and are not well defined.
- Unstable environment for software dvelopment.
- No basis for predicting product quality, time for completion, etc.

Level-2: Repeatable –

- Focuses on establishing basic project management policies.
- Experience with earlier projects is used for managing new similar natured projects.

KPA's:

- Project Planning- It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for successful completion of a good quality software.
- Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
- Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
- Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.
- Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while development.

Level-3: Defined –

- At this level, documentation of the standard guidelines and procedures takes place.
- It is a well defined integrated set of project specific software engineering and management processes.

KPA's:

- Peer Reviews- In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- Intergroup Coordination- It consists of planned interactions between different development teams to ensure efficient and proper fulfilment of customer needs.
- Organization Process Definition- It's key focus is on the development and maintenance of the standard development processes.
- Organization Process Focus- It includes activities and practices that should be followed to improve the process capabilities of an organization.
- Training Programs- It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

Level-4: Managed –

- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.

KPA's:

- Software Quality Management- It includes the establishment of plans and strategies to develop a quantitative analysis and understanding of the product's quality.
- Quantitative Management- It focuses on controlling the project performance in a quantitative manner.

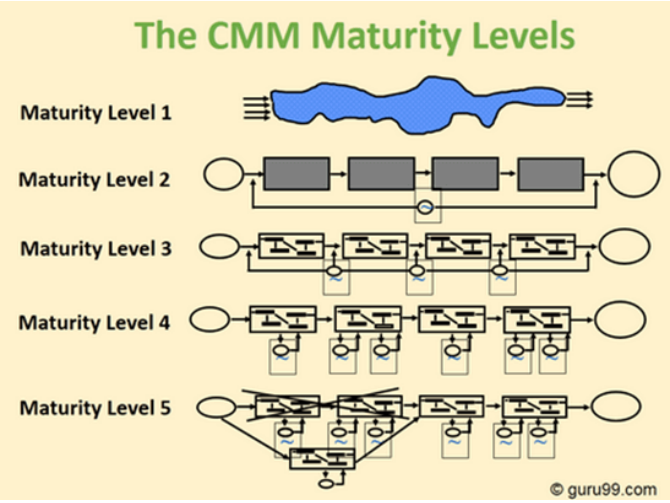
Level-5: Optimizing –

- This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.
- Use of new tools, techniques and evaluation of software processes is done to prevent recurrence of known defects.

KPA's:

- Process Change Management- Its focus is on the continuous improvement of organization's software processes to improve productivity, quality and cycle time for the software product.
- Technology Change Management- It consists of identification and use of new technologies to improve product quality and decrease the product development time.
- Defect Prevention- It focuses on identification of causes of defects and to prevent them from recurring in future projects by improving project defined process.

Following diagram, gives a pictorial representation of what happens at different CMM level




## How long does it Take to Implement CMM?

CMM is the most desirable process to maintain the quality of the product for any software development company, but its implementation takes little longer than what is expected.

- CMM implementation does not occur overnight

- It's just not merely a "paperwork."
- Typical times for implementation is
    - 3-6 months **->** for preparation
    - 6-12 months **->** for implementation
    - 3 months **->** for assessment preparation
    - 12 months **->b**for each new level

## Internal Structure of CMM

Each level in CMM is defined into **key process area or KPA**, except for level-1. Each KPA defines a cluster of related activities, which when performed collectively achieves a set of goals considered vital for improving software capability

For different CMM levels, there are set of KPA's, for instance for CMM model-2, KPA are

- REQM- Requirement Management
- PP- Project Planning
- PMC- Project Monitoring and Control
- SAM- Supplier Agreement Management
- PPQA-Process and Quality Assurance
- CM-Configuration Management

Likewise, for other CMM models, you have specific KPA's. To know whether implementation of a KPA is effective, lasting and repeatable, it is mapped on following basis

1. Commitment to perform
2. Ability to perform
3. Activities perform
4. Measurement and Analysis
5. Verifying implementation

## Limitations of CMM Models

- CMM determines what a process should address instead of how it should be implemented
- It does not explain every possibility of software process improvement
- It concentrates on software issues but does not consider strategic business planning, adopting technologies, establishing product line and managing human resources
- It does not tell on what kind of business an organization should be in
- CMM will not be useful in the project having a crisis right now

## Why Use CMM?

Today CMM act as a "seal of approval" in the software industry. It helps in various ways to improve the software quality.

- It guides towards repeatable standard process and hence reduce the learning time on how to get things done
- Practicing CMM means practicing standard protocol for development, which means it not only helps the team to save time but also gives a clear view of what to do and what to expect
- The quality activities gel well with the project rather than thought of as a separate event
- It acts as a commuter between the project and the team
- CMM efforts are always towards the improvement of the process

# ISO 9000

ISO (International Standards Organization) is a group or consortium of 63 countries established to plan and fosters standardization. ISO declared its 9000 series of standards in 1987. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the guidelines for maintaining a quality system. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

## Types of ISO 9000 Quality Standards


The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.

1. **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
2. **ISO 9002:** This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel

and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.

3. **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

## How to get ISO 9000 Certification?

An organization determines to obtain ISO 9000 certification applies to ISO registrar office for registration. The process consists of the following stages:


1. **Application:** Once an organization decided to go for ISO certification, it applies to the registrar for registration.
2. **Pre-Assessment:** During this stage, the registrar makes a rough assessment of the organization.
3. **Document review and Adequacy of Audit:** During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
4. **Compliance Audit:** During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
5. **Registration:** The Registrar awards the ISO certification after the successful completion of all the phases.
6. **Continued Inspection:** The registrar continued to monitor the organization time by time.

# CASE TOOLS

CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

**CASE tools** are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

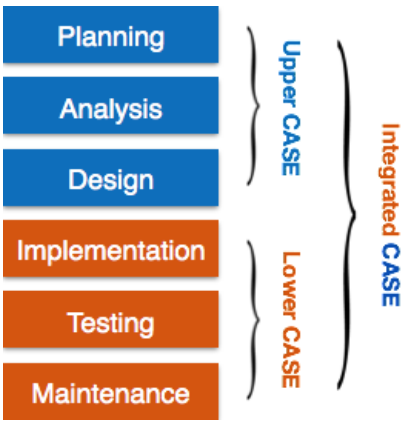
There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

## Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.




- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.
- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

## Scope of Case Tools

The scope of CASE tools goes throughout the SDLC.

# Case Tools Types

Now we briefly go through various CASE tools

### Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

### Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

### Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

### Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

### Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

### Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

### Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

### Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

### Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

### Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

### Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

### Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

### Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

## Features of CASE tools

CASE tools, which are sometimes called integrated CASE or I-CASE tools, cover all aspects of the software development lifecycle, which includes writing the code, implementation and maintenance. The tools help in every aspect of development work: managing, modeling, error-checking, version control, designing, diagraming tools, prototyping and other aspects associated with software engineering. Compilers and testing tools are also considered part of the CASE tool set.

Everything is centralized in a CASE repository, which provides an integrated system for project management information, code specifications, test cases and results, design specifications, diagrams and reports. This setup provides a place for teams and managers to keep track of what has been accomplished and what still needs to be done. This information is often displayed graphically so users can quickly find what they need, as well as get a quick overview of the project.

## History of CASE tools and criticism

CASE software tools emerged as a significant product category in the 1980s. They were developed in a response to a need to bring order to large software development projects, and vendors claimed they would improve IT productivity and reduce errors.

The U.S. government, a major builder of custom development projects, spent millions on CASE tools. But the government later became a critic of vendor claims about their capabilities. "Little evidence yet exists that CASE tools can improve software quality or productivity," wrote the Government Accountability Office in a 1993 report on the use of CASE tools by the U.S. Defense Department.

About a decade later, in 2002, a research paper also noted problems with CASE deployments. It found evidence of "a conceptual gap between the software engineers who develop CASE tools and the software engineers who use them," according to the paper, "Empirical Study of Software Developers' Experiences," by Ahmed Seffah and Rex B. Kline, computer science researchers at Concordia University.

## Uses of CASE tools evolved

CASE tools have evolved to accommodate visual programming, object-oriented programming and Agile software development processes.

Tools that fit in the CASE category are widely available, but this umbrella term or approach doesn't have the relevance it once did in describing software engineering tools. Developers may be more likely to think in terms of specific tool categories, such as visual modeling and simulation software, system architecture tools and diagramming tools such as Microsoft Visio.

The problem that CASE technology attempted to fix remains. The Standish Group, according to its data analysis, says that large software projects, such as enterprise resource planning (ERP) implementations that cost more than $10 million, face a failure rate as high as 41%. The complexity of software development is a continuing challenge.

## Advantages of the CASE approach:

- As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.
- The overall quality of the product is improved as an organized approach is undertaken during the process of development.
- Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.
- CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

## Quality Is Free

Quality is free? Impossible? Nope, it's true. In 1979, Philip Crosby[1] wrote in his book *Quality is Free: The Art of Making Quality Certain*, that indeed it costs nothing extra (actually it costs less) to produce something of high quality versus something of low quality. Given what you've learned so far about software testing and the work involved in finding and fixing bugs, this may seem impossible, but it's not.

Think back to the graph from Chapter 1, "Software Testing Background," (repeated here as Figure 21.1) that showed the cost of finding and fixing bugs over time. The later bugs are found, the more they cost—not just linearly more, but exponentially more.

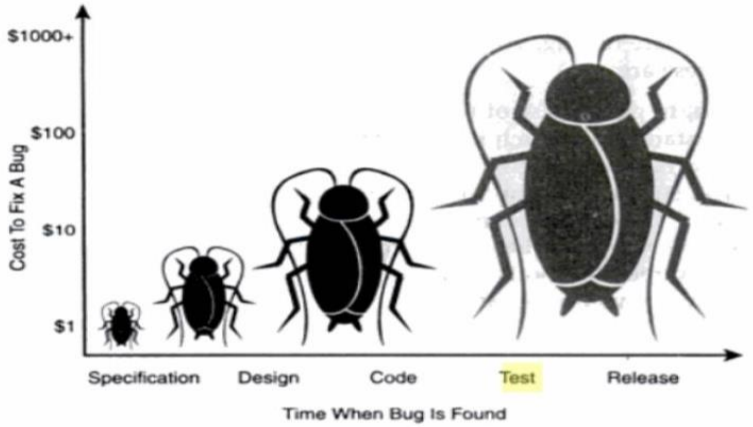



***FIGURE 21.1*** There is very little cost if problems are found early in the project.

Now, divide the cost of quality into two categories: the *costs of conformance* and the *costs of nonconformance*. The costs of conformance are all the costs associated with planning and running tests just one time, to make sure that the software does what it's intended to do. If bugs are found and you must spend time isolating, reporting, and regression testing them to assure that they're fixed, the costs of nonconformance go up. These costs, because they are found before the product is released, are classified as *internal failures* and fall mostly on the left side of Figure 21.1.

*1 Philip Crosby, Joseph Juran, and W. Edwards Deming are considered by many to be the "fathers of quality." They've written numerous books on quality assurance and their practices are in use throughout the world. Although their writings aren't specifically about software, their concepts, often in-your-face common sense, are appropriate to all fields. Good reading.*

If bugs are missed and make it through to the customers, the result will be costly product support calls, possibly fixing, retesting, and releasing the software, and—in a worst-case scenario—a product recall or lawsuits. The costs to address these *external failures* fall under the costs of nonconformance and are the ones on the right side of Figure 21.1.

In his book, Crosby demonstrates that the costs of conformance plus the costs of nonconformance due to internal failures is less than the costs of nonconformance due to external failures. Stomp out your bugs early, or ideally don't have any in the first place, and your product will cost less than it would otherwise. Quality is free. It's common sense.

Unfortunately, portions of the software industry have been slow to adopt this simple philosophy. A project will often start with good intentions and then as problems crop up and schedule dates are missed, rules and reason go out the window. Regard for higher future costs is written off in favor of getting the job done today. The trend is turning, however. Companies are now realizing that their cost of quality is high, and that it doesn't need to be. Customers are demanding and their competitors are creating better quality software. Realization is setting in that the words Crosby wrote more than 25 years ago for the manufacturing industry apply just as well to the software industry today.

# Testing and Quality Assurance in the Workplace

Depending on the company you work for and the project you're working on, you and your peers can have one of several common names that describes your group's function: Software Testing, Software Quality Assurance, Software Quality Control, Software Verification and Validation, Software Integration and Test, or one of many others. Frequently these names are used interchangeably or one is chosen over the others because it sounds more "official"—Software Quality Assurance Engineer versus Software Tester, for example. It's important to realize, though, that these names have deeper meanings and aren't necessarily plug-in replacements for each other. On one hand there's the philosophy that "it's only a name," that what you ultimately do in your job is what counts. On the other hand, your job title or your group's name is what others on the project team see. That label indicates to them how they will work with you and what expectations they will have, what deliverables you will provide to them, and what they will give to you. The following sections define a few of the common software-test-group names and should help clarify the differences among them.

## Other Names for Software Testing Groups

Depending on where you work, your test group may use one of many other names to identify itself. Software Quality Control (SQC) is one that's frequently used. This name stems from the manufacturing industry where QC inspectors sample products taken off the manufacturing line, test them, and, if they fail, have the authority to shut down the line or the entire factory. Few, if any, software test groups have this authority—even ones that call themselves Software QC.

Software Verification and Validation is also commonly used to describe a software test organization. This name is one that actually works pretty well. Although it's a bit wordy, it states exactly what the test group is responsible for and what they do. Look back to Chapter 3 for the definitions of verification and validation. It's even possible to have two groups, one for verification and one for validation.

Integration and Test, Build and Test, Configuration Management and Test, Test and Lab Management, and other compound unrelated names are often a sign of a problem. Many times the software test group takes on roles (voluntarily or not) that are unrelated to testing. For example, it's not uncommon for a test group to own the job of configuration management or building the product. The problem with this is twofold:

- It takes away resources that should be used for testing the product.
- The test group's goal is ultimately to break things, not to make things, and owning the software's build process creates a conflict of interest.

It's best to let the programmers or a separate team build the software. Testing should concentrate on finding bugs.

# Your job as a Software Tester

Software Testing is a process of verifying a computer system/program to decide whether it meets the specified requirements and produces the desired results. As a result, you identify bugs in software product/project.

Software Testing is indispensable to provide a quality product without any bug or issue.

## Skills required to become a Software Tester

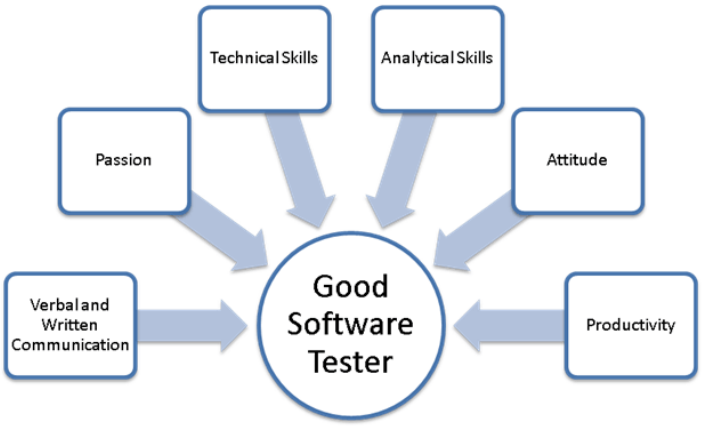
We will discuss the Technical and Non-Technical required to become a Software Tester

### Non-Technical Skills

Following skills are essential to becoming a good software tester. Compare your skill set against the following checklist to determine whether Software Testing is a reality for you-

- **Analytical skills**: A good software tester should have sharp analytical skills. Analytical skills will help break up a complex software system into smaller units to gain a better understanding and create test cases. Not sure that you have good analytical skills - Refer this link - if, if you can solve at least ONE problem you have excellent analytical skills.
- **Communication skill**: A good software tester must have good verbal and written communication skill. Testing artifacts (like test cases/plans, test strategies, bug reports, etc.) created by the software tester should be easy to read and comprehend. Dealing with developers (in the event of bugs or any other issue) will require a shade of discreetness and diplomacy.
- **Time Management & Organization Skills:** Testing at times could be a demanding job especially during the release of code. A software tester must efficiently manage workload, have high productivity, exhibit optimal time management, and organization skills
- **GREAT Attitude:** To be a good software tester you must have a GREAT attitude. An attitude to 'test to break', detail orientation, willingness to learn and suggest process improvements. In the software industry, technologies evolve with an overwhelming speed, and a good software tester should upgrade his/her technical skills with the changing technologies. Your attitude must reflect a certain degree of independence where you take ownership of the task allocated and complete it without much direct supervision.
- **Passion:** To Excel in any profession or job, one must have a significant degree of the passion for it. A software tester must have a passion for his / her field. BUT how do you determine whether you have a passion for software testing if you have never tested

before? Simple TRY it out and if software testing does not excite you switch to something else that holds your interest.




## Technical Skills

This list is long, so please bear with us

- **Basic knowledge of Database/ SQL:** Software Systems have a large amount of data in the background. This data is stored in different types of databases like Oracle, MySQL, etc. in the backend. So, there will be situations when this data needs to be validated. In that case, simple/complex SQL queries can be used to check whether proper data is stored in the backend databases.
- **Basic knowledge of Linux commands:** Most of the software applications like Web-Services, Databases, Application Servers are deployed on Linux machines.So it is crucial for testers to have knowledge about Linux commands.
- **Knowledge and hands-on experience of a Test Management Tool:**Test Management is an important aspect of Software testing. Without proper test management techniques, software testing process will fail. Test management is nothing but managing your testing related artifacts.

  For example - A tool like Testlink can be used for tracking all the test cases written by your team.

  There are other tools available that can be utilized for Test Management. So, it is important to have knowledge and working experience of such tools because they are used in most of the companies.

- **Knowledge and hands-on experience of any Defect Tracking tool-** Defect Tracking and Defect life cycle are key aspects of software testing. It is extremely critical to managing defects properly and track them in a systematic manner. Defect tracking becomes necessary because the entire team should know about

the defect including managers, developers, and testers. Several tools are used to lock defects including QC, Bugzilla, Jira, etc.

- **Knowledge and hands-on experience of Automation tool:** If you see yourself as an "Automation tester" after a couple of years working on manual testing, then you must master a tool and get in-depth, hands-on knowledge of automation tools.

  **Note** - Only knowledge of any Automation tool is not sufficient to crack the interview, you must have good hands-on experience, so practice the tool of your choice to achieve mastery. Knowledge of any scripting language like VBScript, JavaScript, C# is always helpful as a tester if you are looking for a job into automation. Few companies also use Shell/Perl scripting, and there is a lot of demand for testers having knowledge of the same. Again, it will depend on the company and which tools are used by that company.

There is also a lot of scope for performance testing tools because applications need to be tested for their performance which is a part of non-functional testing.

That's it to technical knowledge. Please note you do not need ALL the technical skills listed above. The technical skill sets required vary with the Job Role and company processes.

## Academic Background

Academic background of a software tester should be in Computer Science.

A BTech/ B.E., MCA, BCA, BSc- Computers, will land you a job quickly.

If you do not hold any of these degrees, then you must complete a software testing certification like ISTQB and CSTE which help you learn Software Development/ Test Life Cycle and other testing methodologies.

## Remuneration

Compensation of a software tester varies from company to company. Average salary range of a software tester in the US is $45,993 - $74,935. Average salary range of a software tester in India is Rs 247,315 - Rs 449,111.

Also, a software tester is also given health insurance, bonuses, gratuity and other perks.

## What Does a Software Tester do?

On any typical work day, you will be busy understanding requirement documents, creating test cases, executing test cases, reporting and re-testing bugs, attending review meetings and other team building activities.

## Software Tester Career Path

Your career progression as a software tester (QA Analyst) in typical CMMI level 5 company will look like following but will vary from company to company

1. QA Analyst (Fresher)
2. Sr. QA Analyst (2-3 years' experience)
3. QA Team Coordinator (5-6 years' experience)
4. Test Manager (8-11 years' experience)
5. Senior Test Manager (14+ experience)

## Alternate Career Tracks as a Software Tester

Once you have got your hand dirty in manual testing, you can pursue following specializations

- **Automation Testing**: As an Automation Test Engineer, you will be responsible for automating manual test case execution which otherwise could be time-consuming. Tools used IBM Rational Robot, Silk performer, and QTP
- **Performance Testing:** As a performance test engineer, you will be responsible for checking application responsiveness (time is taken to load, maximum load application can handle), etc. Tools used WEBLoad, Loadrunner.
- **Business Analyst**: A major advantages Testers have over Developers is that they have an end to end business knowledge. An obvious career progression for testers is to become a Business Analyst. As a Business Analyst, you will be responsible for analyzing and assessing your company's business model and workflows. As a BA, you will intergrate these models and workflows with technology.

## Common Myths

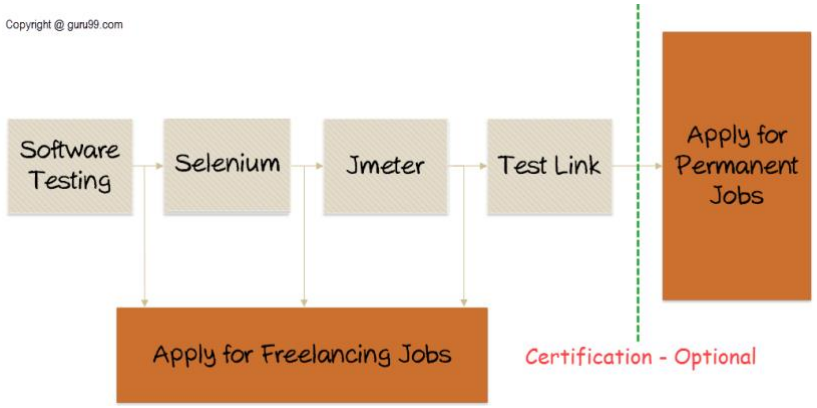
Software Testing as a Career pays Less Developers are more respected as compared to Testers

Contrary to popular belief, Software Testers (better known as QA professionals) are paid and treated at par with Software Developers in all "aspiring" companies. A career in Software Testing should never be considered as "second rated."

Software Testing could actually "test" your nerves since you need to make sense of Business Requirements and draft test cases based on your understanding. Software testing is not boring. What is boring is doing the same set of tasks repeatedly. The key is to try new things. For that matter, have you ever spoken to a software developer with more than 3 years' experience? He will tell you how boring his job has become off-lately.

## How to Become Software Tester

For a complete newbie, here is our suggested approach to learning Software Testing




You start with learning Basic principles of Software Testing. Once done you **apply for freelancing jobs. This will help you gain practical knowledge and will fortify the testing concepts you have learned.**

Next, you proceed to Selenium - Automation tool, then JMeter - Performance Testing tool and finally TestLink - Test Management Tool. All the while you are learning, we suggest you apply for freelancing jobs (apart from other benefits you will make some moolah too!).

Once you are through with all the tools, you may consider taking a certification. We recommend ISTQB. However, this is optional.

## Certification Exams:

ISTQB Foundation level is the basic certification in Testing field.

It is not mandatory, but it will help increase your chances of getting the job. Most of the companies have this criterion.

A software tester with ISTQB cleared will be given more priority as compared to others.

After this, when you apply for permanent jobs in big corporations you will have many skills to offer as well some practical freelancing experience which may be of value and will increase your chances of being selected.

You can also pursue certification in a Testing tool of your choice.

## Professional Organizations Dedicated to Software or Software Quality

Several professional nonprofit organizations are dedicated to software, software testing, and software quality assurance that may be of interest to you. Their websites provide details on their specific area of coverage:

- The Association for Software Testing (AST) at www.associationforsoftware-testing.org is a nonprofit professional service organization dedicated to advancing the understanding and practice of software testing. They hold an annual conference and frequent workshops. Their archives make available a wealth of software testing information—emphasizing practice over theory.
- The American Society for Quality (ASQ) at www.asq.org publishes journals and articles on quality and administers the Certified Quality Engineer (CQE) and the Certified Software Quality Engineer (CSQE) designation.
- The Association of Computing Machinery (ACM) at www.acm.org and its Special Interest Group on Software Engineering (SIGSOFT) at www.acm.org/sigsoft has more than 80,000 members in educational and scientific computing.
- The Society for Software Quality (SSQ) at www.ssq.org lists its vision as "To be recognized as the Society for those interested in promoting 'quality' as a universal goal for software."