UNIT-1

SYSTEM ANALYSIS & DESIGN AND SOFTWARE ENGINEERING

- Definitions:-
  (System, SubSystem, Business System, Information System (Defination only)

- System Analyst
  (Role: Information Analyst, System Designer & Programmer Analyst)

- SDLC

- Fact-Finding techniques
  (Interview, Questionnaires, Record review & observation)

- Tools for documenting procedures and decidions
  (Decision tree & Decision table)

- Data Folw Analysis Tool
  (DFD (context and zero level) and data dictionary)

- Software Engineering
  (Brief introduction)

Definitions:

→ **System: -**

- A system is simply a **set of components that interact with each other** to accomplish some purpose or a particular goal.

→ **Subsystem: -**

- A business is also a system. Its components are marketing, manufacturing, sales, research, shipping, accounting and personnel – **all work together to create a profit that benefits the employees and stock holders of the firm. Each of these components is itself a system and it is called *subsystem*.**

→ **Business System: -**

- A business is also a system. Which is having a set of components that interact with each other to **fulfill the business goals**? And fulfill the business needs.

→ **Information System: -**

- **Every business system depends on a more or less abstract entity called an information system**. This system is the means by which data flow from one person/department to another. Information system helps all the system of business, linking the different component in such a way that they effectively work towards the same purpose.
- The purpose of information systems are to process input, maintain data and produces information, reports and other output.

➢ **System Analyst:**

- System analyst is a person who is **responsible to fulfill the needs of organization and provide the information.**
- A system analyst's primary responsibility is to **identify information needs of an organization and obtain a logical design of an information system** which will meet these needs.
- **Three groups of people are involved in developing information systems for organization managers, users of the systems and computer programmers.** The efforts of the system analyst is to co-ordinates all these group, to effectively develop and operate computer based information system, some important function of system analyst can be expressed as follow:

  Defining requirement
  Categorized the requirements and determines the priority.
  Gathering data, facts and opinions of facts.
  Analysis and evaluation
  Solving up specifications
  Designing system
  Evaluating syste

- **Role:-**
  - Information analyst: - In this role the **analyst is responsible to find the information and together the information to fulfill the requirement** of the organization.
  - System Analysis & Design:- System analysis and design refers to the process of organizing situation with the intent of improving it through better procedures & methods.
  - System developments have two major components:

- system analysis
- system design

System design is the process of planning a new business system or one to replace an exiting system.
System analysis is the process of gathering & diagnosing problems & using information to recommends important to the system.

- **SUMMARY : -**
  - System analyst is a person who is responsible to fulfill the needs of organization and provide the information.
  - Three groups of people are involved in developing information systems for organization managers, users of the systems and computer programmers.
- **Role:-**
  - Information analyst :-
  - System Analysis & Design:-

- **SDLC:**

SDLC is stand for **system development life cycle**. It is a **cyclic process to develop the system for any organization**. It consist the following steps to process:

(1) Preliminary investigation
(2) System analysis
(3) System design
(4) Coding
(5) Testing
(6) Implementation
(7) Maintenance

SDLC is a classical thought of as the set of activities that analysts, designers, and users carry out to develop and implement an information system. SDLC consist of the following six activities :

(1) Preliminary investigation :

A request is made by a manager, an employee or a system specialist for information system. From this point the first system activity, the preliminary investigation starts. It consists of three parts

- **Request clarification :** The users or the persons who wants the information system, their request are not clear, therefore before any system investigation can be consider, the project request must be examined to determine precisely what the originator wants.

- **Feasibility study :** When ever any user request is clarified, then it is very much important to determine whether the system request is feasible or not. There are three aspects in the feasibility study i.e.

  Technical feasibility : Can the work for the project be doe with current equipment, existing software technology and available person?

  Economic feasibility : Are there sufficient benefits in creating the system to make the cost acceptable?

  Operational feasibility : will the system be used if it is developed and implemented?

The feasibility study is carried out by a small group of people who are familiar with information system techniques as well as the routine and detail activities of the organization.

- **Request approval :** All requested projects are not desirable or feasible. However, those projects that are both feasible and desirable should be put into a schedule. If the systems developer are free then the development process will be immediately started otherwise, the proposal will be put into the priority queue depending upon the important.

(2) System Analysis (Determination of system requirements):

The detailed understanding of all **important facts of the business area under investigation is the key point or heart of the system analysis.** The analyst must study the business process, so that the questions related to study, can be answered. For these the system analyst has to work with variety of persons to gather details about the business process.

As the details are gathered, the analyst study the requirement data to identify features the new system should have.

(3) System Design:

The design of information system produces the details that state how a system will meet the requirements identified during systems analysis. **Some times this stage is called logical design.** In controls to the process of development program software, which is referred to as physical design?

System analyst begins the design process by identifying reports and other outputs, usually designer sketch it to appear when the system is complete. This may be done on paper.

The system design **also describes the data to be input, calculated or stored.** Individual data items and calculation procedure are written in detail.

The detailed design information is passed on to the programmers with complete and clearly outlined software specifications. **As programming starts, designers are available to answer questions, clarify fuzzy areas & handle problems that confront the programmers when using the design specifications.**

(4) Coding (Development of software):

Software developers may install purchased software or they may write new, custom designed programs. The choice depends on the cost of each option, the time available to write software and the availability of programmers.

Programmers are also responsible for documenting the program, providing an explanation of how and why certain procedures are coded in specific ways.

(5) System testing:

In testing, **the system is used experimentally to ensure that the software does not fail.** I.e. that it will run according to its specifications and in the way users expect. Special test data are inputted for processing, and the result examined. **A limited number of users may be allowed to use the system so analyst can see whether they try to use it in unforeseen ways.**

In many organizations testing is performed by persons other than those who wrote the original programs to ensure more complete and unbiased testing and more reliable software.

(6) Implementation & evaluation:

Implementation is the process of having systems personnel check out and put new equipment into use, train users, install he new applications and construct any files of data needed to use it.

Evaluation of the system is perform to identify its strengths and weakness. The evaluation process can be categorized in following three way :

i. **Operational evaluation :** In this, it will determine how system is functioning, it also includes ease of use, response time, suitability of information formats, overall reliability and level of utilization.
ii. **Organizational impact :** Identification and measurement of benefits to the organization in such area as financial concerns (cost, revenue and profit), operational efficiency and competitive impact.

iii. **User manager assessment :** Evaluation of the attitudes of senior and user managers within the organization, as well as end users.

iv. **Development performance :** It measure overall development time and effort, conformance to budgets and standards, and other project management criteria, includes assessment of development methods and tools.

(7) Maintenance:-

Maintenance follows a CBIS. As users develop faith in a CBIS, their demands on the system will grow. The system design should be flexible enough to accommodate future requests; refinements, modifications and changes to suit users' requirements. Well documented logical and physical designs of a CBIS will facility its maintenance considerably.

- **SUMAMRY:-**
  - SDLC is stand for **system development life cycle**. It is a **cyclic process to develop the system for any organization**.
  - It consist the following steps to process:
    - Preliminary investigation
    - System analysis
    - System design
    - Coding
    - Testing
    - Implementation
    - Maintenance
  - SDLC is a classical thought of as the set of activities that analysts, designers, and users carry out to develop and implement an information system.

- **FACT FINDING TECHNIQUES**

What is fact finding Techniques?

The specific methods analysts use for collecting data about requirement are called fact-finding techniques. It includes:

(1) Interview
(2) Questionnaire
(3) Record Review
(4) Observation

(1) Interview:-

The analyst use interviews **to collect information from individuals or from Groups**. The respondents are generally current users of the existing system or managers.

The interview can either be structured or unstructured type.

Structured Interview

Ensures uniform wording of questions for all respondents.
Results in shorter interviews
Easy to administer and evaluate.
Limited interviewer training is needed.
Cost of preparation is high.
Respondent may not accept high level of structure and mechanical posing of question.

Unstructured Interview

Interviewer has greater flexibility in wording questions to suit respondent. May produce information about the areas that were overlooked or not thought to be important.

Takes extra time to collect essential facts.

Analysis and interpretation of results may be lengthy.

The success of an interview depends on the skill of the interviewer and on his or her preparation for the interview.

(2) Questionnaire

The use of Questionnaireallows **analyst to collect information about various aspects of a system from large number of a system from a large number of persons.** The use of standardized question formats can yield more reliable data than other fact-finding techniques.

Questionnaires should **also be tested and if necessary, modified before being printed and distributed.** The analyst should ensure that the respondents' background and experiences qualify them to answer the question.

(3)Record Review

Analysts examine the information that has been recorded about the system and users. Record inspection can be performed at the beginning of the study, as an introduction, or later in the study, as a basis for comparing actual operations with what the records indicate should be happening.

It includes written policy manuals, regulations and standard operating procedures used by most organizations as a guide for managers and employees.

(4)Observation

Through observation, **analyst can obtain firsthand information about how activities are carried.** This method is useful when analyst need to actually observe how documents are handled, how processes are carried out and whether specified steps are actually followed. **Experience observer knows what to look for and how to assess the significance of what they observe.**

- **SUMMARY : -**
  - The specific methods analysts use for collecting data about requirement are called fact-finding techniques. It includes:
  - **Interview**
  - **Questionnaire**
  - **Record Review**
  - **Observation**

- **Tools for documenting procedure and decisions**

A tool is any device, object or operations used to accomplish a specific task. System analyst rely on such type of tools. There tools help analyst in so many different ways (i.e. To collect data, present data, explain processes etc.) To explain the procedures or documenting the procedures there are three tools :

Decision tree

Decision table

When analyst starts the study of any information system, the first question is about, what are possibilities? or what can happen? Means he/she is asking about the condition to take any appropriate action. In real situation the problem is not same, hence the conditions vary for different problems and different situations, so some time it is referred as decision variable.

When all possible conditions are known, the analyst next determines what to do when certain condition occurs. Actions are alternatives, the steps, activities or procedures that an individual may decide to take when confronted with a set of conditions. The actions will be simple or it may be complex in different situation.

- **Decision tree:**

  A single matter can be explained in so many different ways, for example, a company might give discount amount on three different values for the condition on size of order (i.e. over 10,000 – 4 %, in between 5000 to 10000 – 3 % and below 5000 - 2 %) and the payment occurs within 10 days or not. The same process can be explained in following different ways.

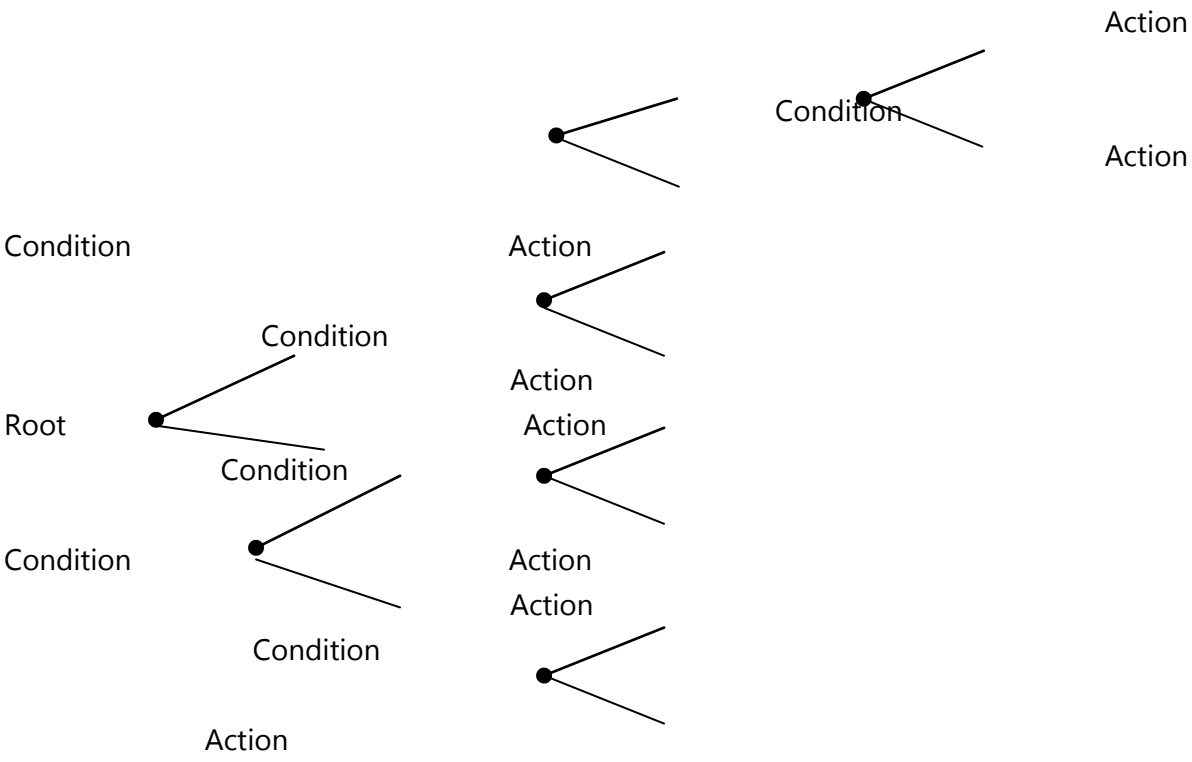
  Greater than 10,000. greater than or equal to 5000 but less than or equal to 10,000 and below 5000.

  Not less than 10,000, not more than 10,000 but at least 5000 and not 5000 or more.

  Having different ways of saying the same thing can create difficulties in communications during system study.

  Decision tree is one of the method for describing decisions, while avoiding difficulties in communications.

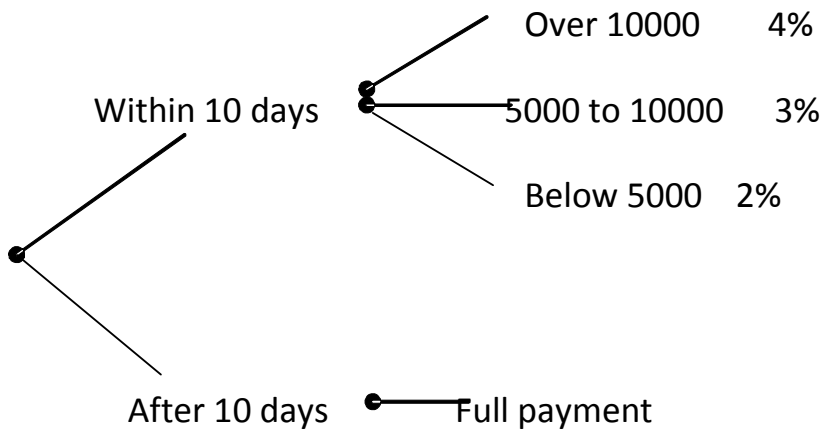
  A Decision tree is diagram that presents conditions and actins sequentially and thus shows which conditions to consider first, which second and so on. It is also a method of showing the relationship of each condition and its permissible actions. The diagram resembles branches on a tree.



- **Decision tree**

The root of the tree, on the left of the diagram, is the storing point of the decision sequence. The particular branch to be followed depends on the conditions that exists and the decision to be made. Progression from left to right in any branch will give the sequence of decision. One decision point will lead to another decision point. The nodes of the three thus represents conditions and indicates that a determination must be made about which condition exists before the next path can be chosen. The right side of the tree lists the actions to be taken, depending upon the sequence of condition that is followed.

Developing decision tree is very much beneficial i.e. The need to describe conditions and actions forces analyst to formally identify the actual decision that must be made. It also force analyst to consider the sequence of condition.



Decision trees may not always be the best tools for decision analysis. A decision tree for a complex system with many sequence of steps and combination of conditions will be unwieldy. A large number of branches with many paths through them will could rather than aid analysis. When these problem arise, decision table should be considered.

- **Decision table**

A decision table is a matrix of rows and columns, rather than a tree, thatshows conditions and actions. Decision rules, included in a decision table, state what procedure to follow when certain conditions exists.

The decision table is made up of four sections: Condition statements, condition entry, action statements, and actions entries. The condition statement **identifies relevant conditions**. Condition entries **tell which value**, if any applies for a particular condition. Action statements list the **set of all steps that can be taken when a certain condition occurs**. Action entries **show what specific actions in the set to take when selected conditions or combinations of conditions ate true.** Sometimes notes are added below the table of indicate when to use the table or to distinguish it from other decision tables.

Condition	Decision rules
Condition statements	Condition entry
Action statements	Action entry

The columns on the right side of the table, linking conditions and actions; form decision rules, which state the conditions that must be satisfied for a particular set of actions to be taken.

Condition	Decision rules			
C1 : Patient has health insurance	Y	Y	N	N
C2 : Patient has social health ins.	Y	N	Y	N
A1 : Pay only visit charge			X	
A2 : Pay nothing	X		X	
A3 : full payment				X

Table T – 1

The above decision table describes action taking in payment to a doctor. There are two types of insurance

Health insurance (Condition – I)

Social health insurance (Condition – II)

If the patient has only health insurance he/she has to pay visit charge, if patient has only social or both type of insurance, he/she has to pay nothing. If the patient does not have any insurance, he/she has to pay full payment.

The above matter is stated in decision table. There are two conditions statements and corresponding four condition entries, with three actions statements and corresponding action entries.

The payment discount (Discuss in decision tree) can also be described using decision table as follow:

	X	X	X		X			X	X	X	X		X			X
Within 10 days	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
> 10,000	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
5000 to 10000	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Below 5000	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
4 % Discount																
3 % discount																
2% discount																
Full payment																

Table T – 2

In the above table 'X' indicates the contradiction entries, so it must be removed from the table, hence the table will be as follow:

Condition	Decision rules					
Within 10 days	Y	Y	Y	N	N	N
> 10,000	Y	N	N	Y	N	N
5000 to 10000	N	Y	N	N	Y	N
Below 5000	N	N	Y	N	N	Y
4 % Discount	X					
3 % discount		X				
2% discount			X			
Full payment				X	X	X

Building decision table : To develop decision table analyst should use the following steps :

- Identifies the conditions in the decision. Each condition selected should have the potential to either occur or nor occur, partial occurrences is not possible.
- Determine the actions.
- Study the combinations of conditions that are possible. For N conditions there are 2n combinations.
- Fill in the table with decision rules.
- Mark the action entries with X to signal action to take, leave a cell blank for no action applies.
- Examine the table for redundant rules or for contradictions within rules.

  After constructing a table, analyst verifies it for correctness and completeness to ensure that the table includes all the conditions. Along with the decision rules that relate them to the actions. Analyst should also examine the table for redundancy and contradictions.

Eliminating redundancy :

Decision table can become too large and unwieldy if allowed to grow in an uncontrolled fashion. Removing redundant entries can help manage table size. Redundancy occurs when both of the following are true

Two decision rules are identical except for one condition row.

The action for the two rules is identical.

In Table T – 1 the decision rules 1 and 3. for both action entry is same. Here action entry is not dependent on condition –1 entry, hence these two rules are redundant and can combined into one rule. The condition row where they differ can be replaced by a – as shown in the following table :

Condition	**Decision rules**		
C1 : Patient has health insurance	-	Y	N
C2 : Patient has social health ins.	Y	N	N
A1 : Pay only visit charge			X
A2 : Pay nothing	X		
A3 : full payment			X

Removing contradictions:

Decision rules contradict each other when tw0 or more rules have the same set of conditions and the actions are different.

Contradictions mean either that the analyst's information is incorrect or that there is an error in the construction of the table. In table T-2 many contradictory rules are shown with 'X' on the top.

The usefulness of decision table processors is in saving programming time and checking for errors.

- **SUMMARY:-**
  - **A tool is any device, object or operations used to accomplish a specific task.** System analyst rely on such type of tools. There tools help analyst in so many different ways (i.e. To collect data, present data, explain processes etc.)
  - **Decision tree**
    - Progression from left to right in any branch will give the sequence of decision. One decision point will lead to another decision point. The nodes of the three thus represents conditions and indicates that a determination must be made about which condition exists before the next path can be chosen.
  - **Decision table**
    - **A decision table is a matrix of rows and columns, rather than a tree, that shows conditions and actions.** Decision rules, included in a decision table, state what procedure to follow when certain conditions exists.

DATA FLOW ANALYSIS TOOL

What is data flow analysis?

While developing a system, analyst wants to know the answer to four specificquestions:

What processes make up a system?
What data are used in each process?
What data are stored?
What data enters and leaves the system?

Data drives the business activities. Systems analyst recognizes the central role of business data in organization. Data flow analysis study the use of data in each activity (i.e. process, stores, retrieved, used, changed & output).

Data Flow Diagram (DFD) graphically shows the relation between processes and data. Data dictionary, which formally describes the system data and where they are used.

Data flow diagram (DFD)

A graphical tool used to describe and analyze the movement of data through a system including the processes, stores of data and delays in the system. DFD are the central tool and the basis from which other components are developed. **The transformation of data from input to output, through process, may be described logically and independently of the physical**

components are called logical DFD. In contrast, physical DFD show the actual implementation and the movement of data between people, departments and workstations.

Logical DFD can be completed using only four simple notations. The symbols are developed by two different organizations (i.e. Yourdon and Gane & Sarson). The symbols are as follow

Data Flow :

It shows the direction of data flow, from an origin to a destination in the form of document, letter, telephone call.

Yourdon Gane & Sarson

Processes :

People, procedures or device that use or produce data.

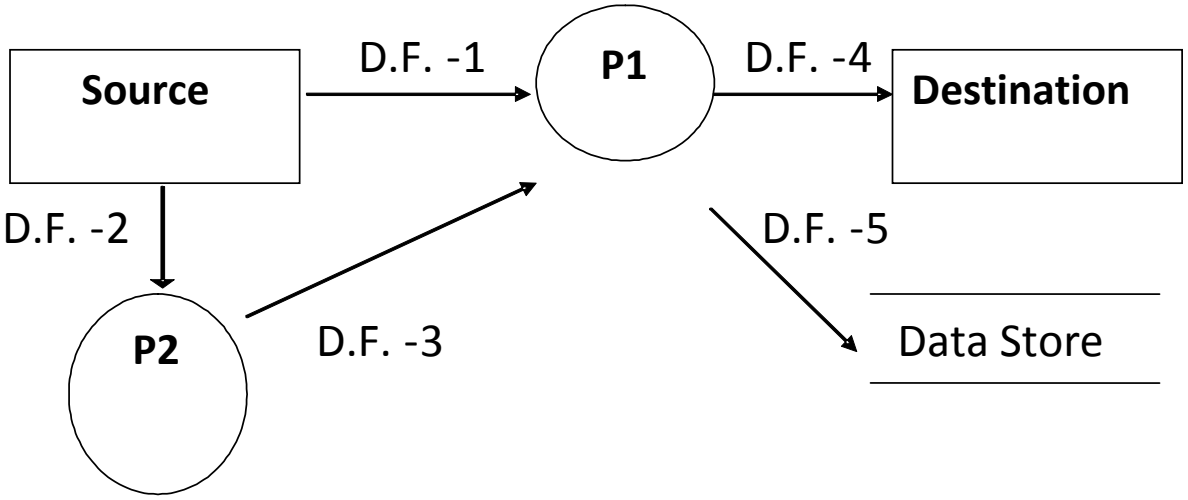
Source & Destination :

External sources or destination of data, which may be people, programs, organizations or other entities interact with the system bur are outside its boundary. The term source or sink are interchangeably used with origin & destination.

Data store :

Here data are stored or referenced by a process in the system.

Each component in a DFD is labeled with a descriptive name. Process name are identified with a number. The umber assigned to a specific process does not represent the sequence of process. It is strictly for identification and will take on added value when we study the components that, make up a specific process.



Several data flow can be going on simultaneously. In above DFD d. flow – 1 & d; flow – 2 may occur I parallel.

As the name suggest, DFD concentrate on the data moving through the system, not on device or equipment. Analyst explain why the data are being input or output and what processing s done. It is just as important to determine when data enter the application area and when they leave. Sometimes data are stored for later use or retrieved from previous storage. DFD also show this.

Developing Data Flow Diagram :

System analyst must first study the current system, that is, the actual activities and processes that occur. In the terminology of structured analysis, this is a study of the physical system.

The physical system is translated into a logical description that focus on data and processes. It emphasize data and processes in order to focus on actual activities that occur and the resources needed to perform them, rather than on who performs the work.

Data flow diagrams are of two type :

Physical data flow diagram :

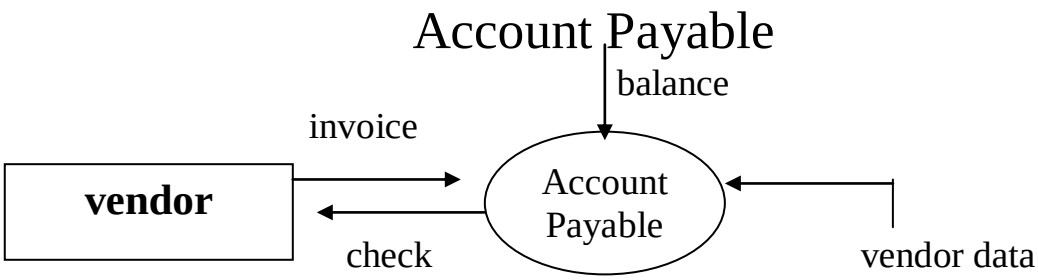
It is an implementation dependent view of the current system, showing what task are carried out and how they are carried out and how they performed. Its characteristics includes (name of people, form and document names or numbers, names of dept, master & transaction file, equipment & device used, locations, name of procedures etc.)

Logical data flow diagram :

It is an implementation independent view of a system, focusing on the flow of the data between processes without any concern of specific devices, storage location or people in the system.

Drawing context diagram :

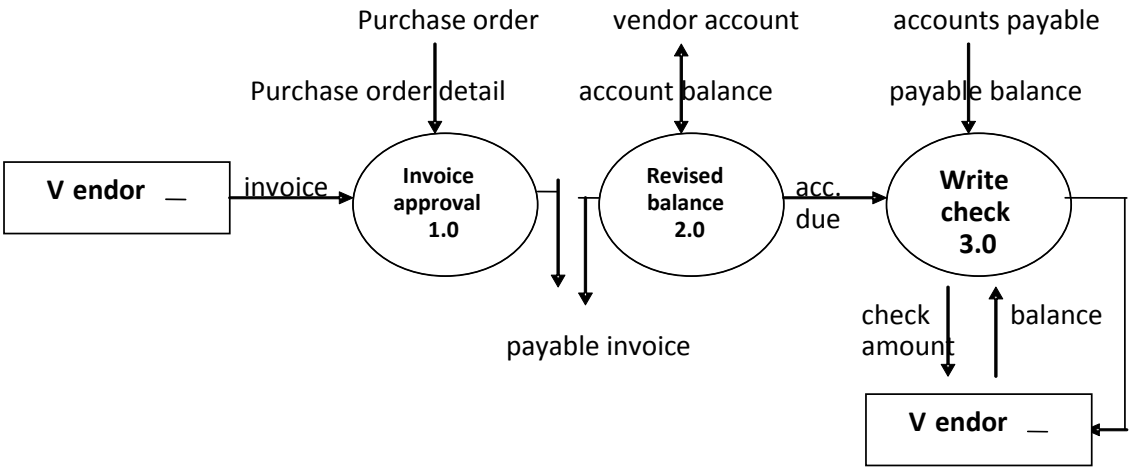
The first steps in requirement analysis is to learn about the general characteristics of the business process under investigation. The data flow diagram describes account payable processing at a very general or top level.



This diagram shows that vendors submit invoice and receives checks from organization. This accounts payable process requires accounts payable and vendor data. In the figure each arrow represent the data flow, is labeled to show what data are being used. Balance data are retrieved from the accounts payable data store and vendor address is retrieved from the vendor data store.

This diagram often called context diagram. It contains a single process, but it plays a very important role in studying the current system.

- **Developing the second level diagram :**



The description of the accounts payable system in context diagram requires more detail. The Fig – II represent 2nd level diagram. In this, three process are explained i.e. invoice approval, revise balance due and write vendor check.

In invoice approval, the invoice using the purchase order store and all invoices are approved or rejected. The approved invoices are stored in an invoice data store.

In revised balance due, the payable invoice (i.e. the out come of the invoice approval) is scrutinized with vendors account, and vendors accounts will be updated and the balance will be stored and accounts due will be return for the next process.

In write vendor check, the account due amount is checked using account payable and then check will be prepared.

The check is sent to the vendor.

While drawing second level diagram the following point should keep in mind :

All data flow that appeared on the previous diagram explaining the processes are included in the lower level diagram.

New data flows and data stores are added if they are used.

No entries should contradict the description of the higher level DFD.

General rules for drawing logical DFD

Any data flow leaving a process must be based on data that are input to the process. All data flow are named, the name reflects the data flowing between processes, data stores, sources, or sinks.Consider only those data which are needed in process.

- **SUMMARY:-**
  - A graphical tool used to describe and analyze the movement of data through a system including the processes, stores of data and delays in the system.
  - DFD are the central tool and the basis from which other components are developed. The transformation of data from input to output, through process, may be described logically and independently of the physical components are called logical DFD.
  - Logical DFD can be completed using only four simple notations. The symbols are developed by two different organizations (i.e. Yourdon and Gane&Sarson).
  - **General rules for drawing logical DFD**
    - Any data flow leaving a process must be based on data that are input to the process. All data flow are named, the name reflects the data flowing between processes, data stores, sources, or sinks.Consider only those data which are needed in process.

- **Data dictionary:-**

  When the volume of data is very large, it is very much difficult for analyst to manage the data definitions. If the information system is very big, then more than one person are working on the same data, at that time, any data defined by any person, can be used by the other person, hence they need the definition or description of the data.

  Data dictionaries are an **integral component of structured analysis; it provides additional information about the system.**

  A data dictionary **is a catalog – a repository – of the elements in a system**. As the name suggest, these elements center around data and the way they are structured to meet user requirements and organization needs. It contains a list of all the elements composing the data flowing through a system. **The major elements are data flows, data stores and processes.** The data dictionary stores details and descriptions of these elements. If data dictionary is developed properly, then any data related questions – answer can be extracted from data dictionary.

  It is developed during data flow analysis and assists the analysts. The stored details are used during system design.

Importance of Data dictionary:

To manage the detail in large system: Large system have huge volumes of data flowing through them in the form of documents, reports and even conversations. The analyst should remember all the definition for letter use, the best organized and most effective analyst use automated data dictionary designed specifically for systems analysis and design.

To communicate a common meaning for all system elements: Data dictionary assists in ensuring common meanings for system elements and activities. It records additional details about the data flow in a system so that all persons involved can quickly look up the description of data flows, data stores, or processes.

To document the features of the system: It includes the parts or components and characteristics that distinguish each. Sometimes we also need to know under what circumstances each process is performed and how often the circumstance occurs. Once a feature have been articulated and recorded, all participants in the project will have a common source for information about the system.

To facilitate analysis of the details in order to evaluate characteristics and determination when system changes should be made : It is used to determine whether new features are needed in a system or whether changes of any type are in order.

Locate errors and omissions: It is also used to locate errors in the system descriptions. Conflicting data flow descriptions, processes that neither receive input nor generate output, data store that are never updated etc. indicate incomplete or incorrect analysis. Automatic data dictionary system have feature that will detect these difficulties to present in report.

Data structure is a set of data items that are related to one another and that collectively describe components of the system.

In addition, the data dictionary also gives information about data element/data structure, process list, cross-reference checking, and error detection.

Data dictionary are an essential aspect of data flow analysis and requirement determination. They should be used in conjunction. They should be used in conjunction with logic and process definitions.

- **SUMMARY:-**
  - When the volume of data is very large, it is very much difficult for analyst to manage the data definitions.
  - Data dictionaries are an **integral component of structured analysis; it provides additional information about the system.**
  - A data dictionary **is a catalog – a repository – of the elements in a system**.

- **SOFTWARE ENGINEERING**

Definition of Software Engineering:

Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machine.

Software Engineering is the application of systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is the application of Engineering to software.

It consist the following phases.

(1) Control Flow design

(2) Data Structure Oriented design

(3) Data Flow Oriented design

(4) Object oriented design

(1) Control Flow:-

As the size and complexity of programs increased programmers found that it is not only difficult to write cost-effective and correct programs but also to understand and maintain programs written by other programmers. To overcome from these problem programmers have started the design of the programs control structure.

(2) Data Structure:-

As computer became **more powerful with the advent of integrated circuits, they were used to solve more complex problems.** The control flow-based program development techniques were not sufficient to handle these problem and more effective program development techniques were needed

While developing a program it is more important to consider the design of the data structure of the program than to the design of its control structure.

Design techniques based on this principle are called data structure oriented design techniques.

The program code structure should correspond to the data structure. The data structure oriented design avoids any error related data.

(3) Data Flow Oriented design:-

As the requirement of more complex, integrated and sophisticated software arises the new concept of data flow-oriented techniques were proposed.

In this concept the major data items handled by a system must be first identified and then the processing required on these data items to be produce the required outputs should be determined.

The data flow techniques identify the different processing statements in system and the data that flow between the different processing stations.

This is useful in creating data flow model of entire system, which covers all the processing and data flow in any system i.e. in below figure represents the data flow representation of a car assembly unit where each processing station consumes certain input items and produces certain output.

(3) Object oriented design:-

With the further advancements in the field of software design, the data flow orientedtechnique or design is reached to a concept of object-oriented design. An object oriented techniques is a design approach where the natural objects such asemployees, payroll, register etc. occurring in a problem are first identified and then therelationship among the objects such as composition, reference and inheritance aredetermined.

Each object essentially acts as data hiding or data abstraction entity. Object oriented designed approach is targeting the convenience of users than developer.

UNIT - 2

BASIC OF SOFTWARE TESTING

- Introduction to software testing
- Software fault & failures
  (BUG/ERROR/DEFECT/FAULTS/FAILURES)
- Testing Artifects
  (Test case, Test Script, Test Plan, Test Harness, Test Suite)

- **Introduction to Software Testing**

  Testing is one type of investigation process for any product whether it is as per specification or not.

Software Testing is an **process to investigation** or insurance to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate.

Software Testing also provides an objective, independent view of the software to allow the **business to appreciate and understand the risks at implementation of the software**.

Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as **the process of validating and verifying that** a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics.

Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

- **SUMMARY:-**
  - Testing is one type of investigation process for any product whether it is as per specification or not.
  - Software Testing is an**process to investigation** or insurance to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate.
  - It can also be stated as **the process of validating and verifying that** a software program/application/product meets the business and technical requirements

- **Software Faults, Failures, Bug, Error and Defect**

  Software Fault & Failure:

  A **system failure** occurs when the delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service.

  This definition applies to both hardware and software system failures.

  Faults or bugs in hardware or a software component cause errors.

  An error is defined as that part of the system which is liable to lead to subsequent failure, and an error affecting the service is an indication that a failure occurs or has occurred. If the system comprises of multiple components, errors can lead to a component failure. As various components in the system interact, failure of one component might introduce one or more faults in another.

Bug:

A software bugis an **error, flaw, mistake, failure, or fault in a computer program** that prevents it from behaving as intended (e.g., producing an incorrect or unexpected result).

Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

A program that contains a large number of bugs, and/or bugs that seriously interfere with its functionality, is said to be buggy. Reports detailing bugs in a program are commonly known as **bug reports, fault reports, problem reports,** trouble reports, change requests, and so forth.

Error:

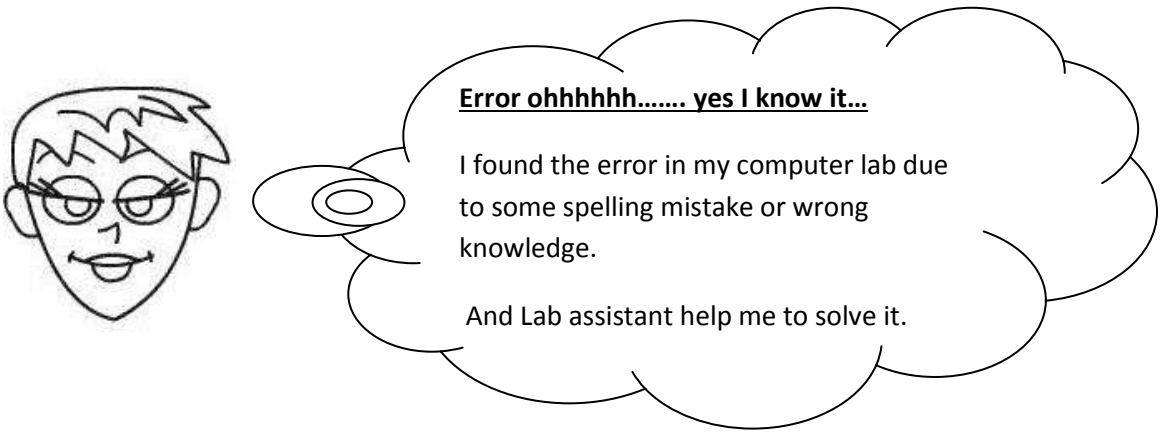
The word error has different meanings and usages relative to how it is conceptually applied. **The concrete meaning of the Latin word error is "wandering" or "straying".** To the contrary of **an illusion, an error or a mistake** can sometimes be dispelled through knowledge (knowing that one is looking at a mirage and not at real water doesn't make the mirage disappear).

However, some errors can occur even when individuals have the required knowledge to perform a task correctly.

Examples include forgetting to collect your change after buying chocolate from a vending machine, forgetting the original document after making photocopies, and forgetting to turn the gas off after cooking a meal. These slip errors can occur when an individual is distracted by something else.

DEFEAT:

The defect means the **failure of some angles** to add up to the expected amount of 360° or 180°, when such angles in the plane would. The opposite notion is the excess. The software product is in working but fail to perform its task properly.



- **Different Testing places or Testing Artifacts**
  (Test Case, Test script, Test Suite, Test plan, Test Harness)

- **Test case:**
  - A test case is usually **a single step,** or sequence of steps, and its expected result, along with various additional **pieces of information.**
  - It can occasionally be a series of steps but with one expected result or expected outcome.
  - **A test case is a set of conditions by which a tester can determine whether an application or software system is working correctly.**
  - The optional fields are a test case ID, Description, test step / order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps.
  - A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database or other common repository.
  - **Test cases include (1) Formal Test Cases and (2) Informal Test Cases.**
  - A formal written test-case includes some input and for find expected output, which is worked out before the test is executed.
  - The known input should test a precondition and the expected output should test a post condition.
  - Informal Test Cases works For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all but the activities and results are reported after the tests have been run.

- **SUMMARY:-**
  - A test case is usually **a single step,** or sequence of steps, and its expected result, along with various additional **pieces of information.**
  - **A test case is a set of conditions by which a tester can determine whether an application or software system is working correctly.**
  - **Test cases include (1) Formal Test Cases and (2) Informal Test Cases.**

- **Test Script:**
  - A test script in software testing is a **set of instructions that** will be performed on the system under test to test that the system functions as expected. These steps can be executed manually or automatically.
  - There are two ways to execute test scripts.
    →Manual testing
    →Automated testing
  - **Test cases are called manual testing.**
  - **Test scripts are short programs that written in a programming language.** It is used to testing the part of the functionality of a software system.
  - Test scripts written as short programs so it can either be written using a special automated functional **GUI test tool(Quick Test Professional) orin a well-known programming language(C,C++,PHP,JAVA)**

- **The major Advantage** of automated testing is:-
  - Tests may be executed continuously.
  - No continuous need of human interaction.
  - It is easily repeatable.
  - It is fast.
- **Disadvantages**
  - The major disadvantage is that if the automated tests may be poorly written and can break during playback.
  - Due to this disadvantage most systems are designed with human interaction in mind, it is good practice that a human tests the system at some point.
  - **Automated tests can only examine what they have been programmed to examine.**
  - A trained manual tester can notice that the system under test is misbehaving without being prompted or directed. Therefore, manual testers can find new bugs while ensuring that old bugs do not reappear while an automated test can only ensure the latter.

➢ **SUMMARY:-**

- A test script in software testing is a **set of instructions that** will be performed on the system under test to test that the system functions as expected. These steps can be executed manually or automatically.
- There are two ways to execute test scripts.

  →Manual testing

  →Automated testing
- **Test cases are called manual testing.**

  - **Test Suite:**
    - The most common term for a **collection of test cases is a test suite.** The test suite often also contains more detailed instructions / goals for each collection of test cases.
    - It definitely contains a section where the tester identifies the system configuration used during testing.
    - **A group of test cases** may also contain prerequisite states or steps, and descriptions of the following tests. Collections of test cases are sometimes incorrectly termed a test plan.
    - They may also be called a test script, or even a test scenario.
    - In software development, a test suite, less **commonly known as a validation suite**, is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors.
    - **A test suite often contains detailed instructions** or goals for each collection of test cases and information on the system configuration to be used during testing. A group of test cases also contain required states or steps, and descriptions of the following tests.

    - **Different types of test suites**

- Test suites are used to **group similar test cases together.**
- An executable test suite is executed by a program. This usually means that a test harness, which is integrated with the suite. The test suite and the test harness together can work on a sufficiently detailed level to correctly communicate with **the system under test (SUT**).
- A test suite for primarily testing subroutine might consist of a list number and testing subroutine. The testing subroutine would supply each number in the list to the primarily tester, and verify that the result of each test is correct.

➢ **SUMMARY:-**

- The most common term for a **collection of test cases is a test suite.** The test suite often also contains more detailed instructions / goals for each collection of test cases.
- **A group of test cases** may also contain prerequisite states or steps, and descriptions of the following tests.
- They may also be called a test script, or even a test scenario

▪ **Test plan:**

- It is the approach that will be used to test the system, not the individualtests. Most companies that use automated testing will call the code that is used their test scripts. Test plan is one type of documents that insure that the software or product is as per specification and satisfied the customer.
- **A test plan is a systematic approach to testing a system such as a machine or software.** The plan typically contains a detailed understanding of what the eventual workflow will be.
- A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with proper inputs from Test Engineers.
- Depending on the product and the responsibility of the organization to which the test plan applies, **a test plan may include one or more of the following:**
  - ***Design Verification test*** – this testing is performed during the development or approval stages of the product or software.
  - ***Manufacturing or Production test*** – this testing is performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
  - ***Acceptance test –*** this testing is performed at the time of delivery or installation of the product.
  - ***Service and Repair test*** – this is performed as per requirement during the life of the product.
  - ***Regression test*** – Here the platform is upgraded and the existing application will be run. The testing is performed on an existing product, to verify that existing functionality didn't get broken when other environment is changed.

- **Note: -** (Here remember the steps of SDLC –preliminary investigation,-system analysis,-system design,-coding,-testing,-installation. ---Now compare with Test plan testing points)

Just think that if you have any complex work then what you do???

→yes your answer is right you are doing ***planning.***

- **Test Harness (harness means tools or equipment):**
  - In software testing, a test harness or **automated test framework** is a **collection of software and test data.** Here **different equipments and tools are used to configured test programs and its units by running them under different conditions and monitoring its behavior and outputs.**
  - It has two main parts: the Test execution engine and the Test script.
  - **Test harnesses allow for the automation of tests.** They can call functions with supplied parameters and print out and compare the results to the desired value. The test harness is a hook to the developed code, which can be tested using an automation framework.
  - A test harness should allow specific tests to run (this helps in optimizing), orchestrate a runtime environment, and provide a capability to analyze results.

  - **Objectives of test harness:**
    - **Automation in the testing process.**
    - **Execute test suites of test cases.**
    - **Generate associated test reports.**
  - **Benefits of test harness:**
    - Increased productivity due to automation of the testing process.
    - Increased probability by regression testing.
      (Regression testing is one type of software testing that is used to find uncover errors and new software bug.)
    - Increased quality of software product.
    - Ensure that all the software parts will be checked.
    - No human interaction is needed continuously.
    - Testing done with simulate, which is not possible in real time environment.

TYPES OF SOFTWARE TESTING, VERIFICATION AND VALIDATION

- Static Testing (Informal Review, Walthrough, Technical Review, Inspection)
- Dynamic Testing
- Test Level (Unit Testing, Integration Testing, System Testing, Acceptance Testing)

➢ **Techniques of Software Testing**

- Black Box Testing (Equivalence Partitioning, Boundary Data Analysis, Decision Table Testing, State Transition Testing).
- White Box Testing (Statement Testing and Coverage, Decision Testing and Coverage)
- Grey Box Testing
- NonFunctional Testing (Performance Testing, Stress Testing, Load Testing, Usability Testing, Security Testing)

➢ **Static testing:**

Static testing is a **testing where the actual software is not used or tested.** It means that here the **software parts like coding, algorithms or documents are tested to find the errors.** It is done by software developers who actually wrote the code. The primary syntax checking and manually reviewing the coding testing is done by developers at the time of software development.

It is not detail testing because here the developers are involve in the testing and during the software development the testing is performed with primary basis.

It just checks basic coding, algorithm and document of software.

The main aim of this testing is to find errors at startup. It is primarily syntax checking of the code and/or manually reviewing the code or document **to find errors.** This type of testing can be used by the developer who wrote the code.

Static testing is a process of evaluating the software documents. this document describes static test techniques such as reviews, code analysis and static analysis

It includes:-

Code reviews (Technical review and informal review)
Inspections
Walkthroughs

A **software review** is "A process or meeting during which a software product is [examined by] project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval"

Formal reviews greatly outperform **informal reviews** in cost-effectiveness. Informal reviews may often be unnecessarily expensive (because of time-wasting through lack of focus), and frequently provide a sense of security which is quite unjustified by the relatively small number of real defects found and repaired.

Technical review is a form of peer review in which "a team of qualified personnel examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards. Technical reviews may also provide recommendations of alternatives and examination of various alternatives"

In software engineering, a **walkthrough** or **walk-through** is a form of software peer review "in which **a designer or programmer leads members of the development team a**nd other interested parties through a software product, and the participants ask questions

and make comments about possible errors, violation of development standards, and other problems".

In general, a walkthrough has one or two broad objectives: to gain feedback about the technical quality or content of the document; and/or to familiarize the audience with the content.

A walkthrough is normally organized and directed by the author of the technical document. Any combination of interested or technically qualified personnel (from within or outside the project) may be included as seems appropriate.

Inspection in software engineering refers to peer review of any work product by trained individuals who look for defects using a well defined process.

(Hey just remember that in your exam hall some experts appointed for inspection to stop any copy case)

An inspection is one of the most common sorts of review practices found in software projects. The goal of the inspection is for all of the inspectors to reach consensus on a work product and approve it for use in the project. Commonly inspected work products include software requirements specifications and test plans. In an inspection, a work product is selected for review and a team is gathered for an inspection meeting to review the work product.

The goal of the inspection is to identify defects. In an inspection, a defect is any part of the work product that will keep an inspector from approving it.

The stages in **the inspections process are: Planning, Overview meeting, Preparation, Inspection meeting, Rework and Follow-up. The Preparation, Inspection meeting and Rework stages might be iterated.**

- **SUMMARY : -**
  - Static testing is a **testing where the actual software is not used or tested.** It means that here the **software parts like coding, algorithms or documents are tested to find the errors.**
  - **It is not detail testing.** It just checks basic coding, algorithm and document of software.
  - It includes:-
    - **Code reviews (Technical review and informal review)**
    - **Inspections**
    - **Walkthroughs**
  - **Reviews : -**
    - **Software reviews**
    - **Formal reviews**
    - **Technical reviews**
    - **Walkthrough**
    - **Inspection**
    -
- **Dynamic testing**
  - **Dynamic testingis also known as dynamic analysis.** It is a term used in software engineering to describe the testing **of the dynamic behavior of code.**
  - That is, **dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time.**
  - In dynamic testing the software must be compiled and run. Dynamic Testing involves working with the software, giving input values, dummy data and checking if the output is as

per expected or not. Dynamic testing means testing based on specific test cases by execution of the test object or running programs.

- Dynamic testing is involves during Unit Tests, Integration Tests, System Tests and Acceptance Tests. The difference between Static testing and dynamic testing is that in dynamic testing tester is used to test software through executing it. This is in contrast to Static testing.

Difference between Static testing and Dynamic testing:

No	Static testing	Dynamic testing
1	Static testing done without executing the program.	Dynamic testing done by executing the program.
2	Static testing does verification process.	Dynamic testing does validation process.
3	Static testing is about prevention of defects.	Dynamic testing is about finding and fixing the defects.
4	Static testing gives assessment of code and documentation.	Dynamic testing involves test cases for execution.
5	Static testing involves checklist and process to be followed.	Dynamic testing involves test cases for execution.
6	This testing can be performed before compilation.	This testing is performed after compilation.
7	This testing covers the structural and statement coverage testing.	This testing covers the executable file of the code.
8	Cost of finding defects and fixing is less.	Cost of finding and fixing defects is high.
9	Return on investment will be high as this process involved at early stage.	Return on investment will be low as this process involves after the development phase.
10	More reviews comments are highly recommended for good quality.	More defects are highly recommended for good quality.
11	Requires loads of meetings.	Comparatively requires lesser meetings.

- **SUMMARY : -**
  - **Dynamic testingis also known as dynamic analysis.**
  - Testing **of the dynamic behavior of code.**
  - Dynamic Testing involves working with the software, giving input values, dummy data and checking if the output is as per expected or not.

- **Testing Levels:**

- **Unit testing** - **Testing of individual software components or modules.** Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.
- This type of testing is **performed by the developers before the setup is handed over to the testing team to formally execute the test cases.** Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.
- The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

- **Limitations of Unit Testing**

    - Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.
    - There is a limit to the number of scenarios and test data that the developer can use to verify the source code. So after he has exhausted all options there is no choice but to stop unit testing and merge the code segment with other units.

- **Integration testing** - Testing of integrated modules to verify combined functionality after integration. **Modules are typically code modules, individual applications, client and server applications on a network, etc.This type of testing is especially relevant to client/server and distributed systems.**
- The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.
- **Integration Testing Method**

- **Bottom-up integration**
    - This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

- **Top-Down integration**
    - This testing, **the highest-level modules are tested first and progressively lower-level modules are tested after that**.
    - In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and network

- **System testing tests** a completely integrated system to verify that it meets its requirements.
- This is the next level in the testing and tests the system as a whole. **Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards.** This type of testing is performed by a specialized testing team.
- System testing is so important because of the following reasons:

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.
- **System integration testing** verifies that a system is integrated to any external or third party systems defined in the system requirements.
    - **Regression testing** - Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.
    - Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug hasn't resulted in another functionality or business rule violation is Regression testing. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.

    - **Regression testing is so important because of the following reasons:**
        - Minimize the gaps in testing when an application with changes made has to be tested.
        - Testing the new changes to verify that the change made did not affect any other area of the application.
        - Mitigates Risks when regression testing is performed on the application.
        - Test coverage is increased without compromising timelines.
        - Increase speed to market the product.

- **Acceptance testing** Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.
- This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.
- More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application.
- By performing acceptance tests on an application the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

- **Alpha testing** is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site.
- This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing**. During this phase, the following will be tested in the application:**
    - **Spelling Mistakes**
    - **Broken Links**
    - **Cloudy Directions**

- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.
- **Beta testing comes after alpha testing.** Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs.
- This test is **performed after Alpha testing has been successfully performed.** In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program **a "real-world" test** and partly to provide a preview of the next release. In this phase the audience will be testing the following:
    - Users will install, run the application and send their feedback to the project team.
    - Typographical errors, confusing application flow, and even crashes.
    - Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
    - The more issues you fix that solve real user problems, the higher the quality of your application will be.
    - Having a higher-quality application when you release to the general public will increase customer satisfaction.

- **SUMMARY (TESTING LEVEL): -**
- UNIT TESTING:-
    - **Testing of individual software components or modules.**
    - **Performed by the developers before the setup is handed over to the testing team to formally execute the test cases.**
    - The goal of unit testing is to isolate each part of the program.
- INTEGRATION TESTING:-
    - Testing of integrated modules to verify combined functionality after integration.
    - Modules are:- Code module,individual applications, client and server applications on a network
    - Bottom-up integration:-
    - This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
    - Top-down integration:-

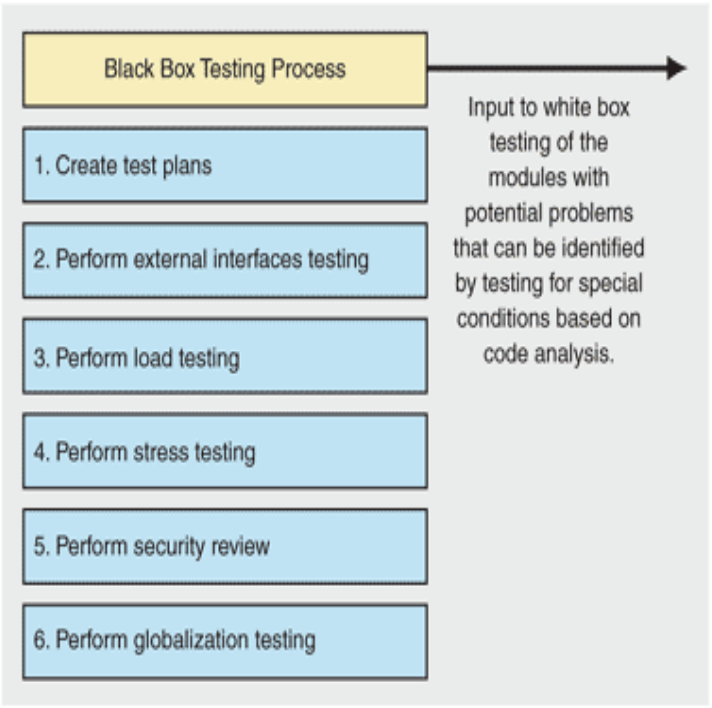
- This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.
- **Regression testing:-**
- Testing the application as a whole for the modification in any module or functionality.
- **Acceptance testing:-**
- This type of testing is done to verify if system meets the customer specified requirements. User or customers do this testing to determine whether to accept application.

➢ **Explain Black box, White box, Grey box testing**

- **Black box testing: (blind testing,Behavioral Testing)**
  - **The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.** The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
  - **Black box testing tests all possible combinations of end-user actions.** Black box testing **assumes no knowledge of code and is intended to simulate the end-user experience.** You can use sample applications to integrate and test the application block for black box testing. You can begin planning for black box testing immediately after the requirements and the functional specifications are available.
  - **Black box testing treats the software as a "black box"—without any knowledge of internal implementation.**
  - Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.
    - **Specification-based testing**:
      - Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester **inputs data into, and only sees the output from**, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.
      - Specification-based testing is necessary, but it is insufficient to guard against certain risks.
    - **Advantages and disadvantages**:
      - The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code *must* have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not. *But,* on the other hand, black box testing has been said to be **"like a walk in a darkwithout a flashlight**," because the tester doesn't know how the software being tested was actually constructed.

- As a result, there are situations when
  - (1) a tester writes many test cases to check something that could have been tested by only one test case, and/or
  - (2) some parts of the back-end are not tested at all.Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

- **Black Box Testing Steps**
  - Black box testing involves testing external interfaces to ensure that the code meets functional and nonfunctional requirements. The various steps involved in black box testing are the following:
    1. **Create test plans.** Create prioritized test plans for black box testing.
    2. **Test the external interfaces.** Test the external interfaces for various type of inputs using automated test suites, such as NUnit suites and custom prototype applications.
    3. **Perform load testing.** Load test the application block to analyze the behavior at various load levels. This ensures that it meets all performance objectives that are stated as requirements.
    4. **Perform stress testing.** Stress test the application block to analyze various bottlenecks and to identify any issues visible only under extreme load conditions, such as race conditions and contentions.
    5. **Perform security testing.** Test for possible threats in deployment scenarios. Deploy the application block in a simulated target environment and try to hack the application by exploiting any possible weakness of the application block.
    6. **Perform globalization testing.** Execute test cases to ensure that the application block can be integrated with applications targeted toward locales other than the default locale used for development.



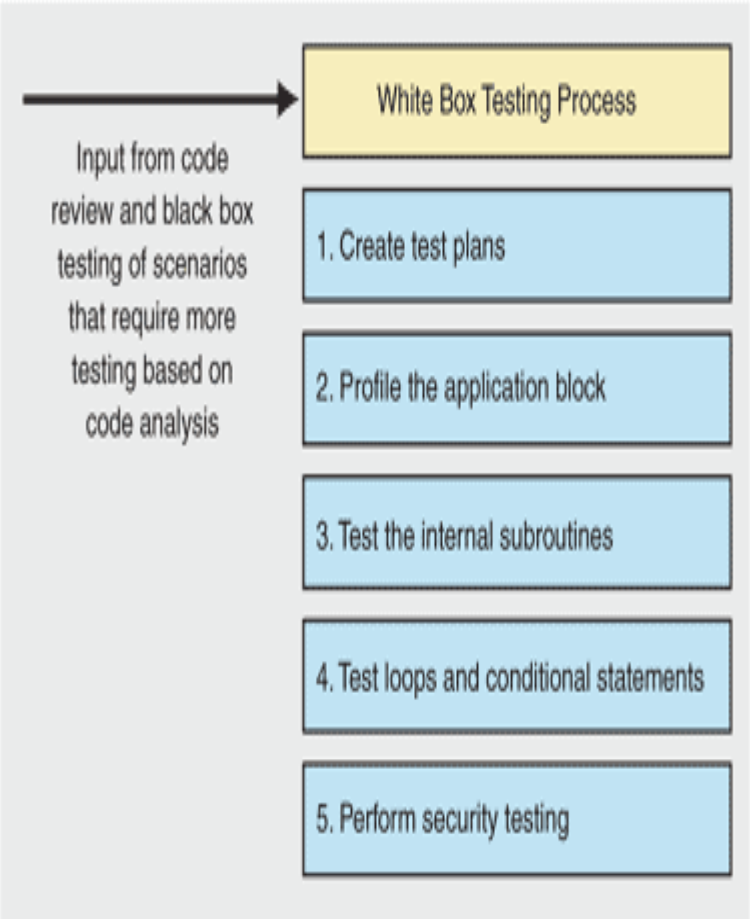
Advantage and Disadvantage of Black box testing

Advantages	Disadvantages
-Well suited and efficient for large code segments. -Code Access not required. -Clearly separates user's perspective from the developer's perspective through visibly defined roles. -Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.	-Limited Coverage since only a selected number of test scenarios are actually performed. -Inefficient testing, due to the fact that the tester only has limited knowledge about an application. -Blind Coverage, since the tester cannot target specific code segments or error prone areas. -The test cases are difficult to design.

- **White box testing:**
  - **White box testing is also known as Structural Testing, clear box testing, open box testing or Logic-driven Testing or Glass Box Testing.**
  - White box testing is the detailed investigation of internal logic and structure of the code. **White box testing is also called glass testing or open box testing.** In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
  - **The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.**
  - In white box testing, you create test cases by looking at the code to detect any potential failure scenarios. You determine the suitable input data for testing various APIs and the special code paths that need to be tested by analyzing the source code for the application block. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available.
  - A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.
  - **White box testing** is **when the tester has access to the internal data structures and algorithms including the code that implement these**.
  - **Types of white box testing**
    - The following types of white box testing exist:
      - **API** testing (application programming interface) - testing of the application using public and private APIs
      - **Code coverage** - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
      - **Fault injection** methods - improving the coverage of a test by introducing faults to test code paths
      - **Static testing** - White box testing includes all static testing.
      - **Test coverage**White box testing methods can also be used to evaluate the completeness of a test suite that was created with black

box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

- Two common forms of code coverage are:
- ***Function coverage***, which reports on functions executed
- ***Statement coverage***, which reports on the number of lines executed to complete the test
- They both return code coverage metric, measured as a percentage.
- White box testing involves the following steps:
  1. **Create test plans.** Identify all white box test scenarios and prioritize them.
  2. **Profile the application block.** This step involves studying the code at run time to understand the resource utilization, time spent by various methods and operations, areas in code that are not accessed, and so on.
  3. **Test the internal subroutines.** This step ensures that the subroutines or the nonpublic interfaces can handle all types of data appropriately.
  4. **Test loops and conditional statements.** This step focuses on testing the loops and conditional statements for accuracy and efficiency for different data inputs.
  5. **Perform security testing.** White box security testing helps you understand possible security loopholes by looking at the way the code handles security.



Source Code is available for Testing

1)Structural Testing process
2)Program Logic-driven Testing
3)Design-based Testing
4)Examines the internal structure of program

White box testing techniques

1) Basis path testing
2) Complexity testing

White box testing is

1) Derive test cases:
2) Based on program structure.
3) To ensure that all independent paths within a module of the program have been tested.

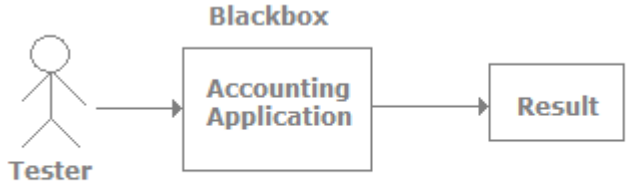
Maximum criteria for White Box Testing

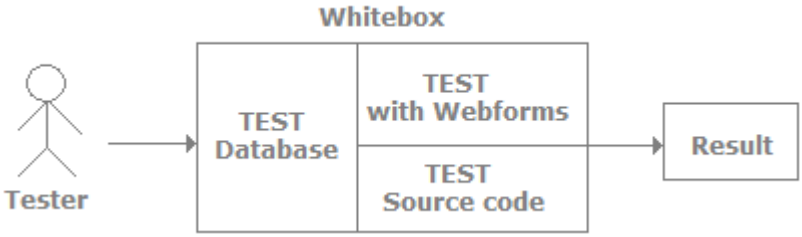
1) Code
2) Document High Level Design
3) Low level Design Document
4) Application Requirements Specification

Advantage and Disadvantage of White box testing

Advantages	Disadvantages
- As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively. - It helps in optimizing the code. -Extra lines of code can be removed which can bring in hidden defects. - Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	- Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased. - Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will go untested. - It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.

A simple diagram to understand Black box testing and White box testing.





- **Grey box testing:**
  - **Grey box testinginvolves having knowledge of internal data structures and algorithms for purposes of designing the test cases, but testing** at the user, or black-box level.
  - **Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application.** In software testing, the term the more you know the better carries a lot of weight when testing an application.
  - Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.
  - Manipulating input data and formatting output do not qualify as grey box, because the input and output are clearly outside of the "black-box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test. However, modifying a data repository does qualify as grey box, as the user would not normally be able to change the **data outside of the system under test**. Grey box testing may also include reverse engineering to determine, for instance, boundary values or error messages.

Advantage and Disadvantage of Grey box testing

Advantages	Disadvantages
- Offers combined benefits of black box and white box testing wherever possible. - Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications. - Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication	- Since the access to source code is not available, the ability to go over the code and test coverage is limited. - The tests can be redundant if the software designer has already run a test case. - Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many

protocols and data type handling. - The test is done from the point of view of the user and not the designer.	program paths will go untested.

- **Finally general idea about these three testing fundas.**

- **Black Box v/s Grey Box v/s White Box**

S.N.	Black Box Testing	Grey Box Testing	White Box Testing
1	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2	Also known as closed box testing, data driven testing, behavior testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4	Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

- **Nonfunctional Testing:**
  - This section is based upon the **testing of the application from its non-functional attributes.** Non-functional testing of Software **involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.**
  - Some of the important and commonly used non-functional testing types are mentioned as follows:
  - **Performance testing** - Term often used interchangeably with 'stress' and 'load' testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

- It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:
    - **Network delay.**
    - **Client side processing.**
    - **Database transaction processing.**
    - **Load balancing between servers.**
    - **Data rendering.**
- Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:
    - Speed (i.e. Response Time, data rendering and accessing)
    - Capacity
    - Stability
    - Scalability
- It can be either qualitative or quantitative testing activity and can be divided into different sub types such as Load testing and Stress testing.
    - **Stress testing** - System is stressed beyond its specifications to check how and when it fails. **Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.**
    - This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.
    - The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:
        - Shutdown or restart of Network ports randomly.
        - Turning the database on or off.
        - Running different processes that consume resources such as CPU, Memory, server etc.
    - **Load testing** - Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.
    - A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.
    - Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.
    - Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements.
    - **Security testing** - Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

- Security testing involves the testing of Software in order to identify any flaws ad gaps from security and vulnerability point of view. Following are the main aspects which Security testing should ensure:

    - Confidentiality.
    - Integrity.
    - Authentication.
    - Availability.
    - Authorization.
    - Non-repudiation.
    - Software is secure against known and unknown vulnerabilities.
    - Software data is secure.
    - Software is according to all security regulations.
    - Input checking and validation.
    - SQL insertion attacks.
    - Injection flaws.
    - Session management issues.
    - Cross-site scripting attacks.
    - Buffer overflows vulnerabilities.
    - Directory traversal attacks.

- **Usability testing** is needed to check if the user interface is easy to use and understand.
- This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

➢ **SUMMARY (BLACKBOX TESTING): -**

- Testing without having any knowledge of the interior workings of the application is Black Box testing.
- Tests all possible combinations of end-user actions.
- Assumes no knowledge of code and is intended to simulate the end-user experience.
- Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

➢ Black Box Testing Steps

1. Create test plans.
2. Test the external interfaces.
3. Perform load testing.
4. Perform stress testing.
5. Perform security testing.
6. Perform globalization testing.

➢ **SUMMARY (WHITEBOX TESTING): -**

- Also known as Structural Testing, clear box testing, open box testing or Logic-driven Testing or Glass Box Testing.
- White box testing is the detailed investigation of internal logic and structure of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

- When the tester has access to the internal data structures and algorithms including the code that implement these.
    - White box testing steps:
        1. Create test plans.
        2. Profile the application block.
        3. Test the internal subroutines.
        4. Test loops and conditional statements.
        5. Perform security testing.

- **SUMMARY (GRAY TESTING): -**
    - Having knowledge of internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.
    - Technique to test the application with limited knowledge of the internal workings of an application.
    - This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

- **SUMMARY (Nonfunctional TESTING): -**
    - **Testing of the application from its non-functional attributes.**
    - **Involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.**
    - Non-functional testing types are mentioned as follows:
        - **Performance testing**
            - **Stress testing**
            - **Load testing**
            - **Security testing**
        - **Usability testing**

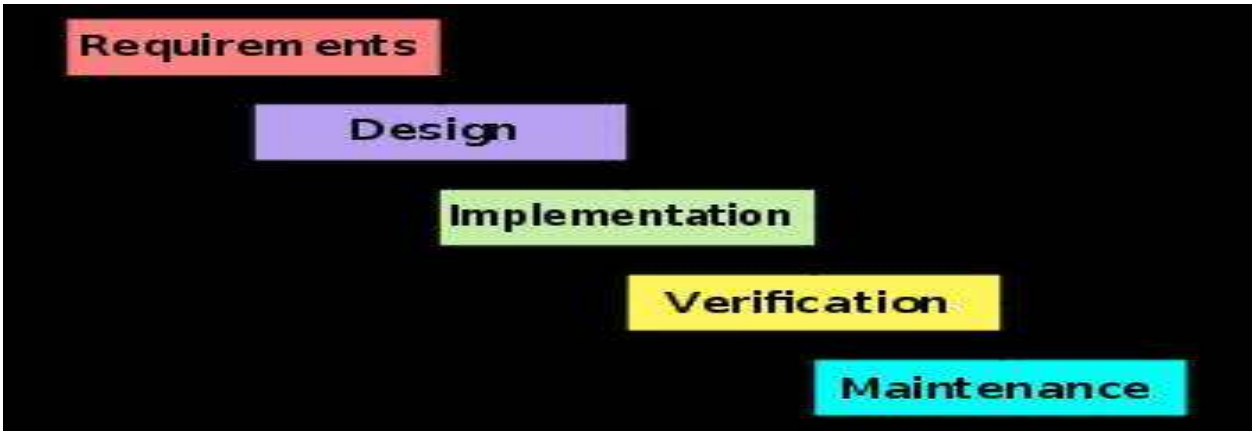
UNIT-3

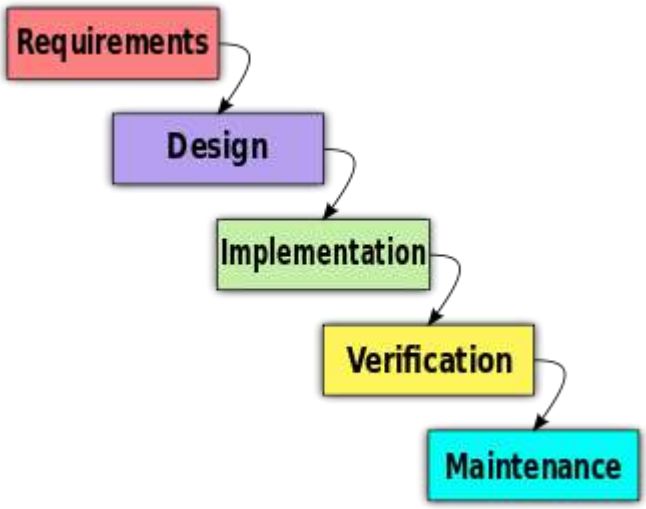
SOFTWARE DEVELOPMENT LIFECYCLE MODEL

- Watefall Model
- Iterative Model
- V-Model
- Sprial Model
- Big Bang Model
- Prototyping Model

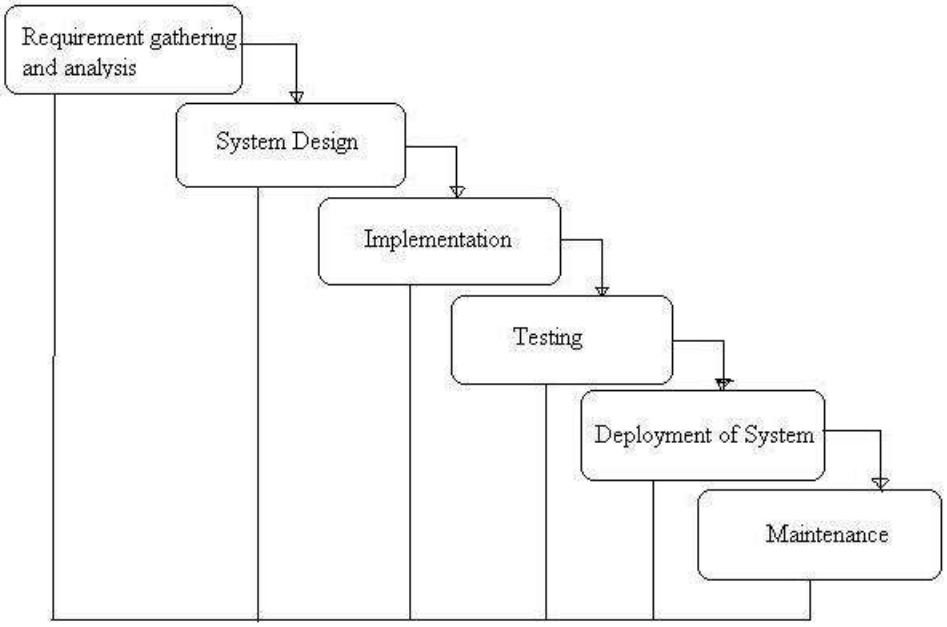
➢ Waterfall Model.

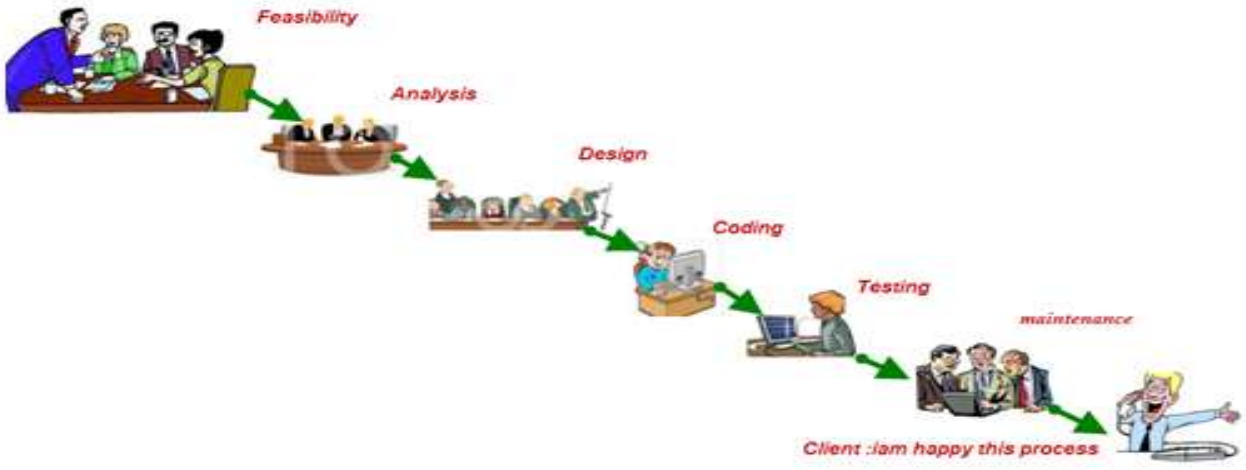
- The **waterfall model** is a sequential software development process, in which progress is seen as flow like a waterfall.
- It is also called the ***classic life cycle***or the ***waterfall model*** *or* **the *linear sequential model.***
- It suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support. Figure illustrates the linear sequential model for software engineering.
- Progress flows from the top to the bottom, like a waterfall.
- To follow **the *waterfall model*, one proceeds from** one phase to the next in a sequential manner.
- For example, one first completes **requirements specification**, which after sign-off are considered "set in stone."
- When the requirements are fully completed, one **proceeds to design**. The software in question is designed and a **blueprint is drawn** for implementers (coders) to follow — this design should be a plan for implementing the requirements given.
- When the design is fully completed, an implementation of that design is made by **coders.**
- Towards the later stages of this implementation phase, separate software components produced are combined to introduce new functionality and reduced risk through the removal of errors.





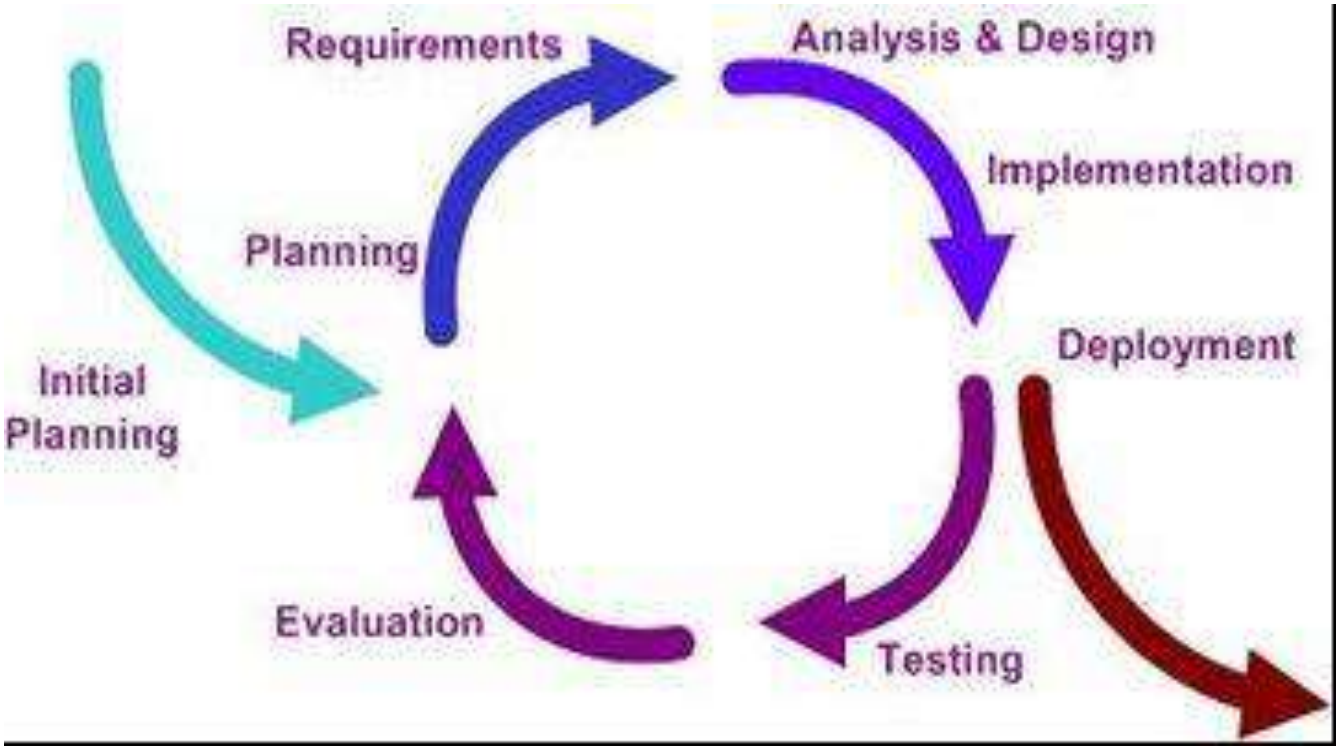
General Overview of "Waterfall Model"

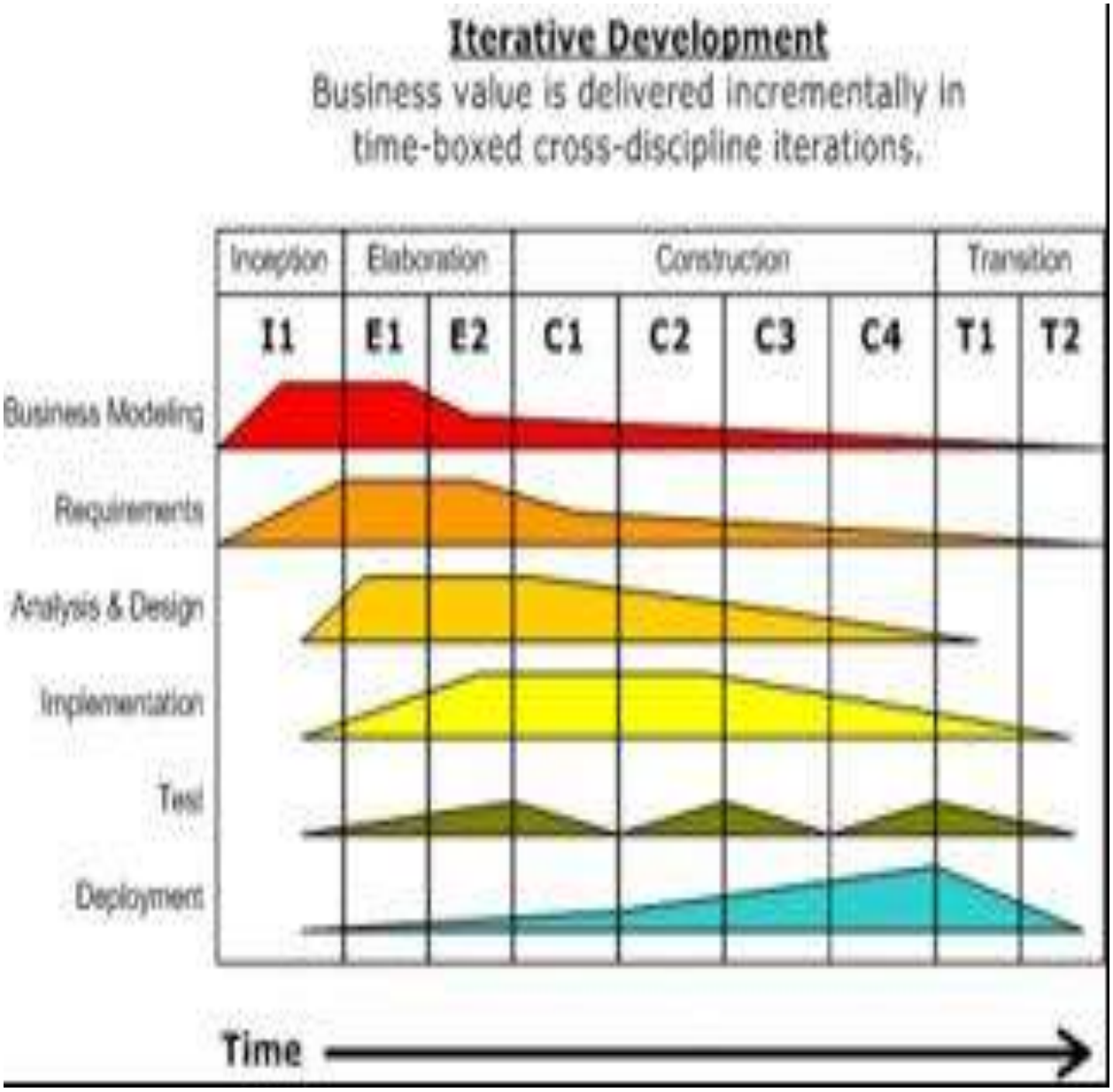


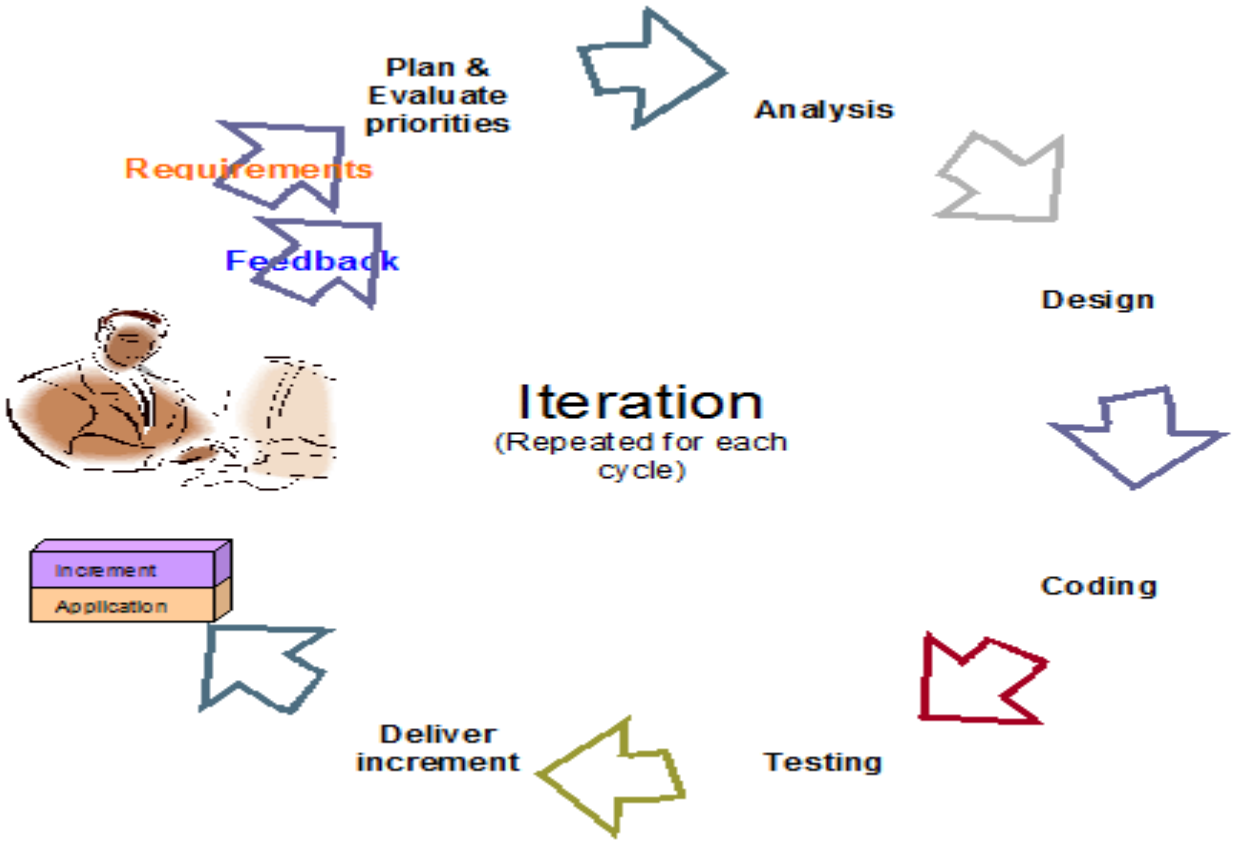


- **Iterative Model.**
  - **Iterative and incremental development**

- **Iterative and Incremental development** is a **cyclic software development** process developed in response to the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interaction in between.
- **The iterative and incremental development is** an essential part of the Rational Unified Process, the Dynamic Systems DevelopmentMethod, Extreme Programming and generally the agile software development frameworks.
- Incremental development is **a scheduling and staging strategy**, in which the various parts of the system are developed.
- The basic idea behind iterative enhancement is to develop a software system incrementally, allowing the developer to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system.




Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Time

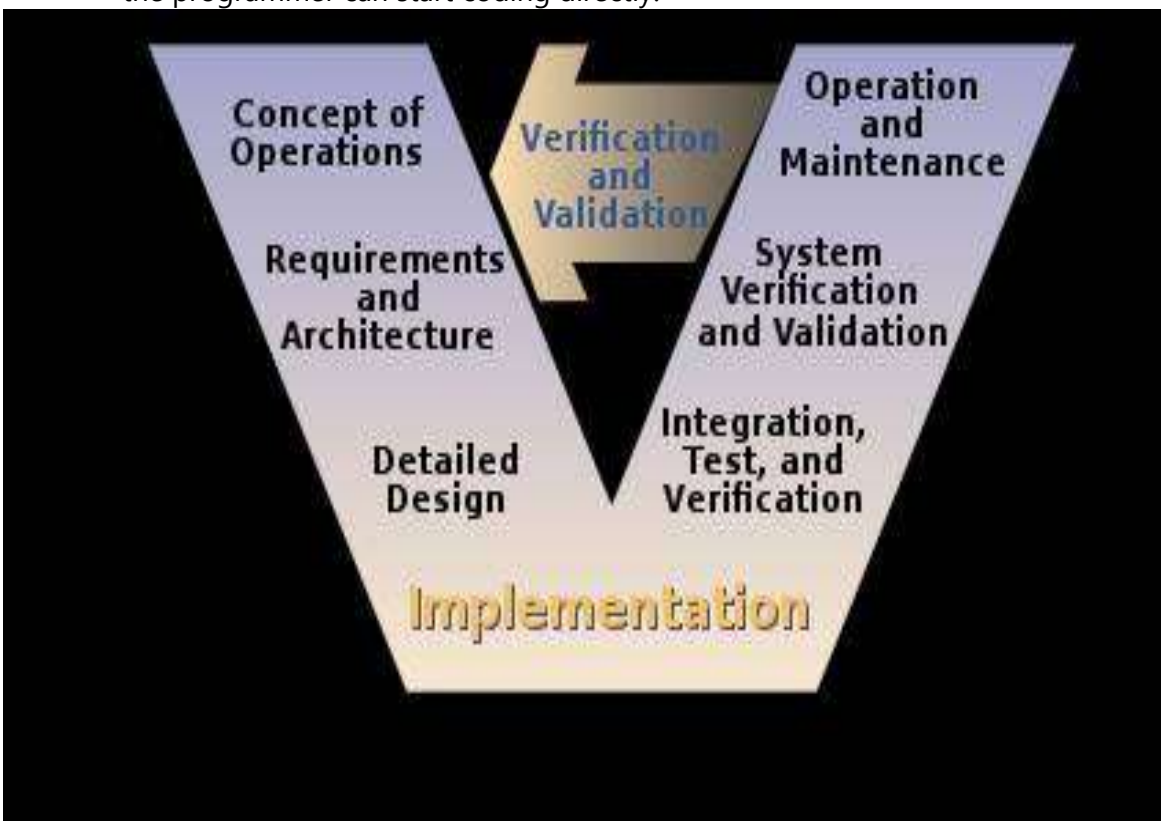


➢ **V Model .**

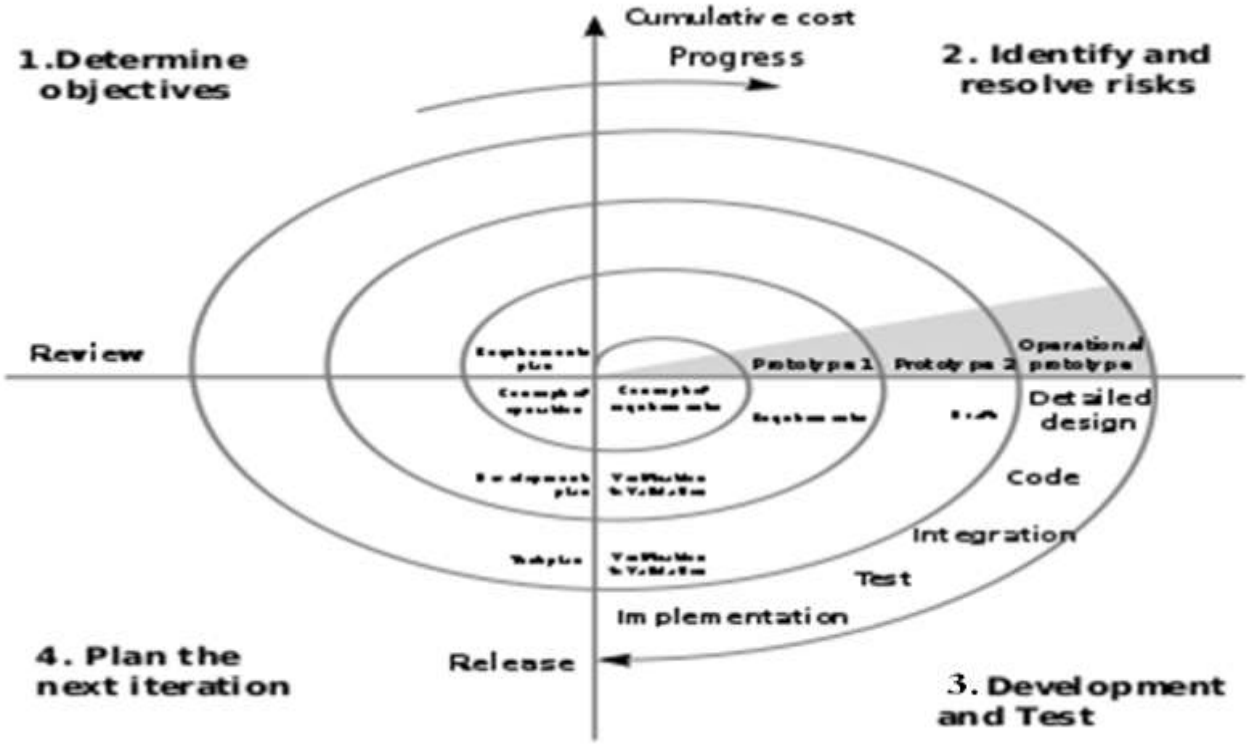
- The **V-model** is a **software development process** (also applicable to hardware development) which can be presumed to be the extension of the waterfall model.
- Instead of moving down in a linear way, the process steps are upwards after the coding phase, to form **the typical V shape**. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes **represents time or projectcompleteness (left-to-right) and** level of abstraction (coarsest-grain abstraction uppermost), respectively.
- **Verification Phases**
- Requirements analysis
- System Design
- Architecture Design

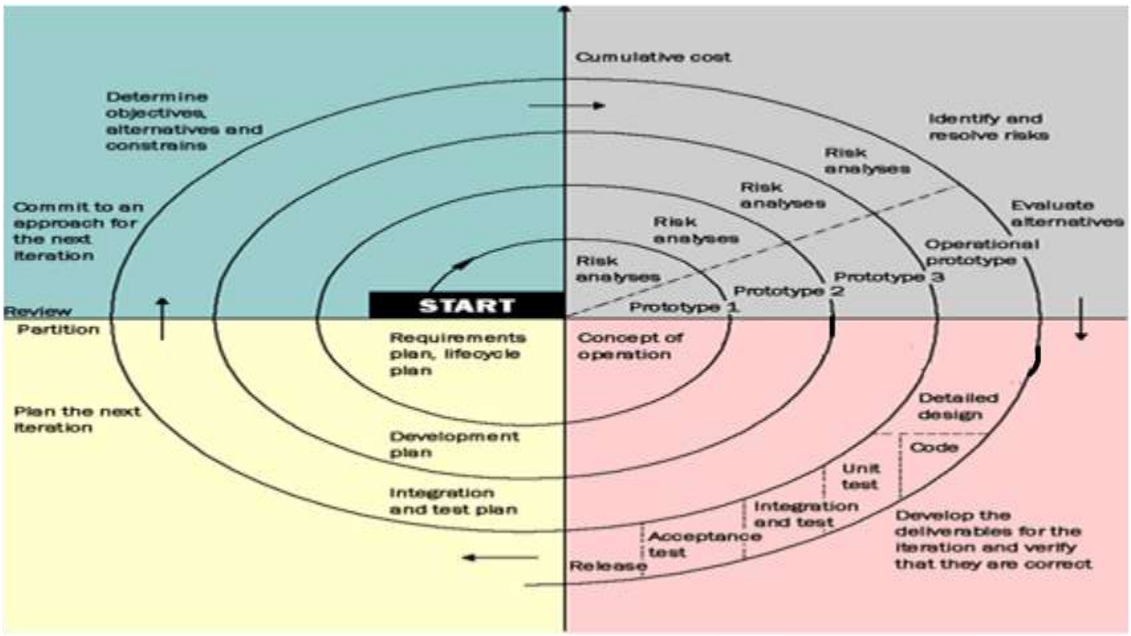
  - ***Module Design***
    - (1)**In the Requirements analysis phase, the**requirements of the proposed system are collected by analyzing the needs of the user(s). This phase is concerned about establishing what the ideal system has to perform.
    - (2)**Systems design is the phase** where system engineers analyze and understand the business of the proposed system by studying the user requirements document. They figure out possibilities and techniques by which the user requirements can be implemented.
    - (3)**The phase of the design of computer architecture** and software architecture can also be referred to as high-level design. The baseline in selecting the architecture is that it should realize all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, databasetables, architecture diagrams, technology details etc.

- (4)**The module design phase** can also be referred to as low-level design. The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly.



- **Spiral model.**
  - The **spiral model** is a **software development process combining which elements of both design and prototyping-in-stages, (the first or original model from which others are copied) in an effort to** combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model (or spiral development), it is a **systems development method (SDM)** used in information technology (IT).
  - This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects.
  - **"A Spiral Model of Software Development and Enhancement".** This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters.





- **Steps**
  - The **steps in the spiral model iteration can be generalized as** follows:
  - The system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
  - A preliminary design is created for the new system. This phase is the most important part of **"Spiral Model". In** this phase all possible (and available) alternatives, which can help in developing a cost effective project are analyzed and strategies to use them are decided. This phase has been added specially in order to identify and resolve all the possible risks in the project development. If risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed with the available data and find out possible solution in order to deal with the potential changes in the requirements.

- A first **prototype of the new system is constructed from the preliminary design.** This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
- **evaluating the first prototype in terms of its strengths, weaknesses, and risks;**
- defining the requirements of the second prototype;
- planning and designing the second prototype;
- Constructing and testing the second prototype.

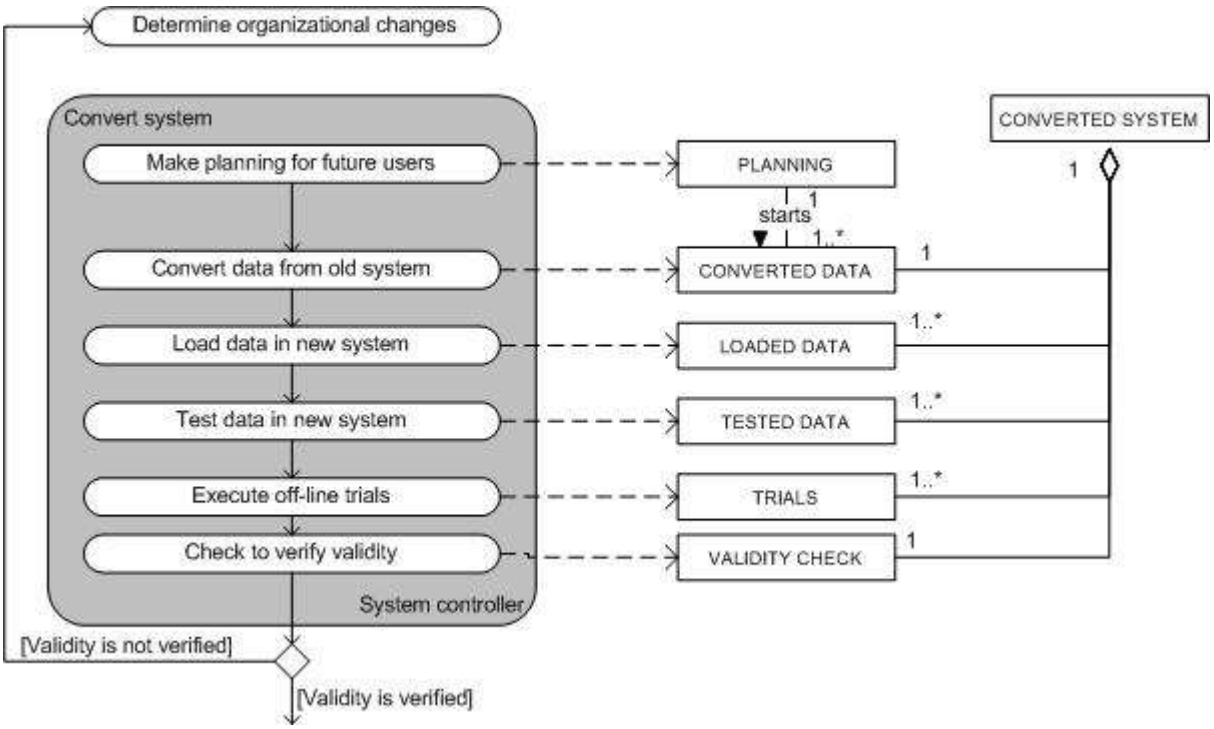
- **Big Bang Model.**
  - Big bang adoption is the **adoption type of the instant changeover, when everybody associated with the new system moves to the fully functioning** new system on a given date.
  - When a new system needs to be implemented in an organization, there are three different ways to adopt this new system:
    - **The big bang adoption, phased adoption and parallel adoption.**
    - **In case of parallel adoption the old and the new system are running parallel, so all the** users can get used to the new system, and meanwhile do their work using the old system.
    - **Phased adoption** means that the adoption will happen in several phases, so after each phase the system is a little nearer to be fully adopted.
    - With **the big bang adoption**, the switch between using the old system and using the new system happens at one single date, the so called instant changeover of the system. Everybody starts to use the new system at the same date and the old system will not be used anymore from that moment on.
    - **The big bang adoption type is riskier than other adoption types because** there are fewer learning opportunities incorporated in the approach, so quite some preparation is needed to get to the big bang. This preparation will be described below, illustrated by the process-data model of the big bang adoption.

Advantages and disadvantages

- This sudden changeover is quite drastic. **This has advantages, but because of the instant changeover there are also disadvantages.**
- **The advantages of this method:**
  - **Training is only needed for the new method, not also for the changeover period.**
  - **User documentation does not need to be updated during the implementation process, because it happens in such a short period.**
  - **The changeover is at one date and this date is clear for everyone.**
  - **There are no special interfaces needed to be able to get used to the new system, because the new system is all there is.**
- **The disadvantages on the other hand are:**
  - There is no time for extra additions

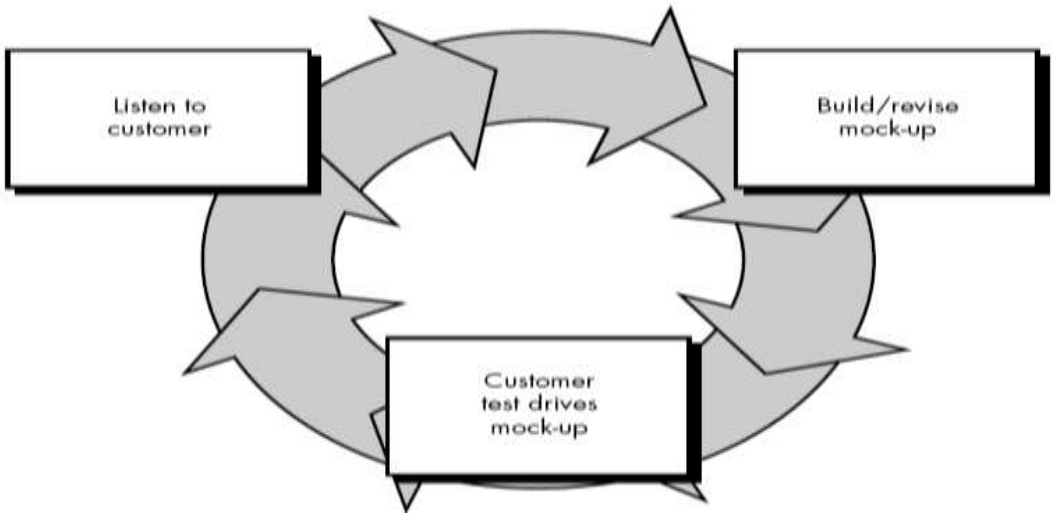
- The completeness and validity of the converted data is not completely proved, only in the pre-phases, but not in the whole system situation.
- Start up problems are a problematic factor
- The operation is complex, one of the main complexities is tuning all activities to happen on one moment: the big bang
- 'Fall back'-plans are hard to develop and become more impossible when the big bang has taken place There can be a catch up period (as explained above)
- This adoption is vulnerable because of the limited possibilities to maneuver. There is a lot of pressure because the deadline must be met.

Convert system with BigBang



➢ **PROTOTYPING MODEL**

- Often, a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements.
- In other cases, **the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a *prototyping paradigm* may offer the best approach.**
- The prototyping paradigm (Display in Figure) **begins with requirements gathering**. **Developer and customer meet and define the overall objectives for the software**, identify whatever requirements are known, and outline areas where further definition is mandatory.
- A "quick design" then occurs. The **quick design focuses on a representation of those aspects of the software that will be visible to the customer/user** (e.g., input approaches and output formats).



- The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed.
- Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.
- Ideally, the prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to use existing program fragments or applies tools (e.g., report generators, window managers) that enable working programs to be generated quickly.
- In most projects, the first system built is barely usable. It may be too slow, too big, and awkward in use or all three.
- There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved.
- When a new system concept or new technology is used, one has to build a system to throw away, for even the best planning is not so omniscient as to get it right the first time.

SUMMARY:-

- **Waterfall Model.**
  - The waterfall model is a sequential software development process, in which progress is seen as flow like a waterfall.
  - It is also called the *classic life cycle* or the *waterfall model or* the *linear sequential model.*
  - Progress flows from the top to the bottom, like a waterfall.
  - To follow the *waterfall model*, one proceeds from one phase to the next in a sequential manner.
  - When the design is fully completed, an implementation of that design is made by coders.

- **Iterative Model.**
  - Iterative and incremental development
  - Iterative and Incremental development is a cyclic software development process developed in response to the weaknesses of the waterfall model.
  - It starts with an initial planning and ends with deployment with the cyclic interaction in between.

- Essential part of the Rational Unified Process, the Dynamic Systems Development Method, Extreme Programming and generally the agile software development frameworks.

- **V Model.**
  - The V-model is a software development process which can be presumed to be the extension of the waterfall model.
  - Instead of moving down in a linear way, the process steps are upwards after the coding phase, to form the typical V shape.
  - The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
  - The horizontal and vertical axes represents time or projectcompleteness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.
    - Verification Phases
    - Requirements analysis
    - System Design
    - Architecture Design
    - *Module Design*

- **Spiral model.**
  - Software development process combining which elements of design and prototyping-in-stages in an effort to combine advantages of top-down and bottom-up concepts.
  - Also known as the spiral lifecycle model (or spiral development).
  - Systems development method (SDM) used in information technology (IT).
  - This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration matters.

  - **Steps**
    - The system requirements are defined
    - A preliminary design is created for the new system.
    - First prototype of the new system is constructed from the preliminary design.
    - A second prototype is evolved by a fourfold procedure:
    - Evaluating the first prototype in terms of its strengths, weaknesses, and risks;
    - Defining the requirements of the second prototype;
    - Planning and designing the second prototype;
    - Constructing and testing the second prototype.

- **Big Bang Model.**
  - Adoption type of the instant changeover
  - There are three different ways to adopt this new system:
    - The big bang adoption, phased adoption and parallel adoption.
    - In parallel adoption the old and the new system are running parallel.
    - Phased adoption means that the adoption will happen in several phases, so after each phase the system is a little nearer to be fully adopted.
    - With the big bang adoption, the switch between using the old system and using the new system happens at one single date
    - The big bang adoption type is riskier than other adoption types

➢ PROTOTYPING MODEL

- The developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take.
- In these, and many other situations, a *prototyping paradigm* may offer the best approach.
- The prototyping paradigm begins with requirements gathering.
- Developer and customer meet and define the overall objectives for the software.
- A "quick design" then occurs.
- The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user.
- The quick design leads to the construction of a prototype.
- The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed.

AUTOMATED TESTING

- Introduction
  (Concept of Freeware, Sharewere, Licensed Tools)

- Theory and Practical case study of Testing Tools
  (Win runner,Load runner, QTP, Rational Suite)

- **Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes,** the settingup of test preconditions, and other test control and test reporting functions. Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.
- There are two general approaches to test automation:
  - ***Code-driven testing.*** The public (usually) interfaces to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.
  - ***Graphical user interface testing.*** A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.
- **Test automation tools can be expensive,** and it is usually employed in combination with manual testing. It can be made cost-effective in the longer term, especially when used repeatedly in regression testing.

- **Freeware**
  - **Freeware (from "free" and "software") is computer software that is available for use at no cost or for an optional fee.**
  - The term freeware was coined by Andrew Fluegelman when he wanted to sell a communications program named PC-Talk that he had created but for which he did not wish to use traditional methods of distribution because of their cost. Fluegelman actually distributed PC-Talk via a process now referred to as shareware. Current use of the term freeware does not necessarily match the original concept by Andrew Fluegelman.
  - Software classified as freeware is normally fully functional for an unlimited time with no cost, monetary or otherwise. **Freeware can be proprietary software available at zero price.** The author usually restricts one or more rights to copy, distribute, and make derivative works of the software.
  - **The software license may impose restrictions** on the type of use including personal use, individual use, non-profit use, non-commercial use, academic use, commercial use or any combination of these. For instance, the license may be "free for personal, non-commercial use".
  - Accordingly, freeware may or may not be free and open source software and, in order to distinguish, the Free Software Foundation asks users to avoid calling "freeware" free software.
  - **The principal difference being that free software can be used, studied, and modified without restriction**; free software embodies the concept of "free speech" while freeware that of "free beer". Freeware is also different from shareware; the latter obliges the user to pay after some trial period or to gain additional functionality.
- **Shareware**
  - The term shareware, popularized by Bob Wallace, refers to proprietary software that is provided to users without payment on a trial basis and is often limited by any combination of functionality, availability or convenience.
  - Shareware is often offered as a download from an Internetwebsite or as a compact disc included with a periodical such as a newspaper or magazine. The aim of shareware is to give

buyers the opportunity to use the program and judge its usefulness before purchasing a license for the full version of the software.

- **Shareware is usually offered as a trial version with certain features only available after the license is purchased,** or as a full version, but for a trial period. Once the trial period has passed the program may stop running until a license is purchased.
- Shareware is often offered without support, updates, or help menus, which only become available with the purchase of a license. **The words "free trial" or "trial version" are indicative of shareware.**
- The term shareware is used in contrast to retail software, which refers to commercial software available only with the purchase of a license which may not be copied for others, public domain software, which refers to software not copyright protected, and freeware, which refers to copyrighted software for which the author solicits no payment (though he or she may request donations).

- **License**
  - **The verb license or grant license means to give permission.** The noun license (licence in British and Canadian spelling) refers to that permission as well as to the document memorializing that permission. License may be granted by a party ("licensor") to another party ("licensee") as an element of an agreement between those parties. A shorthand definition of a license is "an authorization (by the licensor) to use the licensed material (by the licensee)."
  - **A licensor may grant license under intellectual property laws to authorize a use (such as copying software or using a (patented) invention) to a licensee, sparing the licensee from a claim of infringement brought by the licensor.**
  - A license under intellectual property commonly has several component parts beyond the grant itself, including a term, territory, renewal provisions, and other limitations deemed vital to the licensor.
    - ***Term:* many licenses are valid for a particular length of time**. This protects the licensor should the value of the license increase, or market conditions change. It also preserves enforceability by ensuring that no license extends beyond the term of IP ownership.
    - ***Territory:* a license may stipulate what territory the rights pertain to**. For example, a license with a territory limited to "North America" (United States/Canada) would not permit a licensee any protection from actions for use in Japan.
    - ***Mass licensing of software:***Mass distributed software is used by individuals on personal computers under license from the developer of that software. Such license is typically included in a more extensive **end-user license agreement (EULA)** entered into upon the installation of that software on a computer.
    - Under a typical end-user license agreement, the user may install the software on a limited number of computers.
    - The enforceability of end-user license agreements is sometimes questioned.
    - ***Trademark and brand licensing***
    - **A licensor may grant permission to a licensee to distribute products under a trademark.** With such a license, the licensee may use the trademark without fear of a claim of trademark infringement by the licensor.
    - ***Artwork and character licensing***

- **A licensor may grant permission to a licensee to copy and distribute copyrighted works such as "art"** (e.g., Thomas Kincaid's painting "Dawn in Los Gatos") **and characters** (e.g., Mickey Mouse). With such license, a licensee need not fear a claim of copyright infringement brought by the copyright owner.
- Artistic license is, however, not related to the aforementioned license. It is a euphemism that denotes approaches in art works where dramatic effect is achieved at the expense of factual accuracy.

- **SUMMARY:-**
- **Freeware**
  - Freeware (from "free" and "software") is computer software that is available for use at no cost or for an optional fee.
  - The term freeware was coined by Andrew Fluegelman
  - Software classified as freeware is normally fully functional for an unlimited time with no cost.
  - Freeware can be proprietary software available at zero prices.
  - The author usually restricts one or more rights to copy, distribute, and make derivative works of the software.
  - The principal difference being that free software can be used, studied, and modified without restriction;

- **Shareware**
  - The term shareware, popularized by Bob Wallace.
  - Shareware is often offered as a download from an Internetwebsite or as a compact disc included with a periodical such as a newspaper or magazine.
  - The aim of shareware is to give buyers the opportunity to use the program and judge its usefulness before purchasing a license.
  - Shareware is usually offered as a trial version with certain features only available after the license is purchased
  - Shareware is often offered without support, updates, or help menus, which only become available with the purchase of a license.
  - The words "free trial" or "trial version" are indicative of shareware.

- **License**
  - The verb license or grant license means to give permission.
  - The noun license refers to that permission as well as to the document memorializing that permission.
  - License may be granted by a party ("licensor") to another party ("licensee") as an element of an agreement between those parties.
  - A license is "an authorization (by the licensor) to use the licensed material (by the licensee)."
  - A license under intellectual property commonly has several component parts beyond the grant itself, including a term, territory, renewal provisions, and other limitations deemed vital to the licensor.

- **Theory and Practical case study of Testing Tools**
  - **Win Runner:**

- HP/Mercury Interactive's WinRunner is an **automated functional GUI testing tool** that allows a user to record and play back UI interactions as test scripts.
- As a Functional test suite, it works together with HP QuickTest Professional and supports enterprise quality assurance.
- WinRunner is **functional testing software for enterprise IT applications**. It captures, verifies and replays user interactions automatically, so you can identify defects and determine whether business processes work as designed.
- The software implements a proprietary **Test Script Language (TSL) that** allows customization and parameterization of user input.
- HP WinRunner's intuitive **recording process** helps produce robust functional tests. To create a test, HP WinRunner simply records a typical business process by emulating user actions, such as ordering an item or opening a vendor account. During recording, it is possible to directly edit generated scripts to meet the most complex test requirements. Next, testers can **add checkpoints,** which compare expected and actual outcomes from the test run. HP WinRunner offers a variety of checkpoints, including test, GUI, bitmap and web links.
- HP WinRunner can **also verify database values to determine** transaction accuracy and database integrity, highlighting records that have been updated, modified, deleted and inserted. With a few mouse clicks, the DataDriver Wizard feature lets convert a recorded business process into a data driven test that reflects the real-life actions of multiple users. For further test enhancement, the Function Generator feature presents a quick and reliable way to program tests, while the Virtual Object Wizard feature permits to teach HP WinRunner to recognize, record and replay any unknown or custom object.
- As HP WinRunner **executes tests, it operates the application automatically,** as though a real user were performing each step in the business process. If test execution occurs after hours or in the absence of a quality assurance (QA) engineer, the Recovery Manager and **Exception Handling mechanisms automatically troubleshoot** unexpected events, errors and application crashes so that tests can complete smoothly. Once tests are run, HP WinRunner's interactive reporting tools help interpret results by providing detailed, easy-to-read reports that list errors and their originations.
- HP WinRunner is **able to build reusable tests to repeat throughout an application's lifecycle.** Thus, if developers modify an application over time, testers do not need to modify multiple tests. Instead, they can apply changes to the Graphical User Interface (GUI) Map, a central repository of test-related information, and HP WinRunner automatically propagates changes to all relevant script.

- **Load Runner:**
  - **LoadRunner** is a **performance and load testing product** by Hewlett-Packard (since it acquired Mercury Interactive in November 2006) for examining system behaviour and performance, while generating actual load. LoadRunner can emulate hundreds or thousands of concurrent users to put the application through the rigors of real-life user loads, while collecting information from key infrastructure components (Web servers, database servers etc). **The results can then be analysed in detail, to explore the reasons for particular behaviour.**
  - Consider the **client-side application for an automated teller machine (ATM).** Although each client is connected to a server, in total there may be hundreds of ATMs open to the public. There may be some peak times — such as 10 a.m. Monday, the start of the work week — during which the load is much higher than normal. In order to test such situations, it is not practical to have a testbed of hundreds of ATMs. So, given an ATM simulator and a computer system with LoadRunner, one can

simulate a large number of users accessing the server simultaneously. Once activities have been defined, they are repeatable. **After debugging a problem in the application, managers can check whether the problem persists by reproducing the same situation, with the same type of user interaction.**

- Working in LoadRunner involves using three different tools which are part of LoadRunner. ***They are Virtual User Generator (VuGen), Controller and Analysis.***

- **Virtual User Generator**
  - **The Virtual User Generator (VuGen) allows a user to record and/or script the test to be performed against the application under test**, and enables the performance tester to play back and make modifications to the script as needed. Such modifications may include Parameterization (selecting data for keyword-driven testing), Correlation and Error handling.
  - LoadRunner supports several protocols like Web HTTP/HTTPS, Remote Terminal Emulator, Oracle and Web Services. **A protocol can be understood as a communication** medium between the clients and the server. For example an AS400 or Mainframe based application use Terminal Emulator to talk to the Server where as a Web Online banking application uses HTTP/HTTPS with some Java and Web services. LoadRunner is capable of recording scripts in both single and multi-protocol modes.
  - During recording, **VuGen records a tester's actions** by routing data through a proxy. The type of proxy depends upon the protocol being used, and affects the form of the resulting script. For some protocols, various recording modes can be selected to further refine the form of the resulting script. For instance, there are two types of recording modes used in LoadRunner Web/HTTP testing: URL based, and HTML based.

- Controller
  - **Once a script is prepared in VuGen, it is run via the Controller**. LoadRunner provides for the usage of various machines to act as Load Generators. For example, to run a test of 1000 users, we can use three or more machines with a LoadRunner agent installed on them.
  - These machines are known as Load Generators because the actual load will be generated from them**. Each run is configured with a *scenario*, which describes which scripts will run, when they will run, how many virtual users will run, and which Load Generators will be used for each script**. The tester connects each script in the scenario to the name of a machine which is going to act as a Load Generator, and sets the number of virtual users to be run from that Load Generator.
  - LoadRunner uses *monitors* during a load test to monitor the performance of individual components under load. Some monitors include Oracle monitors, WebSphere monitors, etc... Once a scenario is set and the run is completed, the result of the scenario can be viewed via the Analysis tool.

- **Analysis**
  - This tool **takes the completed scenario result and prepares the necessary graphs for the tester to view.** Also, graphs can be merged to get a good picture of the performance. The tester can then make needed adjustments to the graph and prepare a LoadRunner report. The report, including all the necessary graphs, can be saved in several formats, including HTML and Microsoft Word format.

- **QTP(Quick Test Professional)**

- **Quick Test Professional** (QTP) is an **automated functional Graphical User Interface** (GUI) testing tool created by the HP subsidiary Mercury Interactive that allows the automation of user actions on a web or client based and desktop computer application. **It is primarily used for functionalregressiontest automation**. QTP uses a scripting language built on top of VBScript to specify the test procedure, and to manipulate the objects and controls of the application under test.
- As part of a functional test suite, it works together with Mercury Interactive WinRunner and HP Quality Center and supports enterprise Quality Assurance.

- ***Record and playback***
  - Initial development of automated tests with QTP is usually done by record-and-playback. A user's actions are recorded and transposed into comprehensible actions using VBScript. Once recorded, the scripts are editable in either Keyword View or Expert View.
  - **To execute, users select the playback button, which re-executes the commands against the application under test.** In real world usage, simply recording and playing-back actions is generally not valuable, as it simply repeats a test already executed and may no longer be valid (because the record now exists in the system, for example).
  - This **record/playback behavior is not unique to QTP**, but is shared by comparable automated functional testing tools, such as IBM Rational Functional Tester, MicroFocus TestPartner, and Borland SilkTest. There are a few other tools which supports "capture & replay" options (with a difference of supported technologies) like free Selenium.

- ***Verification***
  - **Checkpoints are a feature used for verifying that the application under test** functions as expected. One can add a checkpoint to check if a particular object, text or a bitmap is present in the automation run. **Checkpoints are used to verify** that during the course of test execution, the actual application **behavior or state is consistent** with the expected application behavior or state.
  - There are 10 types of checkpoints available in QTP, enabling users to verify various aspects of an application under test, such as: the properties of an object, data within a table, records within a database, a bitmap image, or the text on an application screen. Users can also create user-defined checkpoints.

- ***Exception handling***
  - Recovery is the name for exception handling in QTP, with the goal of enabling the tests to continue to run if an unexpected failure occurs. For instance if an application crash occurs and a message dialog appears, QTP can be instructed to attempt to restart the application and continue with the rest of the test cases from there. Because QTP hooks into the memory space of the applications being tested, some exceptions may cause QTP to terminate, and may be unrecoverable.

- ***Data-driven testing***

  - QTP has features to enable users to perform data-driven testing. For example, data can be output to a data table for reuse elsewhere. **Data-driven testing is implemented as a Microsoft Excel workbook that can be accessed from within QTP.There are two types of Data Tables available in QTP: the Global data sheet and the local data sheets.** The test steps read data from these data tables in order to (for example) drive variable data into the application under test, and verify the expected result.

- ***Automating custom and complex UI objects***

  - Customized user interface objects and other complex objects may not be recognized properly by QTP. QTP offers a Virtual Object concept to enable users to add some degree of support for these objects. Assuming that the required information can be extracted from the object, this allows the users to successfully record and playback against that object. In practice, this is not always possible.

- ***Add-in Extensibility***

  - QuickTest add-in extensibility, available for some environments, enables you to extend the relevant QuickTest add-in to support third-party and custom controls that are not supported out-of-the-box. QuickTest add-in extensibility is currently supported for the Web, .NET, Java, and Delphi add-ins

- ***Results***

  - QTP generates the result file for the test cases at the end of test in the form of XML tree. The result file provides detail regarding PASS and FAILS counts, error messages, and may provide supporting information that allows users to determine the underlying cause of a failure. Frequently, however, users may need to re-execute the test case and observe the failure directly.

- ***Quality Center Integration***

  - Using **QuickTest Professional together with Quality Center provides you an intuitive and efficient system for managing your tests and their resources** (actions, function libraries, object repositories, recovery scenarios, data table files, and environments variables). You can manage asset dependencies and versions, schedule and run tests, collect, analyze, and share results, report defects, and link your tests and defects to project requirements.

- **User interface**
  - QuickTest provides two main views of a script: **Keyword View and Expert View**. They are selected from tabs at the bottom of the QuickTest Professional window.

  - ***Keyword view***
    - **Keyword View is QTP's default test procedure interface.** It displays the automation steps of a test procedure as a descriptive tree of actions and functions. The tree contains columns listing the action or function name, parameters, and comments. This mode is useful for the beginners. This view allows the user to select the objects either from the application or from the Object Repository and the user can select the methods to be performed on those objects. **The script is automatically generated**. Users can also set checkpoints from the keyword view. Users without technical knowledge may be able to understand the Keyword View, but more experienced users and users needing to perform more complicated actions may need to switch to Expert View.

  - ***Expert view***
    - In Expert View, **QTP allows display and editing of the test's source code using VBScript**. All test actions can be edited here except for the root Global action. Expert View acts as an IDE for the test. It includes many standard IDE features, such as breakpoints.

- ***Languages***

    - QTP uses **VBScript as its scripting language**. VBScript supports classes but not polymorphism and inheritance. Compared with Visual Basic for Applications (VBA), VBScript lacks the ability to use some Visual Basic keywords, does not come with an integrated debugger, lacks an event handler, and does not have a forms editor. HP has added some of these features to QTP, such as a debugger, but QTP's functionality is more limited in these areas compared with testing tools that integrate a full-featured IDE, such as those provided with VBA, Java, or VB.NET.

- ***Drawbacks***

    - QTP is **not supported by non-Windows based applications**. Neither can it be used by a plug-in in other environments. It fetches objects like ActiveX from the Windows environment which is not possible in any other OS. It also cannot be used via Remote Desktop Connection due to licensing issues.

- **Rational Test Suite**
    - **IBM Rational is a probably one of few organization,** which touches entire life cycle of software development with its tool set. There are different tools from Rational right from Requirements management to Change Management, Testing and Project/ Portfolio Management.
    - Rational Test Suite is the entire test suite by Rational for supporting automated software quality. This test Suite Consists of different tools:
        - Rational Test Manager
        - Rational Robot
        - Rational Functional Tester
        - Rational Manual Tester
        - Rational Administrator
    - Purpose of this page is to **make you familiar with the Rational Test Suite** and not with the individual products. There will be plenty of information about the individual tools as well in some time. Rational Administrator is the basic tool for managing the authentication for the different repositories of Rational Test Suite. It allows you to create different users with different levels of access. Main purpose of Rational Administrator is to manage authentication and authorization for different Rational Projects.
    - **Rational Test Manager is the backbone for managing all your Test Assets. It is central console for**

        - **Managing test activity**
        - **Executing**
        - **Reporting**
    - Test manager acts as baton for guiding all your testing efforts. It allows you to store all the test cases. These test cases may be linked to the requirements in the Requisite Pro and Automated Scripts in Rational Robot.
    - If test cases are connected to the Automated Test Scripts, they are organized in the form of test suites which can be triggered for execution either on the same machine or the different machines for remote executions with the help of Test Agent.

- After the execution of the Scripts the results are reported in the form of error logs and graphs for the suites depicting the percentage VP's passed and failed. **Rational Robot can be used for automating the test cases**.
- It uses SQABasic scripting language which is similar to VBScript. Robot has the concept of header files and library files. It allows modularization of your automation by breaking it into different functions, which can be stored in the .sbl or .sbh files.
- **Robot allows you to access the windows API by registering their dlls.** You can also create your custom dlls and make calls to them from within the scripts. It has support for command line invocation and web/load testing in the form VU Scripting. Rational Manual Tester allows you to organize your manual testing.
- In manual Tester you can create different steps and specify some Verification Steps. All these steps are saved and you can execute these test cases, mark them for completion. Results of execution are stored and logged in the Test Manager.
- It also has integration with Rational ClearQuest. **ClearQuest can be launched from within the tool and file defects** if found. It also fills some of the field of ClearQuest, based on the knowledge of test cases it has.

UNIT-4

PROJECT ECONOMICS

- Concept of Project Management
- Project Costing based on metrics
- Empirical Project Estimation Techniques
- Decomposition Techniques
- Algorithmic Methods
- Automated Estimation Tools

➢ **Project Management (4 P's)**

- Effective software project management focuses on the **four P's: people, product, process, and project.**
- The order is not arbitrary. The manager who forgets that software engineering work is an intensely human endeavor will never have success in project management.
- A manager who fails to encourage comprehensive customer communication early in the evolution of a project risks building an elegant solution for the wrong problem.
- The manager who pays little attention to the process runs the risk of inserting competent technical methods and tools into a vacuum.
- The manager who embarks without a solid project plan jeopardizes the success of the product.

1 The People

- The cultivation of motivated, highly skilled software people has been discussed since the 1960. In fact, the "people factor" is so important that the Software Engineering institute has developed a people management capability maturity model (PM-CMM), "to enhance the readiness of software organizations to undertake increasingly complex applications by helping to attract, grow, motivate, deploy, and retain the talent needed to improve their software development capability".
- The people management maturity model defines the following key practice areas for software people: recruiting, selection, performance management, training, compensation, career development, organization and work design, and team/culture development.
- Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software engineering practices.
- The PM-CMM is a companion to the software capability maturity model that guides organizations in the creation of a mature software process. Issues associated with people management and structure for software projects are considered later in this chapter.

2 The Product

- Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered, and technical and management constraints should be identified.
- Without this information, it is impossible to define reasonable (and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.
- The software developer and customer must meet to define product objectives and scope.
- In many cases, this activity begins as part of the system engineering or business process engineering and continues as the first step in software requirements analysis.
- Objectives identify the overall goals for the product.(from the customer's point of view) without considering how these goals will be achieved. Scope identifies the primary data, functions and behaviors that characterize the product, and more important, attempts to bound these Characteristics in a quantitative manner.
- Once the product objectives and scope are understood, alternative solutions are considered. Although very little detail is discussed, the alternatives enable managers and practitioners to select a "best" approach, given the constraints imposed by delivery deadlines, budgetary restrictions, personnel availability, technical interfaces, and other factors.

3 The Process

- A software process provides the framework from which a comprehensive plan for software Development can be established. A small number of frame work activities are applicable to all software projects, Regardless of their size or complexity.
- A number of different task sets-tasks, milestones, work products and quality assurance points-enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
- Finally, umbrella activities-such as software quality assurance, software configuration management, and measurement-overlay the process model. Umbrella activities are independent of any one framework activity and occur throughout the process.

4 The Project

- We conduct planned and controlled software projects for one primary reason-it is the only known way to manage complexity. And yet, we still struggle.
- In 1998, industry data indicated that 26 percent of software projects failed outright and 46 percent experienced cost and schedule overruns.
- Although the success rate for software projects has improved somewhat, our project failure rate remains higher than it should be. In order to avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a commonsense approach for planning, monitoring and controlling the project.

- **SUMMARY:-**
- **Project Management (4 P's)**

- Project management focuses on the **four P's:**
  - **people**
  - **product**
  - **process**
  - **project**
- The order is not arbitrary.
- The manager who pays little attention to the process runs the risk of inserting competent technical methods and tools into a vacuum.

1 The People

- The "people factor" is so important that the Software Engineering institute has developed a people management capability maturity model (PM-CMM).
- The people management maturity model defines the following key practice areas for software people:
  - Recruiting
  - Selection
  - Performance management
  - Training
  - Compensation
  - Career development
  - Organization and work design
  - Team/culture development.

2 The Product

- Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified.
- Without this information, it is impossible to define reasonable (and accurate) estimates of the cost.
- The software developer and customer must meet to define product objectives and scope.
- Once the product objectives and scope are understood, alternative solutions are considered.

3 The Process

- A software process provides the framework from which a comprehensive plan for software Development can be established.
- A small number of frame work activities are applicable to all software projects, Regardless of their size or complexity.

4 The Project

- We conduct planned and controlled software projects for one primary reason-it is the only known way to manage complexity. And yet, we still struggle.

➢ **Project Cost Estimating**

- In order to manage a successful software project, we must understand what can go wrong (so that problems can be avoided) and how to do it right. in an excellent paper on software

projects, John Reel [REE99] defines ten signs that indicate that an information systems project.

1. Software people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs changed or are ill-defined.
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost lor was never properly obtained
9. The project team lacks people with appropriate skills.
10. Managers [and practitioners] avoid best practices and lessons learned.

- The following are the points of software project.
- For many development projects, the bulk of project costs are tied to staffing. In this case, the best way to estimate project cost is to prepare a detailed project schedule using Microsoft Project (or a similar tool), and to use the resource management features of that software to identify the types, quantities, and phasing of different types of labor.
- Project cost estimating is usually performed by summing estimates for individual project elements into a project total. The pieces can vary in size and number from a few large chunks of a project with known costs to hundreds or thousands of discrete tasks or individual work packages.
- Sometimes project cost goals are a forced-fit to the amount of money available in the budget. This will require the project manager to initiate a cost estimate to find out if the project is feasible. Adjustments in scope may be needed so the project can survive.
- "Design-to-Cost" is a process where cost goals for development, acquisition, or operations and maintenance are used as design parameters, along with technical performance, in the systems design trade-off process. In cases where the absolute value of a dollar threshold needs to be contained, the project definition, conceptual design, and development can address performance trade-offs to fit the project within a predetermined cost envelope.
- "Cost as the Independent Variable" is an affordability based method for planning project scope. It starts with a fixed budget and works backwards, through an iterative process of prioritizing and selecting requirements, to arrive at a project scope achievable within budget constraints.
- Costs can usually be estimated with acceptable accuracy by using relevant historical cost data, a well constructed and documented estimating methodology, and a good understanding of the work content to be performed. This approach involves putting as much

detail into understanding the tasks as possible and generating assumptions with whatever shreds of knowledge may be available.

- In order to manage a successful software project, we must understand what can gowrong (so that problems can be avoided) and how to do it right. In an excellent paperon software projects, John Reel [REE99] defines ten signs that indicate that an information

Systems project:

1. Software people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change [or are ill-defined].
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost [or was never properly obtained].
9. The project team lacks people with appropriate skills.
10. Managers [and practitioners] avoid best practices and lessons learned.

1. Start on the right foot.-

- This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objects and expectations for everyone who will be involved in the project. It is reinforced by building the right team (Section 3.2.3) and giving the team the autonomy, authority, and technology needed to do the job.

2. Maintain momentum:-

- Many projects get off to a good start and then slowly disintegrate. To maintain momentum, the project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior managementshould do everything possible to stay out of the team's way.7

3. Track progress:-

- For a software project, progress is tracked as work products (e.g., specifications, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity. In addition, software process and project measures can be collected and used to assess progress against averages developed for the software development organization.

4. Make smart decisions:-

- In essence, the decisions of the project manager and the software team should be to "keep it simple." Whenever possible, decide to use commercial off-the-shelf software or existing software components,decide to avoid custom interfaces when standard approaches are available, decide to identify and then avoid obvious risks, and decide to allocate more time than you think is needed to complex or risky tasks (you'll need every minute).

5. Conduct a postmortem analysis:-

- Establish a consistent mechanism for extracting lessons learned for each project. Evaluate the planned and actual schedules, collect and analyze software project metrics, get feedback from team members and customers, and record findings in written form.

- **SUMMARY:-**
  - In an excellent paper on software projects, John Reel [REE99] defines ten signs that indicate that an information systems project.
  - For many development projects, the bulk of project costs are tied to staffing.
  - In this case, the best way to estimate project cost is to prepare a detailed project schedule using Microsoft Project (or a similar tool), and to use the resource management features of that software.

Systems project:

1. Software people don't understand their customer's needs.
2. The product scope is poorly defined.
3. Changes are managed poorly.
4. The chosen technology changes.
5. Business needs change [or are ill-defined].
6. Deadlines are unrealistic.
7. Users are resistant.
8. Sponsorship is lost [or was never properly obtained].
9. The project team lacks people with appropriate skills.
10. Managers [and practitioners] avoid best practices and lessons learned.

- **Empirical Project Estimation**
  - Use one or more empirical models for software cost and effort estimation.
  - Empirical estimation models can be used to complement decomposition techniques and offer a potentially valuable estimation approach in their own right. A model is based on experience (historical data) and takes the form

    $d = f(vi)$
  - Where d is one of a number of estimated values (e.g., effort, cost, project duration) and vi are selected independent parameters (e.g., estimated LOC or FP).
  - An estimation model for computer software uses empirically derived formulas to predict effort as a function of LOC or FP.
  - The empirical data that support most estimation models are derived from a limited sample of projects. For this reason, no estimation model is appropriate for all classes of software and in all development environments.

(1) The Structure of Estimation Models

- A typical estimation model is derived using regression analysis on data collected from past software projects. The overall structure of such models takes the form
  - $E = A + B \times (ev)C$ (5-2)
- where A, B, and C are empirically derived constants, E is effort in person-months, andev is the estimation variable (either LOC or FP).

(2) The COCOMO Model

- COnstructive COst MOdel. The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. It has evolved into a more comprehensive estimation model, called COCOMO II.
- Like its predecessor, COCOMO II is actually a hierarchy of estimation models that address the following areas:
  - **Application composition model.**
    - Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
  - **Early design stage model.**
    - Used once requirements have been stabilized and basic software architecture has been established.
  - **Post-architecture-stage model.**
    - Used during the construction of the software. Like all estimation models for software, the COCOMO II models require sizing information.
    - Three different sizing options are available as part of the model hierarchy:
      - object points, function points, and lines of source code.
      -

(3) The Software Equation

- The software equation is a dynamic multivariable model that assumes a specific distribution of effort over the life of a software development project. The model has been derived from productivity data collected for over 4000 contemporary software projects.

  • Overall process maturity and management practices
  • The extent to which good software engineering practices are used
  • The level of programming languages used
  • The state of the software environment
  • The skills and experience of the software team
  • The complexity of the application

- **SUMMARY:-**
  - Use one or more empirical models for software cost and effort estimation.
  - Empirical estimation models can be used to complement decomposition techniques
  - Offer a potentially valuable estimation approach in their own right.
  - A model is based on experience (historical data) and takes the form

    d = f (vi)
  - Where d is one of a number of estimated values (e.g., effort, cost, project duration) and vi are selected independent parameters (e.g., estimated LOC or FP).
  - An estimation model for computer software uses empirically derived formulas to predict effort as a function of LOC or FP.

    (1) The Structure of Estimation Models

    (2) The COCOMO Model
    - COCOMO II is actually a hierarchy of estimation models that address the following areas:

- Application composition model.
- Early design stage model.
- Post-architecture-stage model.

(3) The Software Equation

- **Decomposition Techniques**
  - Problem decomposition, sometimes called *partitioning* or *problem elaboration,* is an activity that sits at the core of software requirements analysis. During the scoping activity no attempt is made to fully decompose the problem.
  - Rather, decomposition is applied in two major areas: (1) the functionality that must be delivered and (2) the process that will be used to deliver it. Both cost and schedule estimates are functionally oriented, some degree of decomposition is often useful. decomposition will make planning easier.
  - The project planner examines the statement of scope and extracts all important software functions. This process, called decomposition.
  - **DECOMPOSITION TECHNIQUES**
    - Software project estimation is a form of problem solving, and in most cases, the problem to be solved is too complex to be considered in one piece. For this reason, we decompose the problem, recharacterizing it as a set of smaller problems.
    - The decomposition approach was discussed from two different points of view: decomposition of the problem and decomposition of the process. Estimation uses one or both forms of partitioning. But before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size."

(1) Software Sizing

a. The accuracy of a software project estimate is predicated on a number of things:

i. The degree to which the planner has properly estimated the size of the product to be built;

ii. The ability to translate the size estimate into human effort, calendar time, and dollars (a function of the availability of reliable software metrics from past projects);

iii. The degree to which the project plan reflects the abilities of the software team;

iv. The stability of product requirements and the environment that supports the software engineering effort.

- Estimation tools The "size" of software to be built can be estimated using adirect measure, LOC, or an indirect measure, FP.
- We consider the software sizing problem. Because a project estimate is only as good as the estimate of the size of the work to be accomplished, sizing represents the project planner's first major challenge.

(2) "Fuzzy logic" sizing

a. To apply this approach, the planner must identify the type of application,Although personal experience can be used, the planner should also

haveaccess to a historical database of projects8 so that estimates can be comparedto actual experience.

(3) Function point sizing

a. The planner develops estimates of the information domain characteristics Standard component sizing. Software is composed of a number of different"standard components" that are generic to a particular application area.

(4) Change sizing

a. This approach is used when a project encompasses the use of existing software that must be modified in some way as part of a project. The planner estimates the number and type (e.g., reuse, adding code, changing code, deleting code) of modifications that must be accomplished.

- **SUMMARY:-**
  - Problem decomposition, sometimes called *partitioning* or *problem elaboration,*
  - Is an activity that sits at the core of software requirements analysis.
  - During the scoping activity no attempt is made to fully decompose the problem.
  - Two major areas:
    - The functionality that must be delivered
    - The process that will be used to deliver it.
  - The project planner examines the statement of scope and extracts all important software functions. This process, called decomposition.
  - **DECOMPOSITION TECHNIQUES**

    (5) Software Sizing
    (6) "Fuzzy logic" sizing
    (7) Function point sizing
    (8) Change sizing

- **Problem-Based Estimation**
  - Lines of code and function points were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation:
    - As an estimation variable to "size" each element of the software.
    - As baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections.
  - The project planner begins with a bounded statement of software scope and from this statement attempts to decompose software into problem functions that can each be estimated individually.
  - LOC or FP (the estimation variable) is then estimated for each function. Alternatively, the planner may choose another component for sizing such as classes or objects, changes, or business processes affected.

- **An Example of LOC-Based Estimation**
  - As an example of LOC and FP problem-based estimation techniques, let us consider a software package to be developed for a computer-aided design application for mechanical components.
  - A review of the System Specification indicates that the software is to execute on an engineering workstation and must interface with various computer graphics peripherals including a mouse, digitizer, high resolution color display and laser printer.
  - Using the System Specification as a guide, a preliminary statement of software scope can be developed.

Function	Estimated LOC
User interface and control facilities (UICF)	2,300
Two-dimensional geometric analysis (2DGA)	5,300
Three-dimensional geometric analysis (3DGA)	6,800
Database management (DBM)	3,350
Computer graphics display facilities (CGDF)	4,950
Peripheral control function (PCF)	2,100
Design analysis modules (DAM)	8,400
Estimated lines of code	*33,200*

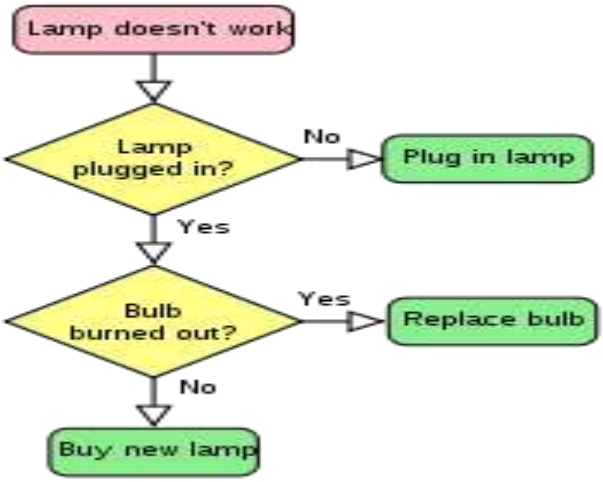
- **An Example of FP-Based Estimation**
  - Decomposition for FP-based estimation focuses on information domain values rather than software functions. Referring to the function point calculation table presented in Figure 5.4, the project planner estimates inputs, outputs, inquiries, files, and external interfaces for the CAD software. For the purposes of this estimate, the complexity weighting factor is assumed to be average.

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of inputs	20	24	30	24	4	97
Number of outputs	12	15	22	16	5	78
Number of inquiries	16	22	28	22	5	88
Number of files	4	4	5	4	10	42
Number of external interfaces	2	2	3	2	7	15
Count total						*320*

- **Process-Based Estimation**
  - The most common technique for estimating a project is to base the estimate on the process that will be used.
  - That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.

- Like the problem-based techniques, process-based estimation begins with a delineation of software functions obtained from the project scope.
- A series of software process activities must be performed for each function.

- **Algorithm**



- This is an algorithm that tries to figure out why the lamp doesn't turn on and tries to fix it using the steps. Flowcharts are often used to graphically represent algorithms.
- In mathematics, computing, linguistics, and related subjects, an algorithm is an effective method for solving a problem using a finite sequence of instructions. Algorithms are used for calculation, data processing, and many other fields.
- Each algorithm is a list of well-defined instructions for completing a task. Starting from an initial state, the instructions describe a computation that proceeds through a well-defined series of successive states, eventually terminating in a final ending state.
- The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate randomness.

- **SUMMARY:-**
  - Lines of code and function points were described as measures from which productivity metrics can be computed.
  - LOC and FP data are used in two ways during software project estimation:
    - As an estimation variable to "size" each element of the software.
    - As baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections.
    - LOC or FP (the estimation variable) is then estimated for each function.

- **AUTOMATED ESTIMATION TOOLS**
  - The decomposition techniques and empirical estimation models described in the preceding sections are available as part of a wide variety of software tools. These automated estimation tools allow the planner to estimate cost and effort and to perform "what-if" analyses for important project variables such as delivery date or staffing. Although many automated

estimation tools exist, all exhibit the same general characteristics and all perform the following six functions:

1. Sizing of project deliverables. The "size" of one or more software workproducts is estimated. Work products include the external representation of software (e.g., screen, reports), the software itself ,functionality delivered (e.g., function points), descriptive information (e.g. documents).

2. Selecting project activities. The appropriate process framework is selected and the software engineering task set is specified.

3. Predicting staffing levels. The number of people who will be available todo the work is specified. Because the relationship between people available and work (predicted effort) is highly nonlinear, this is an important input.

4. Predicting software effort. Estimation tools use one or more models that relate the size of the project deliverables to the effort required to produce them.

5. Predicting software cost. Given the results of step 4, costs can be computed by allocating labor rates to the project activities noted in step 2.

6. Predicting software schedules. When effort, staffing level, and project activities are known, a draft schedule can be produced by allocating labor across software engineering activities based on recommended models for effort distribution.

- **SUMMARY:-**
  - The decomposition techniques and empirical estimation models described in the preceding sections are available as part of a wide variety of software tools.
  - These automated estimation tools allow the planner to estimate cost and effort and to perform "what-if" analyses for important project variables such as delivery date or staffing.
  - Although many automated estimation tools exist, all exhibit the same general characteristics and all perform the following six functions:

    1. Sizing of project deliverables.
    2. Selecting project activities.
    3. Predicting staffing levels.
    4. Predicting software effort.
    5. Predicting software cost.
    6. Predicting software schedules.

PROJECT SCHEDULING AND TRACKING

- Concept of project scheduling and tracking
- Effort estimation techniques
- Timeline Chart
- Pert Chart
- Monitoring and control progress

- **SCHEDULING**
  - *Software project scheduling* is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks.
  - Scheduling of a software project does not differ greatly from scheduling of any multitask engineering effort. Therefore, generalized project scheduling tools and techniques can be applied with little modification to software projects.
  - *Program evaluation and review technique* (PERT) and *critical path method* (CPM) are two project scheduling methods that can be applied to software development.
  - Both techniques are driven by information already developed in earlier project planning activities:

    • Estimates of effort

    • A decomposition of the product function

    • The selection of the appropriate process model and task set

    • Decomposition of tasks

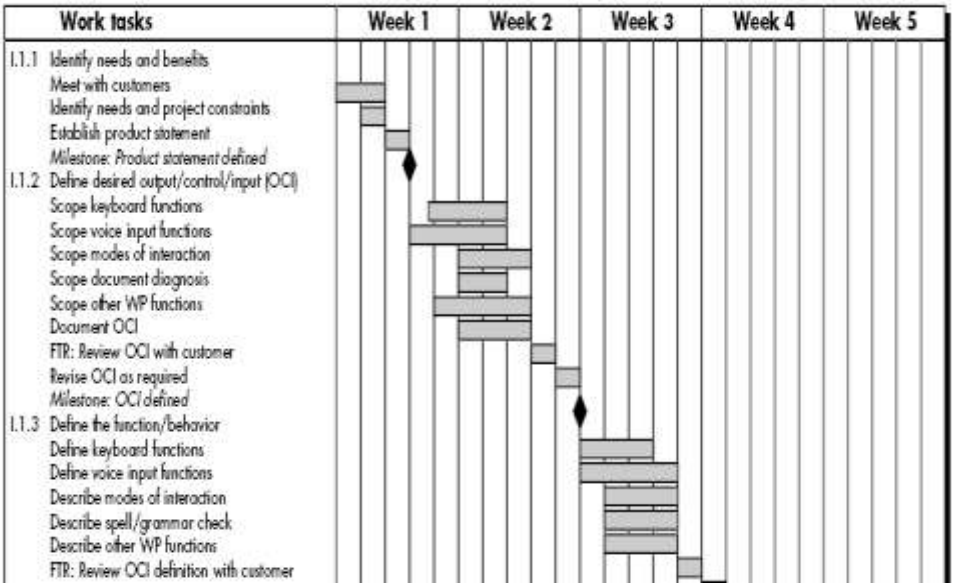
  - Interdependencies among tasks may be defined using a task network. Tasks, sometimes called the project *work breakdown structure* (WBS), are defined for the product as a whole or for individual functions.
  - Both PERT and CPM provide quantitative tools that allow the software planner to
  - (1) determine the *critical path*—the chain of tasks that determines the duration of theproject;
  - (2) establish "most likely" time estimates for individual tasks by applying statisticalmodels; and
  - (3) Calculate "boundary times" that define a time "window" for a particular task.
  - Boundary time calculations can be very useful in software project scheduling. Slippage in the design of one function, for example, can retard further development of other functions.
  - Both PERT and CPM have been implemented in a wide variety of automated tools that are available for the personal computer. Such tools are easy to use and make the scheduling methods described previously available to every software project manager.

- **TRACKING**
  - Although there are many reasons why software is delivered late, most can be traced to one or more of the following root causes:
    - An unrealistic deadline established by someone outside the software development group and forced on managers and practitioners within the group.
    - Changing customer requirements that are not reflected in schedule changes.
    - An honest underestimate of the amount of effort and/or the number ofresources that will be required to do the job.
    - Predictable and/or unpredictable risks that were not considered when theproject commenced.
    - Technical difficulties that could not have been foreseen in advance.
    - Human difficulties that could not have been foreseen in advance.
    - Miscommunication among project staff that results in delays.
    - A failure by project management to recognize that the project is fallingbehind schedule and a lack of action to correct the problem.

- **Effort estimation techniques**

    - The software project estimation is used for problem solving. It solves the problem, of cost, the number of effort, the amount of time for the software project.
    - To solve a project is very complex task.
    - So it will divide into sub task which is known as decomposition.
    - The software cost and effort estimation will never be the perfect but it is based on many variables, technical, environmental, human can effort the software effort, cost and development process.
    - Project estimation is a series of systematic steps that provides estimation with acceptable risk.
    - To achieve reliable cost and effort estimation a number of option arise which are as under.
        - Delay estimation until let in the project.
        - Base estimates on similar project that have already been completed.
        - Use the decomposition technique to generate project cost and effort estimates.
        - Use different software models for software cost and effort estimation.
    - The software will be developed in the mind of user.
    - So, the development team will ask the opinions of different sources during the development process.
    - LOC and FP are used for estimation techniques.
    - The project planner can use LOC and FP data for software project estimation.
    - LOC stands for Line Of Code and FP stands for Function Point.

- **Timeline Charts**
    - When creating a software project schedule, the planner begins with a set of tasks (the work breakdown structure).
    - If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals.
    - As a consequence of this input**, a *timeline chart,* also called a *Gantt chart,* is generated**.
    - A timeline chart can be developed for the entire project. Alternatively, separate charts can be developed for each project function or for each individual working on the project.
    - **All project tasks (for concept scoping) are listed in the left-hand column. The horizontal bars indicate the duration of each task.**
    - When multiple bars occur at the same time on the calendar, task concurrency is implied.
    - The diamonds indicate milestones.
    - Once the information necessary for the generation of a timeline chart has beeninput, the majority of software project scheduling tools produce *project tables*—a tabularlisting of all project tasks, their planned and actual start- and end-dates, and avariety of related information. Used in conjunction with the timeline chart,project tables enable the project manager to track progress.



- **Tracking the Schedule**
  - The project schedule provides a road map for a software project manager. If it hasbeen properly developed, the project schedule defines the tasks and milestones thatmust be tracked and controlled as the project proceeds. Tracking can be accomplishedin a number of different ways:

    • Conducting periodic project status meetings in which each team member reports progress and problems.

    • Evaluating the results of all reviews conducted throughout the software engineering process.

    • Determining whether formal project milestones (the diamonds shown in Figure7.4) have been accomplished by the scheduled date.

    • Comparing actual start-date to planned start-date for each project task listed in the resource table (Figure 7.5).

    • Meeting informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.

    • Using earned value analysis (Section 7.8) to assess progress quantitatively.In reality, all of these tracking techniques are used by experienced project managers.

- **PERT chart (Program Evaluation Review Technique)**
  - A PERT chart is a project management tool used to schedule, organize, and coordinate tasks within a project.
  - PERT stands for *Program Evaluation Review Technique*, a methodology developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program.
  - A similar methodology, the *Critical Path Method* (CPM) was developed for project management in the private sector at about the same time.

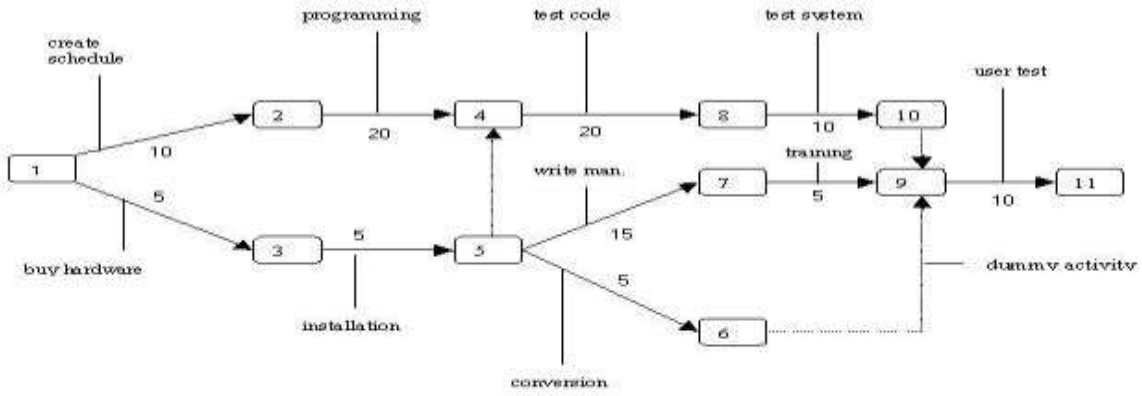


Fig. 1:
PERT Chart

" Numbered rectangles are nodes and represent events or milestones.
" Directional arrows represent dependent tasks that must be completed sequentially.
" Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
" Dotted lines indicate dependent tasks that do not require resources.

- A PERT chart presents a graphic illustration of a project as a network diagram consisting of numbered *nodes* (either circles or rectangles) representing events, or milestones in the project linked by labelled *vectors* (directional lines) representing tasks in the project.
- The direction of the arrows on the lines indicates the sequence of tasks. In the diagram, for example, the tasks between nodes 1, 2, 4, 8, and 10 must be completed in sequence. These are called *dependent* or *serial* tasks.
- The tasks between nodes 1 and 2, and nodes 1 and 3 are not dependent on the completion of one to start the other and can be undertaken simultaneously. These tasks are called *parallel* or *concurrent* tasks.
- Tasks that must be completed in sequence but that don't require resources or completion time are considered to have *event dependency*. These are represented by dotted lines with arrows and are called*dummy activities*.
- For example, the dashed arrow linking nodes 6 and 9 indicates that the system files must be converted before the user test can take place, but that the resources and time required to prepare for the user test (writing the user manual and user training) are on another path.
- Numbers on the opposite sides of the vectors indicate the time allotted for the task.
- The PERT chart is sometimes preferred over the Gantt chart, another popular project management charting method, because it clearly illustrates task dependencies.
- On the other hand, the PERT chart can be much more difficult to interpret, especially on complex projects. Frequently, project managers use both techniques.

- **Monitoring and control progress**

- Monitoring means to be aware with the current status of the system.
- It is also refer as watching.
- When working on the product or documentation at that time staff members should aware with the project process environment.
- It provides stability to the project.
- The software will be developed with the end user mind and developed by software team.
- So the monitoring and controlling is requiring during the development process.

- **Different monitor methods: -**

1. Computer monitor is a device that displays images or symbols generated by computers.

2. Monitor synchronization is an approach where two or more computer task use to share a resource.
3. Machine code monitor software allows users to view or change the memory location or computers.
4. Virtual machine monitor is software which creates virtual computer platform that allows multiple systems to run simultaneously (TURN BY TURN).
5. Risk monitoring is a project tracking activity which is performed for the following reason.
    a. To reduce the effect of risk.
    b. To perform the steps define for the risk and checked that they are properly applied or not
    c. To collect the information that can be used for future risk.

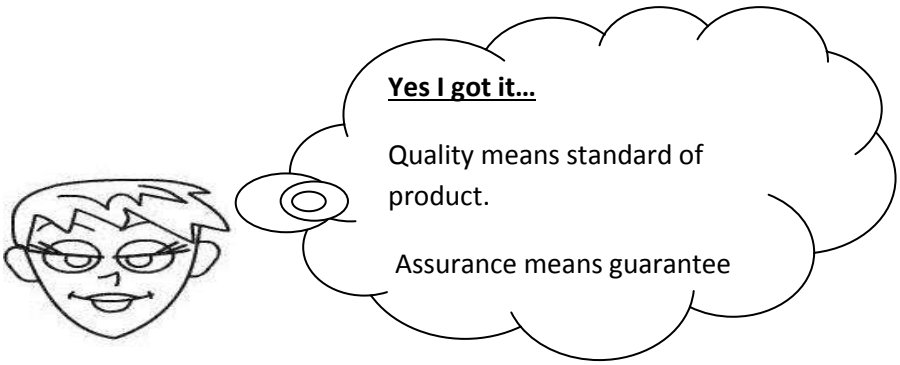
UNIT-5

CONCEPTS OF QUALITY ASSURANCE

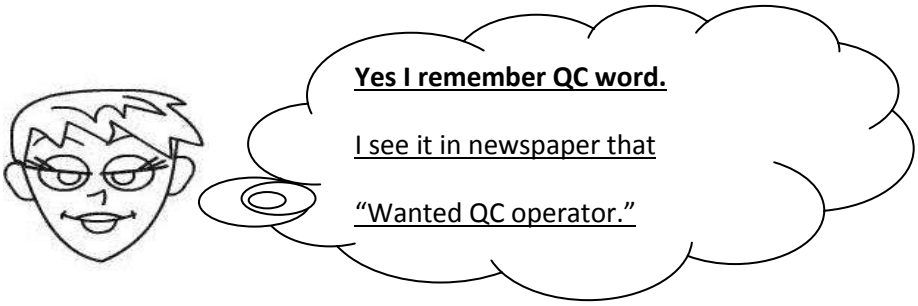
- Introduction of QA
- Quality Control(QC)
- Difference between QA & QCQuality assurance activities

- **Introduction to QA.**
  - **The full form of QA is Quality Assurance.**
  - Now its require to know that what is Quality???????????.....
  - Quality word has many thinking on your mind but one definition tells that quality means **"Characteristic of attribute of something."**
  - As an attribute of an item, quality refers to measurable characteristics— things we are able to compare to known **standards** such as length, color, electrical properties.
  - For example you can buy your shirt with Rs.200 and also you can buy your shirt with Rs.800 also.
  - Why you give more money because you are assured by Co. of shirt that the shirt is best in its category. And the quality is best to use.
  - **Now that's quality and standard**. As per your requirement you can find different quality which is **assured** by different persons.

  - So now question is that what is assurance?????????????????????
    - **Assurance** consists of the auditing and reporting functions of management.
    - **The goal of quality assurance is to provide management with the data necessary to be informed about product quality,** thereby gaining insight and confidence that product quality is meeting its goals.
    - Of course, if the data provided through quality assurance identify problems, it is management's responsibility to address the problems and apply the necessary resources to resolve quality issues.



- **Introduction to QC.**
  - **The full form of QC is Quality control.**
  - QC process includes the Controlling of product that fulfills or satisfied the development process of any product from starting to ending.
  - QC involves **the series of inspections, reviews, andtests of product.**
  - Here you can use QC throughout the software process to ensure each work product meets the customer requirements or not.
  - Remember somewhere you read in newspaper that "Wanted QC operator".



- Quality control includes a **feedback** loop to the process that created the **work product**.
- **Quality control** activities may be fully automated or entirely manual, or a combination of automated tools and human interaction. A key concept of quality control is that **all work products have defined,** measurable specifications to which we may compare the output of each process. The feedback loop is essential to minimize the defects produced
- **Quality assurance** consists of the auditing and reporting functions of management.
- **The goal of quality assurance is to provide management with the data necessary to be informed about product quality,** thereby gaining insight and confidence that product quality is meeting its goals. Of course, if the data provided through quality assurance identify problems.
- It is management's responsibility to address the problems and apply the necessary resources to resolve quality issues.

- **QA v/s QC:-**

(1) QA is a set of activities designed to ensure that the development and/or maintenance Process is adequate to ensure a system will meet its objectives.
While
QC is a set of activities designed to evaluate a developed work product.

(2) QA activities ensure that the process is defined and appropriate.

While
QC activities focus on finding defects in specific task.

(3) Standards development and proper methods are examples of QA activities.
While
Testing is an example of QC activity. (Inspections)

(4) The goal of QA is to provide management with necessary data, & informed about product quality.
While
QC goal is that all work products have defined and compare the output of each process.

(5) QA consists of the auditing and reporting functions of management.
While
QC involves the series of inspections, reviews, and tests used throughout the software process.

CAD PROJECT MANAGEMENT TOOLS

- MS-VISIO for designing & documentation
- MS-Project for controlling and Project Management

- **MS-VISIO**
  - **Microsoft Visio marketed as Microsoft Office Visio, is a diagramming program for Microsoft Windows that uses vector graphics to create diagrams. It is currently available in two editions: Standard and Professional.**
  - **The Standard and Professional editions both share the same interface,** but the latter has additional templates for more advanced diagrams and layouts as well as unique functionality that makes it easy for users to connect their diagrams to a number of data sources and display the information graphically.
  - Microsoft acquired Visio Corporation in 2000. Visio 2007 was released on 30 November 2006. Also released alongside version 2002 were Enterprise Network Tools, an add-on product enabling automated network and directory services diagramming, and the Visio Network Center, a Web site where users could locate the latest network documentation content and exact-replica network equipment shapes from 500 leading manufacturers. The former has been discontinued, while the latter's shape-finding features are now integrated into the program itself.
  - Microsoft has revealed that the next version of Microsoft Visio will feature the ribbon user interface.
  - **Visio began as a standalone product produced by the Visio Corporation**; as of Visio 2000, Microsoft acquired it and branded it as a Microsoft Office application, like Microsoft Project; however, it has never been included in any of the Office suites. Microsoft included a Visio for Enterprise Architects edition with some editions of Visual Studio .NET 2003 and Visual Studio 2005.

- **Different Versions**

  - Visio 1.0 (Standard, Lite, Home)
  - Visio 2.0
  - Visio 3.0
  - Visio 4.0 (Standard, Technical)
  - Visio 4.1 (Standard, Technical)
  - Visio 4.5 (Standard, Professional, Technical)
  - Visio 5.0 (Standard, Professional, Technical)
  - Visio 2000 (6.0; Standard, Professional, Technical, Enterprise), later updated to SP-1 and Microsoft branding after Visio Corporation's acquisition
  - Visio 2002 (10.0; Standard, Professional)
  - Visio Enterprise Network Tools, Visio Network Center
  - Visio for Enterprise Architects 2003 (VEA 2003) (based on Visio 2002 and included with Visual Studio .NET 2003 Enterprise Architect)
  - Office Visio 2003 (11.0; Standard, Professional)
  - Office Visio for Enterprise Architects 2005 (VEA 2005) (based on Visio 2003 and included with Visual Studio 2005 Team Suite and Team Architect editions)
  - Office Visio 2007 (12.0; Standard, Professional).
  - Office Visio 2010 (14.0; Upcoming Version, Currently on Technical Preview)

- **MS-Project.**
  - **Microsoft Project (or MSP) is a project management software program** developed and sold by Microsoft which is designed to assist project managers in developing plans, assigning resources to tasks, tracking progress, managing budgets and analyzing workloads.

- The application **creates critical path schedules, although critical chain and event chain methodology third-party add-ons are available.** Schedules can be resource leveled, and chains are visualized in a Gantt chart. Additionally, Project can recognize different classes of users. These different classes of users can have differing access levels to projects, views, and other data.
- Custom objects such as calendars, views, tables, filters and fields are stored in an enterprise global which is shared by all users.
- Microsoft Project was the company's third Windows-based application, and within a couple of years of its introduction WinProj was the dominant PC-based project management software.
- Although branded as a member of the Microsoft Office family, it has never been included in any of the Office suites before Office 2010 beta 1. It is available currently in two editions, Standard and Professional. MS Project's proprietary file format is .mpp.
- Microsoft Project and Microsoft Project Server are the cornerstones of the Microsoft Office**Enterprise Project Management (EPM) product.** Microsoft has revealed that the next version of Microsoft Project will be featuring the Fluent user interface.

➢ ***Features:***

- Project creates budgets based on assignment work and resource rates. As resources are assigned to tasks and assignment work estimated, the **program calculates the cost** equals the work times the rate, which rolls up to the task level and then to any summary tasks and finally to the project level. Resource definitions (people, equipment and materials) **can be shared between projects using a shared resource pool**. Each resource can have its own calendar, which defines what days and shifts a resource is available.
- Resource rates are used to calculate resource assignment costs which are rolled up and summarized at the resource level. Each resource can be **assigned to multiple tasks in multiple plans and each task can be assigned multiple** resources, and the application schedules task work based on the resource availability as defined in the resource calendars. All resources can be defined in an enterprise-wide resource pool.
- **MS Project presumes additional physical raw materials are always available without limit.**
- Therefore it cannot determine how many finished products can be produced with a given amount of raw materials. This makes MS Project unsuitable for solving problems of available materials constrained production. Additional software is necessary to manage a complex facility that produces physical goods.

SUMMARY:-

- **MS-VISIO**
  - Microsoft Visio marketed as Microsoft Office Visio, is a diagramming program for Microsoft Windows that uses vector graphics to create diagrams.
  - It is currently available in two editions: Standard and Professional.
  - The Standard and Professional editions both share the same interface
  - Microsoft acquired Visio Corporation in 2000.
  - Visio 2007 was released on 30 November 2006.
  - Microsoft has revealed that the next version of Microsoft Visio will feature the ribbon user interface.
  - Visio began as a standalone product produced by the Visio Corporation; as of Visio 2000, Microsoft acquired it and branded it as a Microsoft Office application, like Microsoft Project.
  - Different Versions

    - Visio 1.0 (Standard, Lite, Home)
    - Visio 2.0
    - Visio 3.0
    - Visio 4.0 (Standard, Technical)
    - Visio 4.1 (Standard, Technical)
    - Visio 4.5 (Standard, Professional, Technical)
    - Visio 5.0 (Standard, Professional, Technical)
    - Visio 2000 (6.0; Standard, Professional, Technical, Enterprise), later updated to SP-1 and Microsoft branding after Visio Corporation's acquisition
    - Visio 2002 (10.0; Standard, Professional)
    - Visio Enterprise Network Tools, Visio Network Center
    - Visio for Enterprise Architects 2003 (VEA 2003) (based on Visio 2002 and included with Visual Studio .NET 2003 Enterprise Architect)
    - Office Visio 2003 (11.0; Standard, Professional)
    - Office Visio for Enterprise Architects 2005 (VEA 2005) (based on Visio 2003 and included with Visual Studio 2005 Team Suite and Team Architect editions)
    - Office Visio 2007 (12.0; Standard, Professional).
    - Office Visio 2010 (14.0; Upcoming Version, Currently on Technical Preview)

- **MS-Project.**
  - Microsoft Project (or MSP) is a project management software program developed and sold by Microsoft
  - Designed to assist project managers in developing plans, assigning resources to tasks, tracking progress, managing budgets and analyzing workloads.
  - Although branded as a member of the Microsoft Office family, it has never been included in any of the Office suites before Office 2010 beta 1.
  - It is available currently in two editions, Standard and Professional. MS Project's proprietary file format is .mpp.
  - Microsoft Project and Microsoft Project Server are the cornerstones of the Microsoft OfficeEnterprise Project Management (EPM) product.

UML

- UML design and skill based tools

- Overview of
  - Class Diagram
  - Use Case Diagram
  - Activity Diagram

- UML is **a language for specifying, visualizing, documenting and constructing the artifacts of software systems**, as well as for business modeling and other non-software systems.
- **Goals of UML as stated by the designers are**

    - To model systems using **OO concepts**
    - To establish an **explicit coupling to conceptual as well as executable artifacts**
    - To address the issues of scale inherent in complex, mission-critical systems
    - To create a modeling language usable by humans and machine
    - Provide users a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
    - Provide extensibility and specialization mechanisms to extend the core concepts.
    - Be independent of particular programming languages and development processes.
    - Encourage the growth of the OO tools market.
    - Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
    - Integrate best practices.

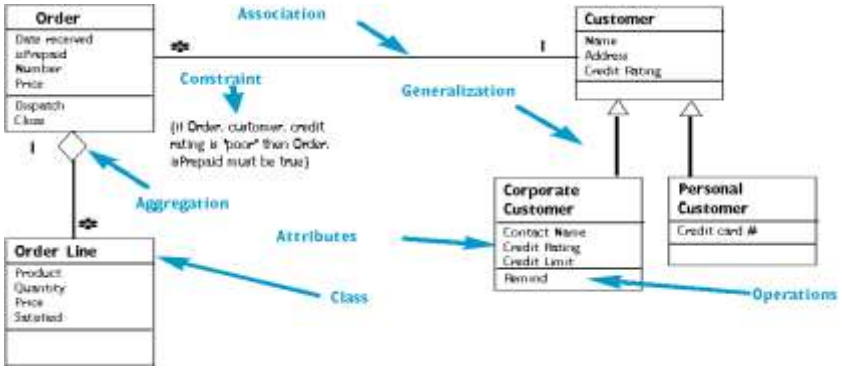
- **The Unified Modeling Language (UML) is a language for**
    - Specifying the **structure and behavior of a system**
    - Visualizing a system as it is or as we want it to be
    - Constructing a system from the template provided by the model
    - Documenting the decisions made
    - UML represents a **collection of best engineering practices** that have proven successful in the modeling of large and complex systems

- **UML Consist Three types of Diagrams**
    - **Class Diagram**
    - **Use Case Diagram**
    - **Activity Diagram**

- **Class Diagrams**
    - **UML class diagrams are the mainstay of object-oriented analysis and design.**
    - UML class diagrams show the **classes of the system**, their **interrelationships** (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling. Although I prefer to create class diagrams on whiteboards because simple tools are more inclusive most of the diagrams that I'll show in this article are drawn using a software-based drawing tool so you may see the exact notation
    - Class diagrams **show the static structure of the model,** in particular, the things that exist (such as classes and types), their internal structure, and their relationships to other things.
    - A class in a class diagram can be directly implemented in an OOP language that has the construct for a class.
    - To create a class diagram, the classes have to be identified and described and when a number of classes exist, they can be related to each other using a number of relationships.
    - Class diagrams **show the static structure of the model**, in particular, the things that exist (such as classes and types), **their internal structure,** and their relationships to other things. Class diagrams do not show temporal information, although they may contain reified

occurrences of things that have or things that describe temporal behavior. An object diagram shows instances compatible with a particular class diagram.

- Class Diagram – Example



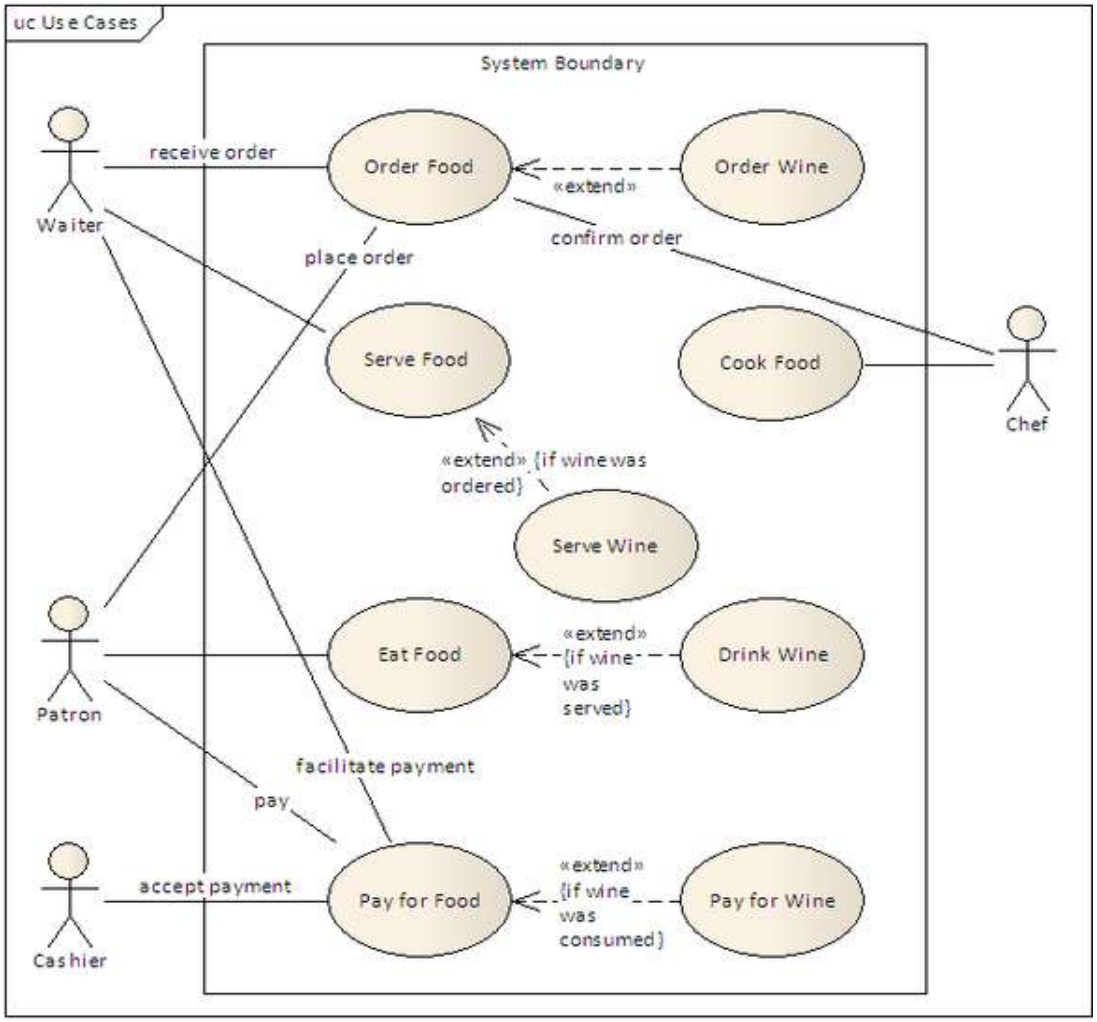
Well Structured Class Diagram.

- Is focused on communicating one aspect of a system
- Contains only elements that are essential to understanding that aspect
- Provides details consistent with its level of abstraction
- Is not so minimalist.
- Has a name that communicates its purpose.

➢ **Use case Diagram:-**

- In software engineering, a use case diagram in the Unified Modeling Language (UML) is **a type of behavioral diagram defined** by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
- The main purpose of a use case diagram is to show **what system functions** are **performed for which actor.Roles of the actors in the system can be depicted.**
- **A Use Case Diagram shows a set of external Actors and their Use Cases connected with communication associations.** The communication associations between the Actors and their Use Cases define the boundary between the system and its external environment. The communication associations may be augmented to show the messages and events exchanged between Actors and Use Cases. Messages and events may be shown through relevant notes attached to specific communication associations.

- **Diagram building blocks**
- **Actor inheritance**
- **Use case relationships**
- **Actor interaction**
  - Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined.
  - Alternatively, interaction among actors can be part of the assumptions used in the use case.



- **Actor Generalization**
  - One popular relationship between Actors is Generalization/Specialization. This is useful in defining overlapping roles between actors. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general actor.
- **Use Case Relationships**
  - Three relationships among use cases are used often in practice

Include

"Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case"[1].

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the **label "«include»".** This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior. There are no parameters or return values.

Extend

This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the

extended use case, with the label **"«extend»".** Notes or constraints may be associated with this relationship to illustrate the conditions under **which this behaviour will be executed**.

Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case. Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal."

Generalization

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may be a specialized form of an existing use case. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case. This resembles the object-oriented concept of sub-classing, in practice it can be both useful and effective to factor out common behaviors, constraints and assumptions to the general use case, describe them once, and deal with it in the same way, except for the details in the specialized cases.

- **Activity Diagram:-**
  - Activity diagrams are a loosely defined diagram**technique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency**. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
  - In SysML the activity diagram has been extended to indicate flows among steps that convey physical element (e.g., gasoline) or energy (e.g., torque, pressure). Additional changes allow the diagram to better support continuous behaviors and continuous data flows.
  - In UML 1.x, an activity diagram is a variation of the UML State diagram in which the "states" represent activities, and the transitions represent the completion of those activities.

Construction

Activity diagrams are typically used for business process modeling. They consist of:

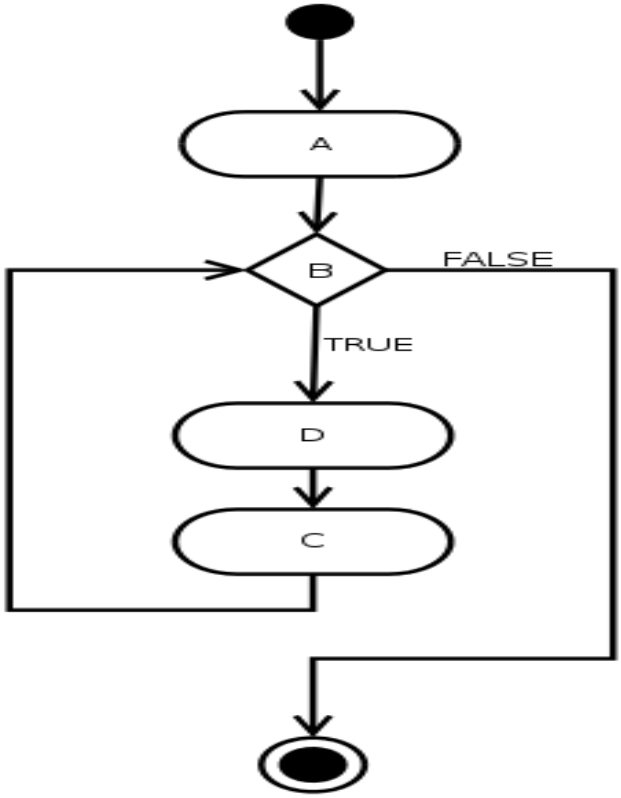
Initial node.

Activity final node.

Activities

The starting point of the diagram is the initial node, and the activity final node is the ending. An activity diagram can have zero or more activity final nodes. In between activities are represented by rounded rectangles.

```
for(A;B;C)
    D;
```



- **SUMMARY:-**
  - UML is a language for specifying, visualizing, documenting and constructing the artifacts of software systems.
  - Goals of UML as stated by the designers are
    - To model systems using OO concepts
    - To establish an explicit coupling to conceptual as well as executable artifacts
    - To create a modeling language usable by humans and machine
    - Provide users a ready-to-use.
    - Encourage the growth of the OO tools market.
    - Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
    - Integrate best practices.
  - The Unified Modeling Language (UML) is a language for
    - Specifying the structure and behavior of a system
    - Visualizing a system as it is or as we want it to be
    - Constructing a system from the template provided by the model
    - Documenting the decisions made
  - UML Consist Three types of Diagrams
    - Class Diagram
    - Use Case Diagram
    - Activity Diagram
  - Class Diagrams
    - UML class diagrams are the mainstay of object-oriented analysis and design.
    - UML class diagrams show the classes of the system, their interrelationships

- Use case Diagram:-
  - A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
  - The main purpose of a use case diagram is to show what system functions are performed for which actor.
  - Roles of the actors in the system can be depicted.
- Activity Diagram:-
  - Activity diagrams are a loosely defined diagramtechnique for showing workflows of stepwise activities and actions, with support for choice, iteration and concurrency.
  - An activity diagram shows the overall flow of control.
  - In UML 1.x, an activity diagram is a variation of the UML State diagram in which the "states" represent activities, and the transitions represent the completion of those activities.