Unit – 01: Introduction to Software Engineering

- Introduction to software engineering.
- Lifecycle.
- Process models:
  - Perspective Process Models.
  - Specialized Process Models.
  - The Waterfall model or Linear Sequential model.
  - The Prototyping Model.
  - Spiral Model.
- Software Engineering as Layered Approach and its characteristics.
- Types of software.
  - Traditional v/s Agile Processes and its importance.
  - Selection criteria for software process model.

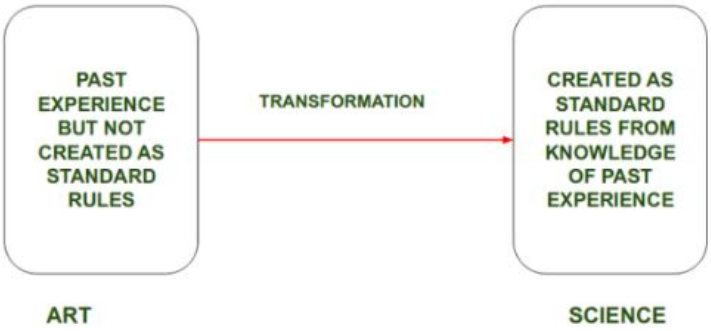
Questions to be discussed:

1. What is software engineering? Is it art, craft or a science? Discuss.
2. Why Software Engineering required? What are the goal of software engineering?
3. What are the component of a software? Discuss how a software differs from a program.
4. What is software life cycle? Discuss the generic waterfall model.
5. Sketch a neat diagram of spiral model of software life cycle.
6. Compare the waterfall model and the spiral model of software development.
7. Difference between Traditional and Agile Software Development.

What is software engineering? Is it art, craft or a science? Discuss.

- It is the process of analyzing user needs and designing a software that will satisfy these needs.
- Software engineering is the application of engineering principles to software development.
- In simple words, we can say that software engineering is the process of designing, developing, and maintaining software systems.
- It is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures.
- The outcome of software engineering is an efficient and reliable software product.

- Thus, like other Engineering Discipline, Software Engineering is a Science that is transformed from an Art.



What are the component of a software? Discuss how a software differs from a program?

Software:

- Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components.
- To produce a software product the set of activities is used.
- This set is called a software process.

Components of Software:

- There are three components of the software:
  1. Program,
  2. Documentation, and
  3. Operating Procedures.

Program: A computer program is a list of instructions that tell a computer what to do.

Documentation: Source information about the product contained in design documents, detailed code comments, etc.

Operating Procedures: Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

Difference between Software and Program:

Software	Program
Software's are mainly dependent on operating system.	Programs are mainly dependent on compiler.
Software's can be a program that generally runs on computer.	Programs cannot be a software.
If software's are not present in computers, then computer is useless.	If programs are not present in computer, then also computer can function well because of operating system.
Software's can be downloaded on computer using internet without any need of program.	Program cannot run on computer without any software present in computer.
It requires more time to create software than program.	It requires less time to create program than software.
Examples of software includes Adobe Photoshop, Google Chrome, PowerPoint, Adobe Reader, etc.	Examples of program includes Web browsers, word processors, video games, etc.

Why is Software Engineering required?

Software Engineering is required due to the following reasons:

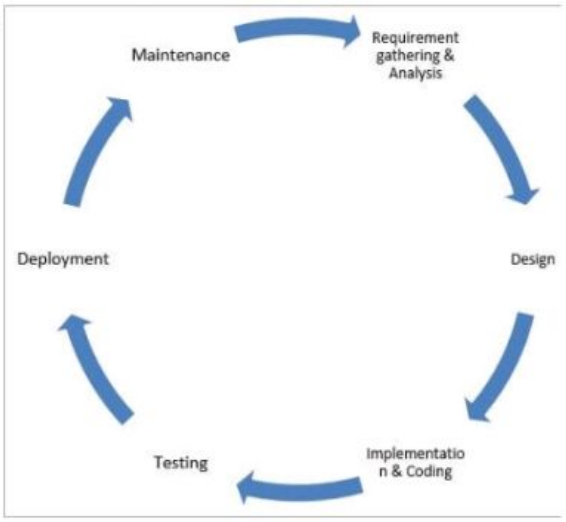
- To manage Large software
- For more Scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

Goals of Software Engineering:

- The goals of software engineering are to produce high-quality software that is reliable, efficient, and easy to maintain.
- Also being developed in a way that is cost-effective and meets the needs of the users.
- Some specific goals of software engineering include:
  - Producing software that is reliable and free from defects
  - Developing software that is efficient and performs well
  - Creating software that is easy to maintain and modify
  - Developing software in a cost-effective manner
  - Meeting the needs of the users.

Software Development Life Cycle (SDLC):

- SDLC stands for Software Development Life Cycle.
- It is a process used by the software industry to design, develop and test high quality software.
- The aims of SDLC is to produce a high-quality software that meets customer expectations.
- It consists of a detailed plan describing how to develop, maintain, replace and alter a specific software.
- There are various phases in SDLC which are given below:
  1. Requirement gathering and analysis
  2. Design
  3. Implementation or coding
  4. Testing
  5. Deployment
  6. Maintenance



Requirement Gathering and Analysis

- During this phase, all the relevant information is collected from the customer to develop a product as per their expectation.

Design:

- In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

Implementation or Coding

- Coding starts once the developer gets the Design document.
- The Software design is translated into source code.

- All the components of the software are implemented in this phase.

Testing

- Testing starts once the coding is complete and the modules are released for testing.
- In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Deployment

- Once the product is tested, it is deployed in the production environment.

Maintenance

- After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

What is a software process model?

- A software process model is an abstraction of the software development process.
- The models specify the stages and order of a process.
- So, think of this as a representation of the order of activities of the process and the sequence in which they are performed.
- A model will define the following:
  1. The tasks to be performed
  2. The input and output of each task
  3. The pre and post-conditions for each task
  4. The flow and sequence of each task

Prescriptive process models:

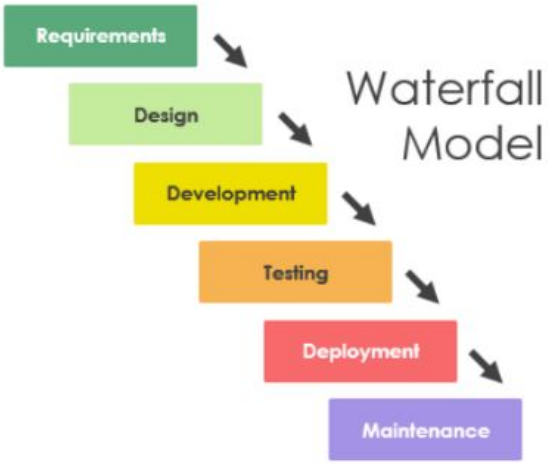
- As we know that the descriptive model describes the history of how a particular software system was developed.
- While a prescriptive model prescribes how a new software system should be developed.
- The name 'prescriptive' is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.
- There are three types of prescriptive process models:
  1. The Waterfall Model
  2. Incremental Process model
  3. RAD model

Types of software process models

- There are many kinds of process models for meeting different requirements.
- The most popular and important SDLC models are as follows:
  1. Waterfall model
  2. Prototype model
  3. Spiral model

Waterfall Model(Linear Sequential Model):

- Waterfall model is the very first model that is used in SDLC.
- The Waterfall model was introduced by Winston w. Royce in 1970.
- It is also known as the linear sequential model or 'Classic life cycle model'.
- In this model, the outcome of one phase is the input for the next phase.
- Development of the next phase starts only when the previous phase is complete.
- This model is used for the small projects.
- It has the following phases:
  1. Requirements
  2. Design
  3. Implementation
  4. Testing
  5. Deployment
  6. Maintenance



Advantages of the Waterfall Model:

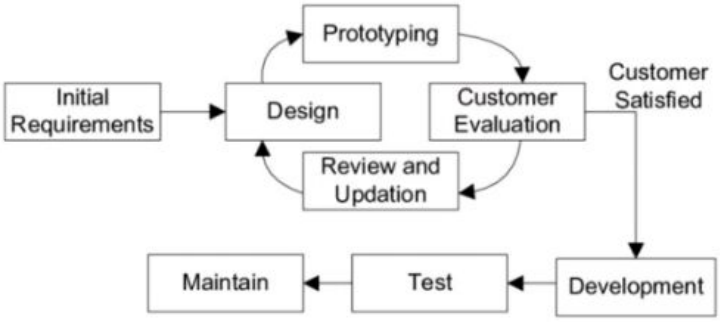
- It is the simple model which can be easily understood.
- In this model all the phases are done step by step.
- Here, each phases are well defined, this leads to no complexity and the project easily manageable.

Disadvantages of Waterfall model:

- Waterfall model is time-consuming & cannot be used in the short duration projects.
- In this model a new phase cannot be started until the ongoing phase is completed.
- It is a poor model for long projects.

Prototype Model:

- Prototype model is a systems development method in which a prototype is built.
- It requires that before development of actual software, a working prototype of the system should be built.
- Dummy functions are used to create prototypes.
- This is a valuable mechanism for understanding the customers' needs.
- Feedbacks are implemented and the prototype is again reviewed by the customer for any change.
- This process goes on until the model is accepted by the customer.
- Once the customer approves the prototype, it is used as a requirement for building the actual software.
- The actual software is build using the Waterfall model approach.



PROTO TYPE MODEL

Advantages of Prototype Model:

- Prototype model reduces the cost and time of development as the defects are found much earlier.
- Missing features can be identified in the evaluation phase.
- Involvement of a customer from the initial stage reduces any confusion in the requirement.

Disadvantages of Prototype Model:

- Since the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product.

Spiral Model:

- It is a SDLC model that provides a systematic and iterative approach to software development.
- Spiral Model is a risk-driven model, that means it focus is on managing risk through multiple iterations.
- The Spiral Model includes iterative and prototype approach.
- The spiral model, proposed by Barry Boehm in 1986.
- Spiral model phases are followed in the iterations.
- Spiral Model has four phases:
  1. Planning
  2. Risk Analysis
  3. Engineering
  4. Evaluation

Advantages of Spiral Model:

- Risk Analysis is done extensively using the prototype models.
- Any enhancement or change in the functionality can be done in the next iteration.

Disadvantages of Spiral Model:

- The spiral model is best suited for large projects only.
- The cost can be high as it might take a large number of iterations which can lead to high time to reach the final product.

Compare the waterfall model and the spiral model of software development:

Waterfall Model	Spiral Model
The Waterfall model is simple and easy.	The spiral model is more complex.
The waterfall model works in a sequential method.	While the spiral model works in the evolutionary method.
The waterfall model is adopted by customers.	While the spiral model is adopted by developers.
The waterfall model is applicable for small projects.	While the Spiral model is used for large projects.
There is high amount risk in waterfall model.	There is low amount risk in spiral model.
Waterfall model is comparatively inexpensive.	While cost of spiral model is very expensive.
It requires least maintenance.	It requires typical maintenance.

Traditional Software Development:

- Traditional software development is the software development process used to design and develop simple software.
- It is used when the security and many other factors of the software are not much important.
- It is used by freshers to develop the software.
- It consists of five phases:
  1. Requirements analysis
  2. Design
  3. Implementation
  4. Coding and Testing
  5. Maintenance

Agile Software Development:

- Agile software development is the software development process used to design complicated software.
- It is used when the software is quite sensitive and complicated.
- It is used when security is much more important.
- It is used by professionals to develop the software.
- It consists of three phases:
  1. Project initiation
  2. Sprint planning
  3. Demos

Difference between Traditional and Agile Software Development:

Traditional Software Development	Agile Software Development
It is used to develop simple software.	It is used to develop complicated software.
In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
It follows a linear organization structure.	It follows an iterative organizational structure.
It provides less security.	It provides high security.
Client involvement is less.	Client involvement is high.
It supports a fixed development model.	It supports a changeable development model.
It is used by freshers.	It is used by professionals.
Development cost is less using this methodology.	Development cost is high using this methodology.
It majorly consists of five phases.	It consists of only three phases.

Selection criteria for software process model:

- Choosing the right software process model for your project can be difficult.
- If you know your requirements well, it will be easier to select a model that best matches your needs.
- You need to keep the following factors in mind when selecting your software process model:
  1. Project requirements
  2. Project size
  3. Project complexity
  4. Cost of delay
  5. Customer involvement
  6. Familiarity with technology
  7. Project resources

Unit – 02: Software Requirement Engineering

- Requirement Gathering and Analysis.
- Types of Requirements.
- Requirement Elicitation.
- Software Requirement Specification.
- Need of SRS and its Characteristics.

Questions to be discussed:

1. Describe the various steps of requirement engineering. Is it essential to follow these steps?
2. What do you mean by "Requirement Elicitation"? Discuss any two techniques in detail.
3. What is SRS? List out the advantage of SRS.

Unit – 02: Software Requirement Engineering

What is requirements engineering?

- Clients have a long list of goals when creating new software.
- Requirement Engineering provides a framework for meeting those goals.
- It is the process of defining, documenting and maintaining the requirements.
- Here, Important needs are transformed into a detailed set of requirements and creates a blueprint.
- The terms "requirements management" and "requirements engineering" are often used interchangeably, but they are different.
- RE transforms a real-world problem into a highly functional software solution.

Steps of requirement engineering:

1. Requirement Elicitation and Analysis
2. Requirement Specification
3. Verification and validation
4. Requirement Management

Unit – 02: Software Requirement Engineering

Elicitation requirements:

- Elicitation is about becoming familiar with all the important details involved with the project.
- The customer will provide details about their needs and furnish critical information.
- You will study those details and also become familiar with similar types of software solutions.
- This step provides important context for development.

Requirements specification:

- During the specification phase, you gather functional and nonfunctional project requirements.
- A variety of tools are used during this stage, including data flow diagrams, to add more clarity to the project goals.

Verification and validation:

- Verification ensures the software is built to the customer's requirements.
- In contrast, validation ensures the software is implementing the right functions.

Requirements management:

- In RM, you're matching all the relevant processes to their requirements.
- You will analyze the requirements – and communicate with relevant stakeholders.
- Any requirements that need modification are handled in an efficient and systematic manner.

Unit – 02: Software Requirement Engineering

Types of Requirement:

- There are three types of software requirements as follows:
  1. Functional requirements
  2. Non-Functional requirements
  3. Domain requirements

Functional Requirements:

- In this the end user demands as basic facilities that the system should offer.
- It can be a calculation, data manipulation, user interaction, or any other specific functionality.
- They are the requirements stated by the user which one can see directly in the final product.
- For example, in a hospital management system, a doctor should be able to retrieve the information of his patients.

Non-functional requirements:

- Non-functional requirements, not related to the system functionality.
- They are also called non-behavioral requirements.
- They basically deal with issues like:
  - Security
  - Maintainability
  - Reliability
  - Performance
  - Flexibility

Domain requirements:

- It is the requirements which are characteristic of a particular category or domain of projects.
- Domain requirements can be functional or nonfunctional.
- For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement.

Unit – 02: Software Requirement Engineering

What is Requirement Elicitation?

- It is all about obtaining information from stakeholders.
- Once the business analysis has communicated with stakeholders for understanding their requirements, it can be described as elicitation.
- Requirement Elicitation is also known as "requirement gathering".
- It is an important part of requirements management for understanding of the project.
- It can be done by communicating with stakeholders directly or by doing some research, brainstorming or survey etc.

Requirements Elicitation Techniques:

There are many elicitation techniques some of them are given below:

1. Stakeholder Analysis
2. Brainstorming
3. Interview
4. Survey

Unit – 02: Software Requirement Engineering

Stakeholder Analysis

- Stakeholders can include team members, customers, any individual who is impacted by the project or it can be a supplier.

Brainstorming:

- This technique is used to generate new ideas and find a solution for a specific issue.
- The members included for brainstorming can be domain experts, subject matter experts.

Interview:

- This is the most common technique used for requirement elicitation.
- Interview techniques should be used for building strong relationships between business analysts and stakeholders.

Survey:

- For Survey, a set of questions is given to stakeholders to quantify their thoughts.

Unit – 02: Software Requirement Engineering

Software Requirements Specification (SRS):

- It is a description of all the requirements stated by the user for software development.
- The SRS fully describes what the software will do and how it will be expected to perform.
- Software documentation is also called a software requirements specification.
- An SRS minimizes the time and effort required by developers to achieve desired goals.
- It is also minimize the development cost of the software.
- A good SRS defines how an application will interact with system hardware, other programs and users.

Advantages of SRS:

- An SRS provides a reference for validation of the final software.
- A high-quality SRS reduces the development time and effort.
- Software requirements specification reduces development efforts.
- SRS provides a baseline for verification and validation.

Need of SRS:

- SRS is a Communication Point
- SRS Includes Final Description of Features
- SRS is a Standard Guide for Product Developers

Unit – 03: Software Planning

- Major responsibilities of Software Project Manager.
- Important project parameters.
- Metrics for Size Estimation:
  - Line of Code (Loc),
  - Function Points(FP).
- Project Estimation Techniques-Using COCOMO Model.
- Risk management.

Questions to be discussed:

1. What are various activities during software project planning?
2. Explain the concept of function points. Why FPs are becoming acceptable in industry?
3. Discuss various types of COCOMO mode. Explain the phase wise distribution of effort.
4. What is meant by a code walk through? Explain with examples.
5. Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e. organic, semidetached and embedded.
6. What are risk management activity?

Major Responsibilities of a software Project Manager:

- A software project manager is the most important person inside a team who takes the overall responsibilities to manage the software projects.
- A project manager has to face many difficult situations to accomplish these works.
- The task of a project manager are classified into two major types:
  1. Project planning
  2. Project monitoring and control

Project planning:

- Project planning involves estimating several characteristics of a project and then planning the project activities based on these estimates made.

Project monitoring and control:

- The focus of project monitoring and control activities is to ensure that the software development proceeds as per plan.

Skill necessary for Managing Software Project:

Some necessary skills are required for Managing Software Project which are given below:

- Knowledge of project estimation techniques
- Good decision-making abilities at the right time
- Previous experience managing a similar types of projects
- Good communication skills to meet the customer satisfaction
- A project manager must encourage all the team members to successfully develop the product
- He must know the various type of risks that may occur and the solution to these problems

Important project parameters:

- Project Parameters are certain characteristics and features that can define a project or its aspects.
- Every project can be exhaustively defined with a help of these six parameters and all of them can be determined before the beginning of any project.
  1. Project scope – actual working content that a project embraces;
  2. Project time – duration of a project, and life-span of its results;
  3. Project integration – a variety and type of participation and collaboration that a project demands from different concerned groups;
  4. Project quality – specifications on project efficiency, capabilities and effectiveness of its results;
  5. Project risks – severity and probability of destructive influence that existing negative factors can put on a project execution;
  6. Project costs – determination of money that a project will most likely consume to be completed;

What are various activities during software project planning?

- After the project has been defined and the project team has been appointed, you are ready to enter the second phase in the project management life cycle: the detailed project planning phase.
- Project planning is the second stage in the project management process.
- During this phase, the project manager creates a project plan, which maps out project requirements.
- The project planning phase includes:
  - Goals and project objectives
  - Success metrics
  - Stakeholders and roles
  - Scope and budget
  - Timeline and schedule
  - Communication plan

Metrics for project size estimation:

- Before discussing the available metrics to estimate the size of project, let us examine what does the term "project size" exactly mean.
- The size of a project is not the number of bytes neither is it the size of the executable code.
- The project size is the measure of the problem complexity in terms of the effort and time required to develop the product.
- Currently, two metrics are popularly being used to measure the project size:
  1. Lines of code(LOC)
  2. Function Points(FP).

Lines of code(LOC):

- LOC is the simplest among all metrics available to measure project size.
- This metric measures the size of a project by counting the number of source instruction in the program.
- While counting the number of lines, blank lines, comment lines and header lines are ignored.
- LOC was developed for line-oriented procedural languages, such as Fortran and Assembly.
- Most used metric in cost estimation and it is very easy in estimating the efforts.

Function point(FP):

- Function point metric was proposed by Albrecht in 1983.
- This metric overcomes many disadvantage of the LOC metric.
- Function point analysis is an effective method for measuring software size.
- By focusing on the features and functions that a user can access and use, this metric can accurately determine the complexity of an application.

Why FPs are becoming acceptable in industry?

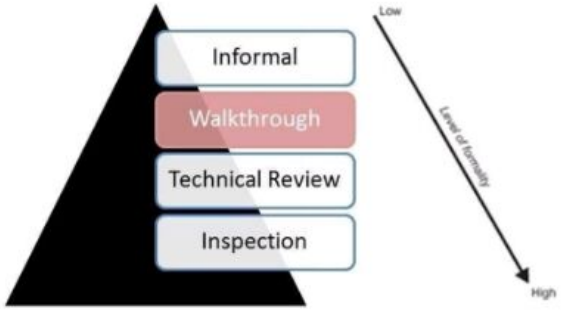
- A FP is a unit of measurement to express the amount of business functionality.
- FPs measure software size.
- They are widely accepted as an industry standard due to:
  - User oriented
  - Language independent
  - Specification based and
  - Used in data processing features.

Differentiate between LOC and Function point:

Line of Code (LOC)	Function Point (FP)
LOC metric is based on analogy.	Function Point metric is specification-based.
LOC metric is dependent on language.	Function Point metric is language independent.
LOC metric is design-oriented.	Function Point metric is user-oriented.
It is changeable to FP (i.e, backfiring)	FP metric is extendible to Line of Code.
LOC is used for calculating the size of the computer program	Function Point is used for data processing systems
LOC is used for calculating and comparing the productivity of programmers.	Function Point can be used to portray the project time

What is Code Walkthrough?

- Code Walkthrough is a form of peer review in which a programmer leads the review process and the other team members ask questions and spot possible errors against development standards and other issues.
- The meeting is usually led by the author of the document under review and attended by other members of the team.
- Review sessions may be formal or informal.
- The main purpose of walkthrough is to enable learning about the content of the document under review to help team members gain an understanding of the content of the document and also to find defects.



Project Estimation Techniques:

- Estimation of various project parameters is an important project planning activity.
- The different parameters of a project that need to be estimated include -
  - Project size
  - Effort required to complete the project
  - Project duration and
  - Project cost
- Accurate estimation of these parameters is important.
- The project estimation techniques are classified into three main categories:
  1. Empirical estimation techniques
  2. Heuristic techniques
  3. Analytical estimation techniques

Empirical estimation techniques:

- Empirical estimation techniques are based on making an educated guess of the project parameters.
- While using this technique, prior experience with development of similar products is helpful.
- Although empirical estimation techniques are based on common sense.
- There are two types of Empirical estimation techniques:
  1. Expert judgement
  2. Delphi techniques

Heuristic techniques:

- Heuristic word is derived from a Greek word that means "to discover".
- The heuristic technique is used for solving problems, learning, or discovery the practical methods which are used for achieving immediate goals.
- In this technique, the relationship among different project parameters is expressed using mathematical equations.
- The popular heuristic technique is given by Constructive Cost Model (COCOMO).

COCOMO Model:

- COCOMO stands for Constructive Cost Estimation Model.
- This model was proposed by Boehm in 1981.
- COCOMO prescribed a three stage process for project estimation.
- In the first stage, an initial estimation is arrived at.
- Over the next two stages, the initial stage is refined to arrive at more accurate estimate.
- COCOMO uses both single and multivariable estimation model at different stage of estimation.
- The three stage of COCOMO estimation techniques are:

1. Organic
2. Semi-detached
3. Embedded

Organic:

- A software project is said to be an organic type if the team size required is adequately small.
- Here, the problem is well understood and has been solved in the past.
- The team members have a nominal experience regarding the problem.

Semi-detached:

- A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environment lie in between that of organic and Embedded.
- Exp: Compilers or different Embedded Systems can be considered Semi-Detached types.

Embedded:

- A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category.
- Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Mode	Project Size	Nature of Project	Innovation	Deadline of the Project
Organic	2–50 KLOC	Small size project, experience developers. Exp: Pay roll, inventory project etc.	Little	Not tight
Semi-detached	50–300 KLOC	Medium size project, Medium size time, previous experience on similar project. Exp: Compiler, Editor, Database etc.	Medium	Medium
Embedded	Over 300 KLOC	Large project, Real time system, complex interface. Exp: ATM, Air traffic control etc.	Significant	Tight

What is person-month(PM)?

- Person-month is a popular unit for effort measurement.
- Because developers are assigned to a project for certain number of months.

Estimation of development effort:

Organic	Effort = $2.4(KLOC)^{1.05}$ PM
Semi-detached	Effort = $3.0(KLOC)^{1.12}$ PM
Embedded	Effort = $3.6(KLOC)^{1.20}$ PM

Analytical Estimation Technique:

- Analytical estimation is a type of technique that is used to measure work.
- In this technique, firstly the task is divided into its basic component.
- Second, if the standard time is available, then they are applied to the component of work.
- Third, if there is no such time available, then the work is estimated based on the experience of work.
- In this technique, results are derived by making certain basic assumptions about the project.
- Hence, the analytical estimation technique has some scientific basis.
- Halstead's software science is based on an analytical estimation model.

Risk Management:

- A risk is a probable problem- it might happen or it might not.
- There are main two characteristics of risk
- Uncertainty- the risk may or may not happen that means there are no 100% risks.
- loss – If the risk occurs in reality , undesirable result or losses will occur.
- Risk management is a sequence of steps that help a software team to understand , analyze and manage uncertainty.
- Risk management consists of
  - Risk Identification
  - Risk analysis
  - Risk Planning
  - Risk

Unit – 04: Software Modeling and Design

- Translating Requirement model into design model:
- Data Modelling.
- Fundamental Design Concepts (Abstraction, Information hiding, Structure, Modularity).
- Software Design Approaches:
  - Function Oriented Design (Data flow Diagrams, Data Dictionary, Decision Trees a Tables).
  - Object-oriented design approach.
- Cohesion and coupling.
- Structured flowcharts.
- Decision Tables.
- Basic UI design

Questions to be discussed:

1. What do you understand by software design?
2. Discuss the difference between object oriented and function oriented design.
3. What is difference between a flow chart and a structure chart?
4. Differentiate between cohesion and coupling.
5. Write short notes on:
   a. Data flow diagram
   b. Data dictionary

What is software model?

- Software models are ways of expressing a software design.
- The models specify the stages and order of a process.
- A software process model is an abstraction of the software development process.

What is a Data Model?

- Good data allows organizations to establish baselines and goals to keep moving forward.
- A Data Model is this abstract model that allows the further building of conceptual models and to set relationships between data items.

What is Data Modeling?

- Data Modeling is the process of creating data models.
- It is the process of simplifying the data model by applying certain formal techniques.
- It conceptually represents data with diagrams, symbols, or text to visualize the interrelation.
- It involves expressing data and information through text and symbols.
- The data model provides the blueprint for building a new database application.

What do you understand by Software Design?

- Software design is the process of defining software solutions to one or more sets of problems.
- It deals with representing the client's requirement, as described in SRS document.
- It is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
- One of the main components of software design is the software requirements analysis (SRA).
- The software design phase is the first step in SDLC.

Fundamental Design Concepts:

- The software design concept simply means the idea or principle behind the design.
- It describes how you plan to solve the problem and how you will design software.
- There are many concepts of software design and some of them are given below:
  - Abstraction
  - Information hiding
  - Structure
  - Modularity

Abstraction - hide Irrelevant data

- Abstraction means to hide the details to reduce complexity and increases efficiency or quality.

Information Hiding:

- Information hiding means to hide the information so that it cannot be accessed by an unwanted party.

Structure- design a structure of something

- Architecture simply means a technique to design a structure of something.

Modularity- subdivide the system

- Modularity means dividing the project into smaller parts to reduce the complexity of the project.

Software Design Approaches:

There are two main approaches to software design:

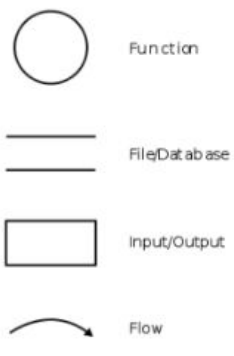
1. Function-Oriented Approach and
2. Object-Oriented Approach.

Difference between Function Oriented Design and Object Oriented Design:

Function-Oriented Design	Object-Oriented Design
The basic abstractions, which are given to the user, are real world functions.	The basic abstractions are not the real world functions but real world entities are represented.
Functions are grouped together by which a higher level function is obtained.	Function are grouped together since the classes are associated with their methods.
It is a top down approach.	It is a bottom up approach.
Begins by considering the use case diagrams.	Begins by identifying objects and classes.
Here, we decompose in function/procedure level.	We decompose in class level.
This approach is mainly used for computation sensitive application.	This approach is mainly used for evolving system which mimics a business.

Data Flow Diagram:

- DFD stands for Data Flow Diagram.
- DFD is a graphical representation using a standard set of symbols and notations.
- The flow of data of a system or a process is represented by DFD.
- It uses defined symbols like rectangles, circles and arrows etc. to show data inputs, outputs, storage points and the routes between each destination.



Data Dictionaries:

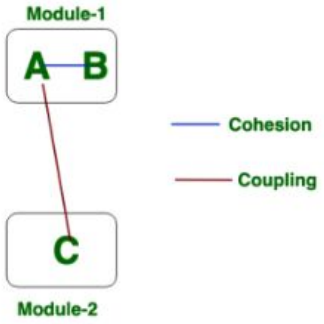
- A data dictionary is a set of files that includes a database's metadata.
- It consists all the data items appearing in DFD.
- The data dictionary hold records about other objects in the database.
- The data dictionary is an essential component of any relational database.
- Typically, only database administrators interact with the data dictionary.
- Data Dictionary is the major component in the structured analysis model of the system.

Data Dictionary Notations tables:

Notations	Meaning
X = a+b	X consists data elements a and b.
X = [a/b]	X consists of either elements a or b.
X = a X	X consists of optimal data elements a.
X = y[a]	X consists of y or more events of data element a
X = [a] z	X consists of z or less events of data element a
X = y [a] z	X consists of some events of data elements between y and z.

Cohesion and coupling:

- Coupling and Cohesion are two key concepts in software engineering.
- It is used to measure the quality of a software system's design.



Coupling:

- Coupling represents the relationships between modules.
- High coupling means modules are closely connected & changes in module may affect the other modules.
- Low coupling means modules are independent & changes in module have little impact on other modules.
- Loosely coupling gives the best software.

Cohesion:

- Cohesion represents the relationship within a module.
- High cohesion means that elements are closely related and focused on a single purpose.
- Low cohesion means that elements are loosely related and serve multiple purposes.
- High cohesion gives the best software.

Difference between cohesion and coupling:

Cohesion	Coupling
Cohesion is the concept of intra-module.	Coupling is the concept of inter-module.
It represents the relationship within a module.	It represents the relationships between modules.
Increasing cohesion is good for software.	Increasing coupling is avoided for software.
It represents functional strength of modules.	It represents the independence among modules.
Highly cohesive gives the best software.	Loosely coupling gives the best software.
In cohesion, the module focuses on a single thing.	In coupling, modules connected to other modules.
Cohesion is created between the same module.	Coupling is created between two different modules.
There are Six types of Cohesion 1. Functional Cohesion. 2. Procedural Cohesion. 3. Temporal Cohesion. 4. Sequential Cohesion. 5. Layer Cohesion. 6. Communication Cohesion.	There are Six types of Coupling 1. Common Coupling. 2. External Coupling. 3. Control Coupling. 4. Stamp Coupling. 5. Data Coupling 6. Content Coupling.

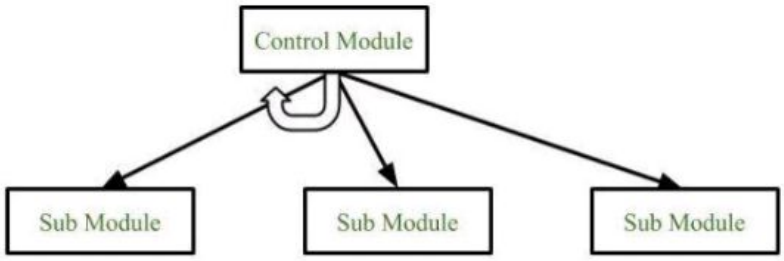
Decision table:

- It is a visual representation for specifying which actions to perform depending on given conditions.
- The information represented in decision tables can also be represented as decision trees.
- It is also represented in programming language using if-then-else and switch-case statements.
- A decision table is a good way to settle with different combination inputs with their corresponding outputs and is also called a cause-effect table.

Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

Structure Chart:

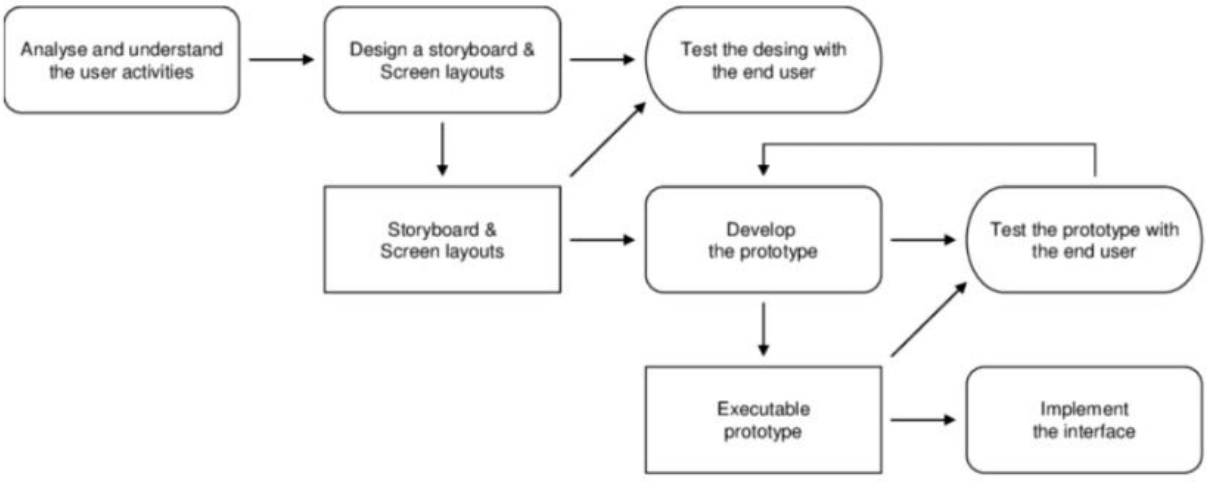
- Structure Chart represent hierarchical structure of modules.
- It breaks down the entire system into lowest functional modules.
- Which describe functions & sub-functions of each module of a system to a greater detail.
- Modules at top level called modules at low level.
- Components are read from top to bottom and left to right.



Basic UI Design:

- UI design stands for User interface design.
- It is the process to build interfaces in software focusing on looks or style.
- Designers aim to create interfaces which users find easy to use and pleasurable.
- UI design refers to graphical user interfaces and other forms—e.g., voice-controlled interfaces.
- It is the front-end application view to which user interacts in order to use the software.
- The software becomes more popular if its user interface is:
  - Attractive
  - Simple to use
  - Responsive in short time
  - Clear to understand
  - Consistent on all interface screens

User Interface Design Process:



Unit – 05: Coding and Testing

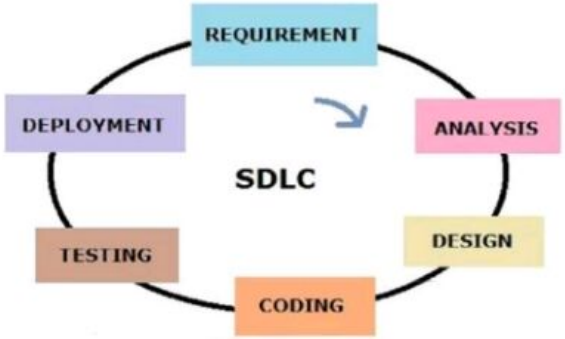
- Coding standard and guidelines.
- Code review.
- Software documentation.
- Verification v/s Validation.
- Concept of Testing.
- Testing: - Unit testing.
- Black-box testing and white-box testing.
- Debugging.
- Integration testing.
- system testing.
- User interface inspection

Questions to be discussed:

1. What is software testing? Discuss the role of software testing during software life cycle.
2. Differentiate between black box testing and white box testing.
3. What is regression testing?
4. What do you understand by structural testing?
5. What are various kinds of functional testing? Describe any one in detail.
6. What is the difference between alpha testing and beta testing?

What is Coding?

- The coding is the essential phase of SDLC.
- It is the process of transforming the design of a system into a computer language format.
- Coding is done by the programmers who are independent people than the designer.
- The main goal of the coding phase is to code from the design document prepared after the design phase.



Coding Standards and Guidelines:

- Different design documents are coded in the Coding phase according to the module specification.
- Good software development organizations want their programmers to maintain some well-defined and standard style of coding called coding standards.
- They make their own coding standards & guidelines depending on what suits their organization best.
- It is very important for the programmers to maintain the coding standards otherwise the code will be rejected during code review.

What is a Code review?

- After completed coding, a code review is an important step in the software development process.
- The reviewer can also act as a second step in identifying bugs, logic problems or other issues.
- Reviewers can be from any team or group as long as they're a domain expert.
- If the lines of code cover more than one domain, two experts should review the code.

What do you mean by software documentation?

- It is the information that describes the product to the people who develop, deploy and use it.
- Software documentation is a written piece of text that is often accompanied by a software program.
- Before the development of any software product requirements are documented which is called SRS.

Advantages of software documentation:

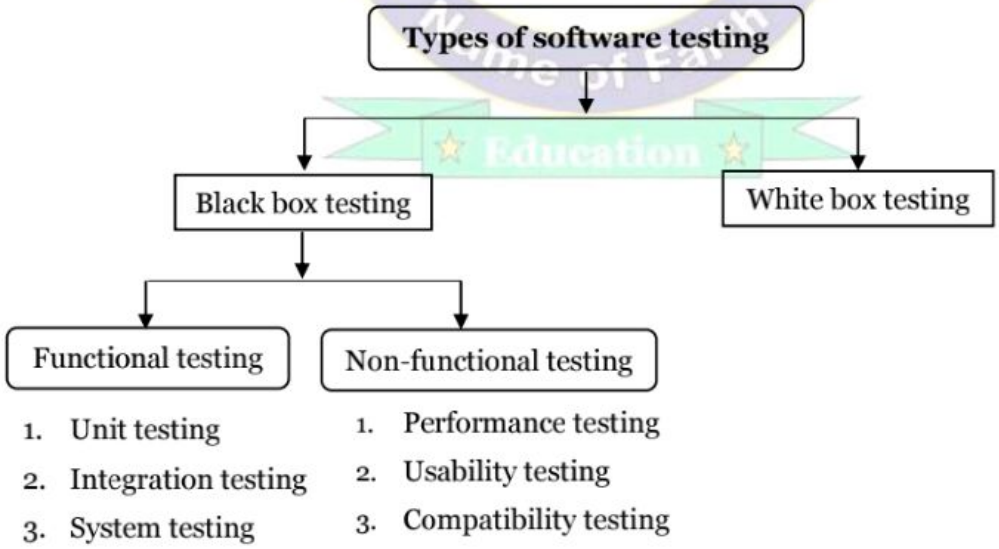
- Helps development teams during development.
- Improves overall quality of software product
- It cuts down duplicative work.
- Makes easier to understand code.

Difference between verification and validation:

Verification	Validation
Verification ensure software correctly implements the specific function.	Validation ensure that the software meets the customer requirements.
Verification is about process, standard and guideline.	Validation is about the product.
Verification is for prevention of errors.	Validation is for detection of errors.
It comes before validation.	It comes after verification.
Done by developer.	Done by tester.

What is testing?

- Testing is the important phase of SDLC.
- Software testing means finding the bug in software.
- Software testing is the process of verifying and validating whether a software is bug-free.
- It checks the software meets the technical requirements as guided by its design and development.
- The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability.



Difference between black-box and white-box testing:

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure is hidden.	It is a way of testing in which the tester has knowledge about the internal structure.
It is mostly done by software testers.	It is mostly done by software developers.
It is the external software testing.	It is the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
No knowledge of programming is required.	Mandatory to have knowledge of programming.
It is also called closed testing.	It is also called as clear box testing.
Example: Search something on google by using keywords	Example: By input to check and verify loops

What is Functional Testing?

- It is a type of Software Testing in which the system is tested against the functional requirements.
- Functional testing ensures that the requirements or specifications are properly satisfied by the application.
- This type of testing is particularly concerned with the result of processing.
- This testing is not concerned with the source code of the application.
- This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test.
- Functional testing can be manual or automated.

Type of Functional Testing:

1. **Unit Testing:** Unit testing is the type of functional testing technique where the individual units or modules of the application are tested. It ensures that each module is working correctly.
2. **Integration Testing:** In Integration testing, combined individual units are tested as a group and expose the faults in the interaction between the integrated units.
3. **Smoke Testing:** Smoke testing is a type of functional testing technique where the basic functionality or feature of the application is tested as it ensures that the most important function works properly.
4. **System Testing:** System testing is a type of software testing that is performed on the complete integrated system to evaluate the compliance of the system with the corresponding requirements.

What is Regression Testing?

- Regression testing is a black box testing techniques.
- It is used to ensure a code change in software does not impact the existing functionality of the product.
- Here, test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.
- It is a software testing conducted after a code update to ensure that the update introduced no new bugs.
- This is because new code may bring in new logic that conflicts with the existing code, leading to defects.

Structural testing:

- It is a type of software testing which uses the internal design of the software for testing.
- In this testing team knows the development phase of the software.
- Structural testing is related to the internal design and implementation of the software.
- It involves the development team members in the testing team.
- It basically tests different aspects of the software according to its types.
- Structural testing is just the opposite of behavioral testing.

Differentiate between alpha testing and beta testing:

Alpha Testing	Beta Testing
Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black-box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user of the product.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Alpha testing may require a long execution cycle.	Beta testing requires only a few weeks of execution.

Unit – 06: Software Project Management

- Project Management Concepts.
- Project scheduling:
  - Basic Principles,
  - Work Breakdown Structure,
  - Activity Network
  - Scheduling techniques (CPM, PERT).
- Configuration and Release Management.
- Version Control and its tools.
- Release Planning

Questions to be discussed:

1. Define the term project. Explain software project management in brief.
2. What is project scheduling? Explain different project scheduling techniques in brief.
3. Explain CPM & PERT with suitable example.
4. Differentiate between CPM & PERT.
5. Explain in details Version control and its tool.
6. Write short notes on:
   a. Work breakdown structure(WBS)
   b. Activity network

What is Project?

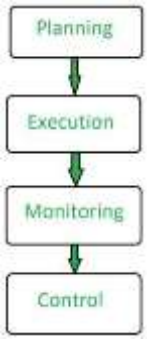
- A project is a group of tasks that need to complete to achieve a specific result.
- A project also defines as a set of inputs and outputs which are required to achieve a goal.
- Projects can vary from simple to difficult and can be operated by one person or a large team.
- Projects usually described and approved by a project manager or team executive.

Software Project Management (SPM):

- SPM is an art of planning and supervising software projects.
- It is a proper way of planning and executing software projects.
- In this, software projects are planned, implemented, monitored, and controlled.
- In SPM, the client and the developers need to know the length, period and cost of the project.

Aspects of Software Project Management:

7. Planning
8. Execution
9. Monitoring
10. Control

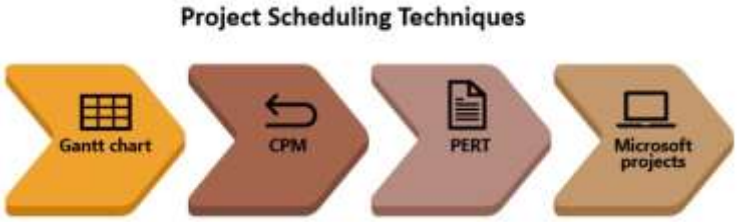


Project scheduling:

- A project scheduling is the sequence of activities.
- It is important for project that are needed to be delivered under a given period of time.
- A project scheduling decide which functions would be taken up and when.
- Effective project scheduling leads to success of project, reduced cost & increase customer satisfaction.
- The most common and important form of project schedule is Gantt chart.

Project scheduling techniques:

1. Gantt chart
2. WBS
3. Activity network
4. CPM & PERT

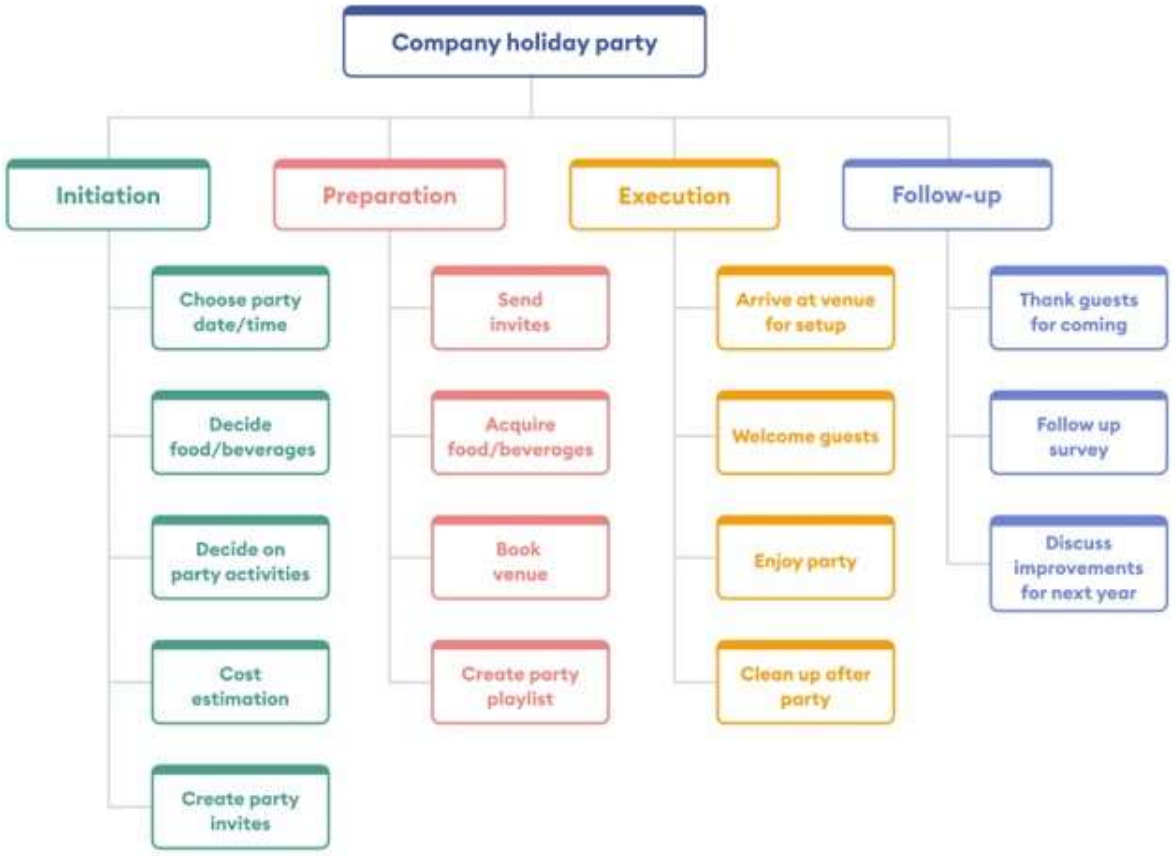


Basic principle of Project Scheduling:

1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

Work Breakdown structure(WBS):

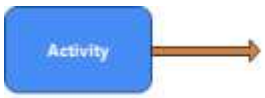
- It is a project management tool that takes a step-by-step approach.
- It breaks large project in several pieces to complete projects efficiently.
- By breaking down the project into smaller components, a WBS can integrate into a single tool.
- A Work Breakdown Structure is a hierarchical outline of the tasks required to complete a project.

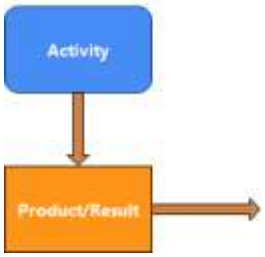


What is an Activity Network Diagram(AND)?

- An Activity Network Diagram (AND) is also called an Arrow Diagram or a PERT.
- It is used for identifying time sequences of events that are pivotal to objectives.
- AND helps to find the specific event sequences driving time requirements for objective achievement.
- There are four symbols extensively used for the activity network diagram.

1. Rectangle with rounded Corners.
2. Arrow.
3. Diamond.
4. Rectangle Box.





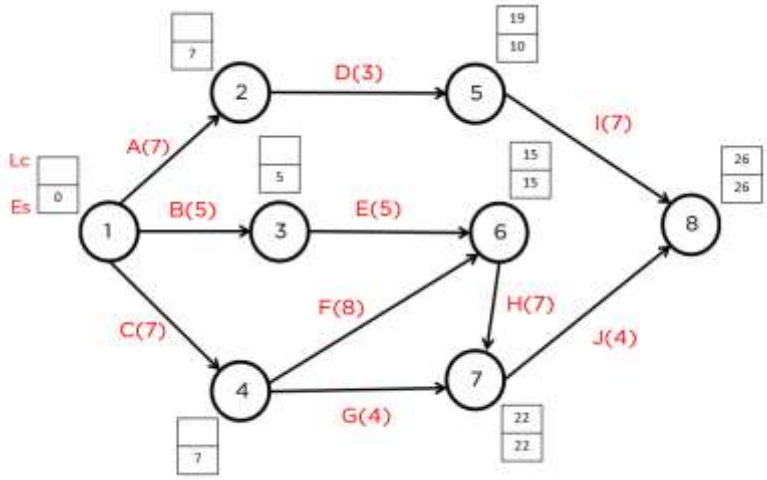
Project management techniques:

There are two types of project management technique:

1. PERT
2. CPM

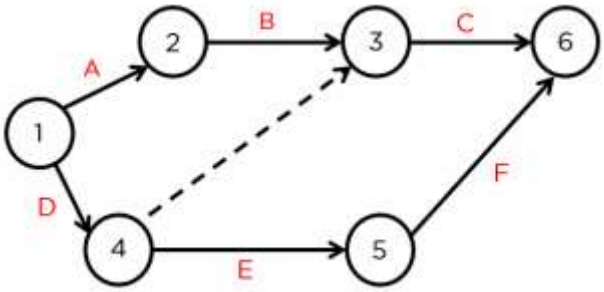
Project Evaluation and Review Technique (PERT):

- PERT stands for Project Evaluation and Review Technique.
- It is a project management tool which is used in project planning.
- It is used to estimate the total time of the project where time is uncertain.
- It was developed by the US Navy in 1950 to manage the Polaris submarine missile program.
- Using PERT a project manager can estimate the minimum amount of time required to complete the project.



Critical Path Method (CPM):

- CPM stands for Critical Path Method.
- It is a project management tool which is used in project planning.
- It is used to estimate the total time & cost of the project where time is certain.
- It was developed by Morgan R. Walker and James of DuPont in 1957.
- Using CPM a project manager can estimate the minimum amount of time & cost required to complete the project.



Difference between PERT and CPM:

PERT	CPM
PERT is a project management technique which is used to manage uncertain activities of project.	CPM is a project management technique which is used to manage only certain activitiesof project.
Here, time is not known.	Here, time is known.
PERT was developed in 1950 by US Navy.	CPM was developed in 1957 by DU PONT.
It is a probability model.	It is a deterministic model.
PERT is used for research & development.	CPM is used for construction work.
PERT analysis does not consider cost.	CPM analysis consider & minimize the cost.

What is Configuration Management?

- CM is a software engineering discipline consisting of standard processes and techniques.
- It is used by organizations to manage the changes introduced to its software products.
- CM helps in identifying individual elements and configurations, tracking changes, and version selection, control, and baselining.
- It precisely answers who, what, when & why for all the configuration items in the configuration management database(CMDB).

What is version control?

- Version Control is a Source Code Management Tools.
- It is the practice of tracking and managing changes to software code.
- VC systems are software tools that help software teams manage changes to source code over time.
- Version control software keeps track of every modification to the code in a special kind of database.
- If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Version Control Software Tools

1. Global Information Tracker(Git)
2. Concurrent Versions System (CVS)
3. Subversion(SVN)
4. Mercurial
5. Monotone
6. Bazaar

What is Release Planning?

- Release planning is the planning process that surrounds the release of a software product.
- A software release plan is an accurate timetable of the various stages of the SDLC.
- This plan involves monitoring software through the development, testing, deployment, and delivery stages.
- It is critical to ensuring smooth migration from internal testing to a customer use environment.
- Release planning also facilitates a holistic view of how various product factors – such as scope, resources, time and quality – should be addressed during a product's life cycle.