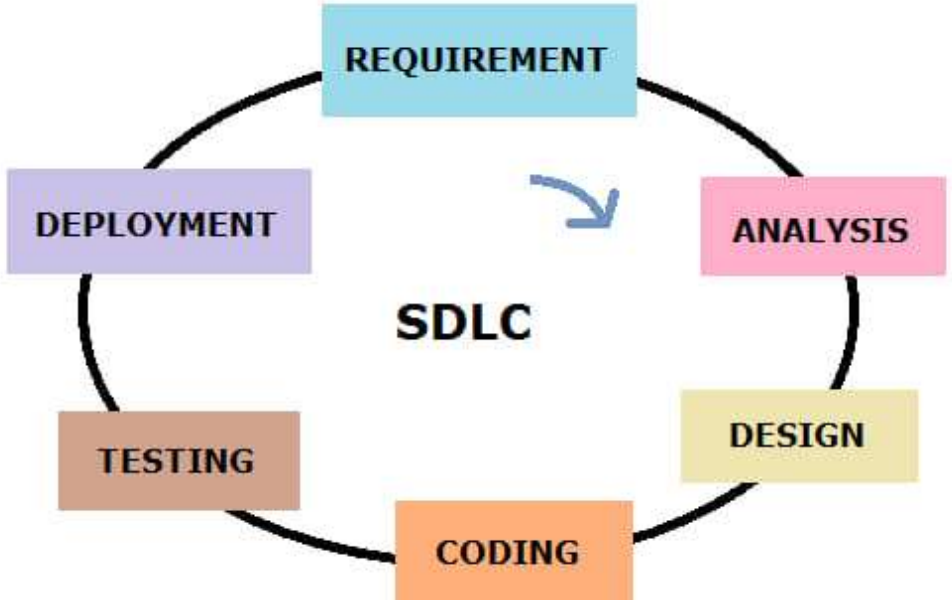
System Architecture and Design Overview

Software Design

- Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
- Software design is the first step in SDLC, which moves the focus from problem domain to solution domain.
- It tries to specify how to fulfill the requirements mentioned in SRS.
- Done by Software Design Engineer OR UI/UX Designer



Objectives of Software Design

1. Correctness: Software design should be correct as per requirement.

2. Completeness: The design should have all components like data structures, modules, and external interfaces, etc.

3. Efficiency: Resources should be used efficiently by the program.

4. Flexibility: Able to modify on changing needs.

5. Consistency: There should not be any inconsistency in the design.

6. Maintainability: The design should be so simple so that it can be easily maintainable by other designers.

Software Design Process

The software design process can be divided into the following **3 levels of phases of design**:

1. **Interface Design:** It focused on interaction between the system and the users devices. Scenario & Behavioral based diagram used. Not focus on internal structure.
2. **Architectural Design:** It focused on major components of a system their responsibilities, properties, interfaces, relationships and interactions between them. Class based & flow-based diagram used.
3. **Detailed Design:** It focused on internal elements of all major system components, thei properties, relationships, processing, algorithms and data structures.

Software Quality Guidelines

1. A design is generated using the recognizable architectural styles and compose a good design characteristic of components and it is implemented in evolutionary manner for testing.

2 . A design of the software must be modular i.e the software must be logically partitioned into sub elements.

3. In design, the representation of data, architecture, interface and components should be distinct.

4. A design must carry appropriate data structure and recognizable data patterns.

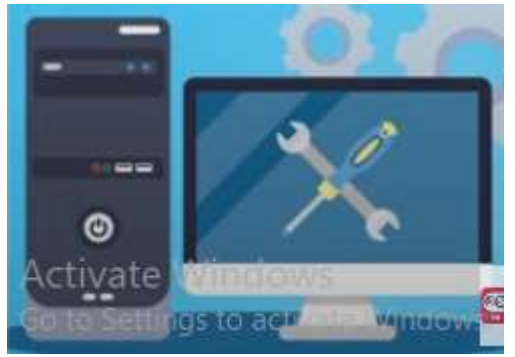
5. Design components must show the independent functional characteristic.

6. A design creates an interface that reduce the complexity of connections between the components with external environment.

7. A design must be derived using the repeatable method.

8. The notations should be use in design which can effectively communicates its meaning.

Software Quality Attributes



- **The attributes of design name as 'FURPS' are as follows:**

1. Functionality: It evaluates the feature set and capabilities of the program.

2. Usability: It is accessed by considering the factors such as human factor, consistency and documentation.

3. Reliability: It is evaluated by measuring parameters like frequency and security of failure, output result accuracy, recovery from failure and program predictability.

4. Performance: It is measured by considering processing speed, response time, resource consumption, throughput and efficiency.

5. Supportability: It combines the ability to extend the program, adaptability, serviceability. Testability, compatibility and configurability are the terms using which a system can be easily installed and found the problem easily. Supportability also consists of more attributes such as modularity, reusability, robustness, security, portability, scalability.

Software Design Concepts

- The software design concept simply means the idea or principle behind the design.
- It describes how you plan to solve the problem of designing software.
- It also shows the logic or thinking behind how you will design software.
- The software design concept for developing the right software provides a supporting and essential structure or model.

Software Design Concepts

1. Abstraction
2. Architecture
3. Design Patterns
4. Modularity
5. Information Hiding
6. Functional Independence
7. Refinement
8. Refactoring
9. Object-Oriented Design Concept

1. Abstraction

- Abstraction is used to hide background details or unnecessary implementation about the data.
- So that users see only required information.

Type 1: Procedural Abstraction:

- There is collections of subprograms.
- One is hidden group another is visible group of functionalities.

- **Example:**
- Private:
  Fuel_machine() Set_top_speed()
  Develop engine()
- Public: Turn on(). Turn off)
  Accelerate(). Break()

Type 2: Data Abstraction:

- Collections of data that describe data objects.
- Show representation data & hide manipulation data.
- **Examples:** Data Structure Programs directly used Push(), Pop(), Top() and Empty() methods.

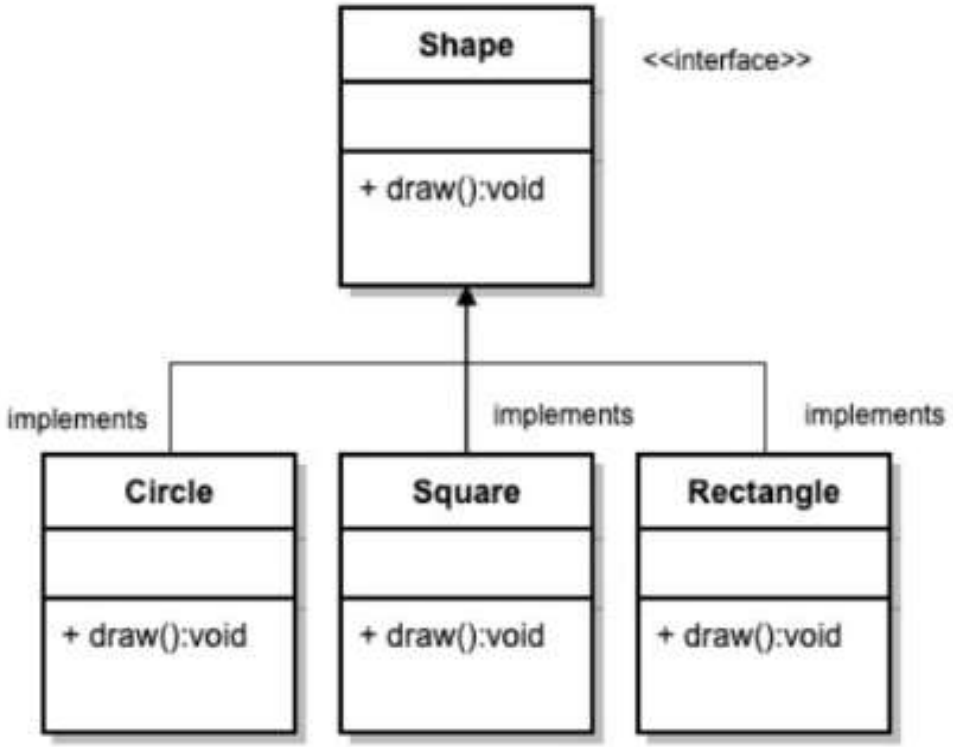
2. Architecture

The architecture is the structure of program modules where they interact with each other in a specialized way.

- **Structural Properties:** Architectural design represent different types of components. modules, objects & relationship between these.
- **Extra-Functional Properties:** How design architecture achieve requirements of Performance, Capacity, Reliability, Security. Adaptability & other System Characteristics.
- **Families of related systems:** The architectural design should draw repeatable patterns. They have ability to reuse repeatable blocks.

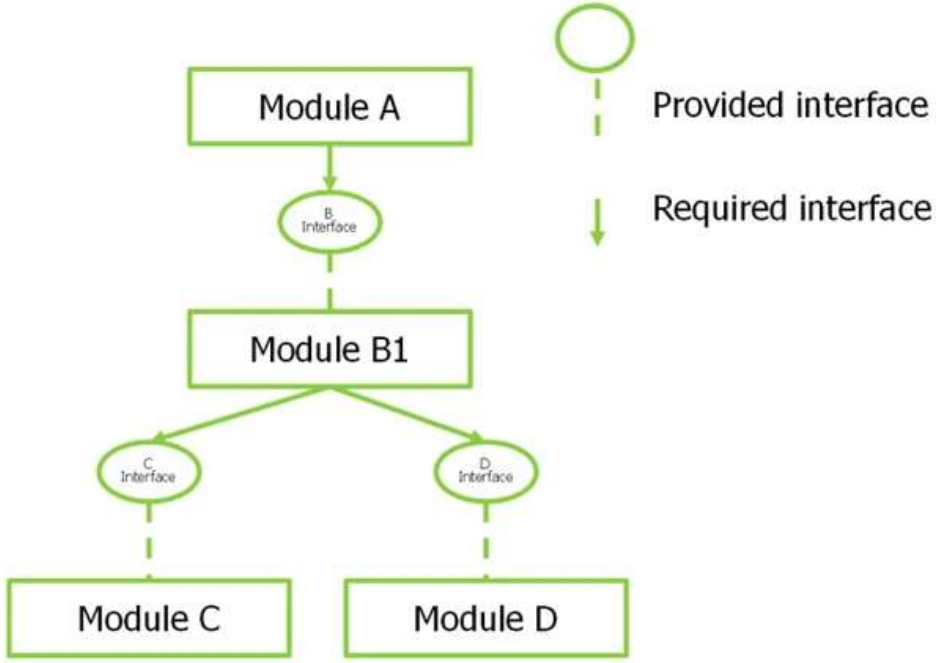
3. Design Patterns

- The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern.
- Example:



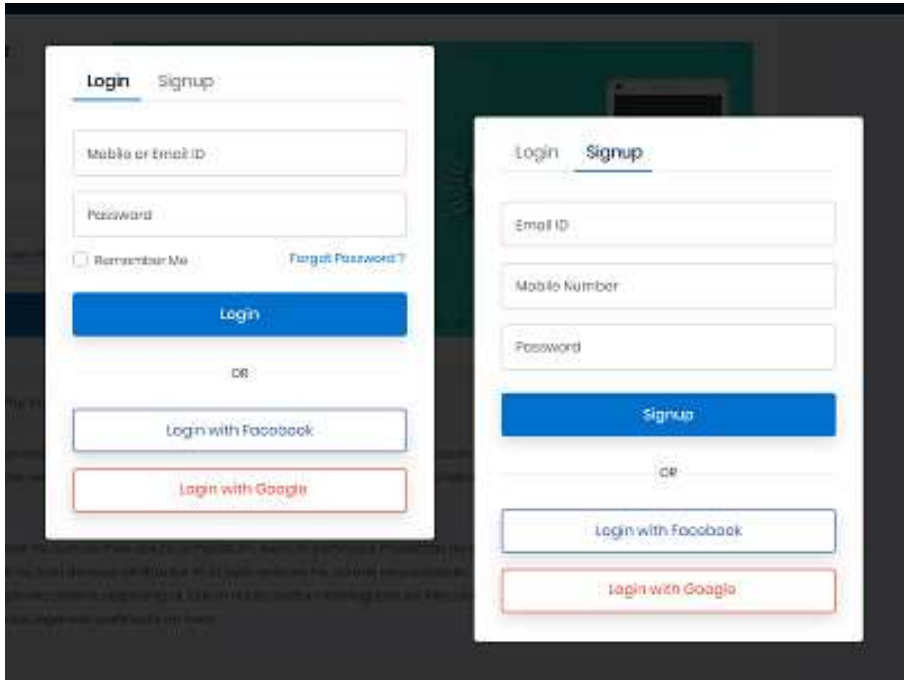
4. Modularity

- Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project.
- After developing the modules, they are integrated together to meet the software requirements.
- Modularizing a design helps to effective development, accommodate changes easily, conduct testing, debugging efficiently and conduct maintenance work easily.



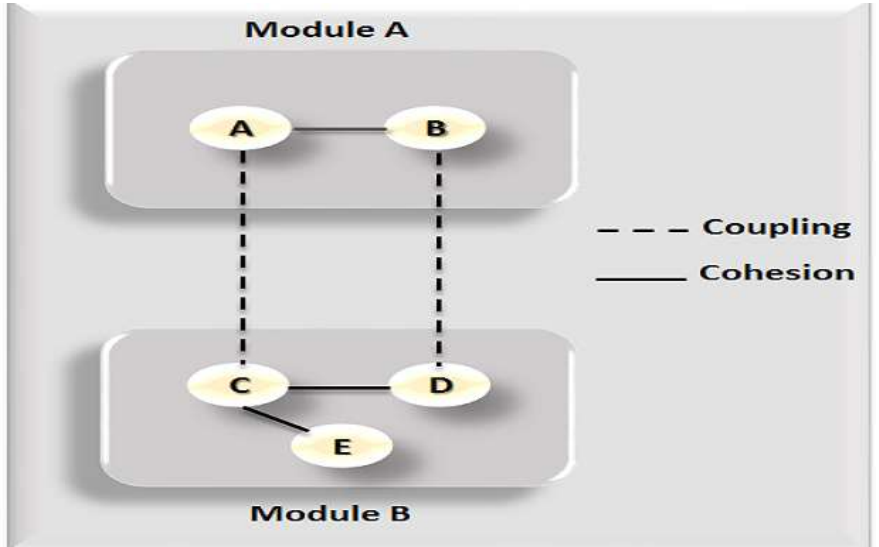
5. Information Hiding

- Modules should be specified and designed in such a way that the data structures and algorithm details of one module are not accessible to other modules.
- They pass only that much information to each other, which is required to accomplish the software functions.
- The way of hiding unnecessary details in modules is referred to as information hiding.



6. Functional Independence

- The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.
- **Criteria 1: Coupling**
  - The degree in which module is "connected" to other module in the system.
  - Low Coupling necessary in good software.
- **Criteria 2: Cohesion**
  - The degree in which module perform functions in inner module in the system.
  - High Cohesion necessary in good software.



7. Refinement

- Refinement is a top-down design approach.
- It is a process of elaboration.
- A program is established for refining levels of procedural details.
- A hierarchy is established by decomposing a statement of function in a stepwise manner till the programming language statement are reached.
- **Example:**

INPUT

Get number 1 (Integer)

Get number 2 (Integer)

PROCESS OUTPUT

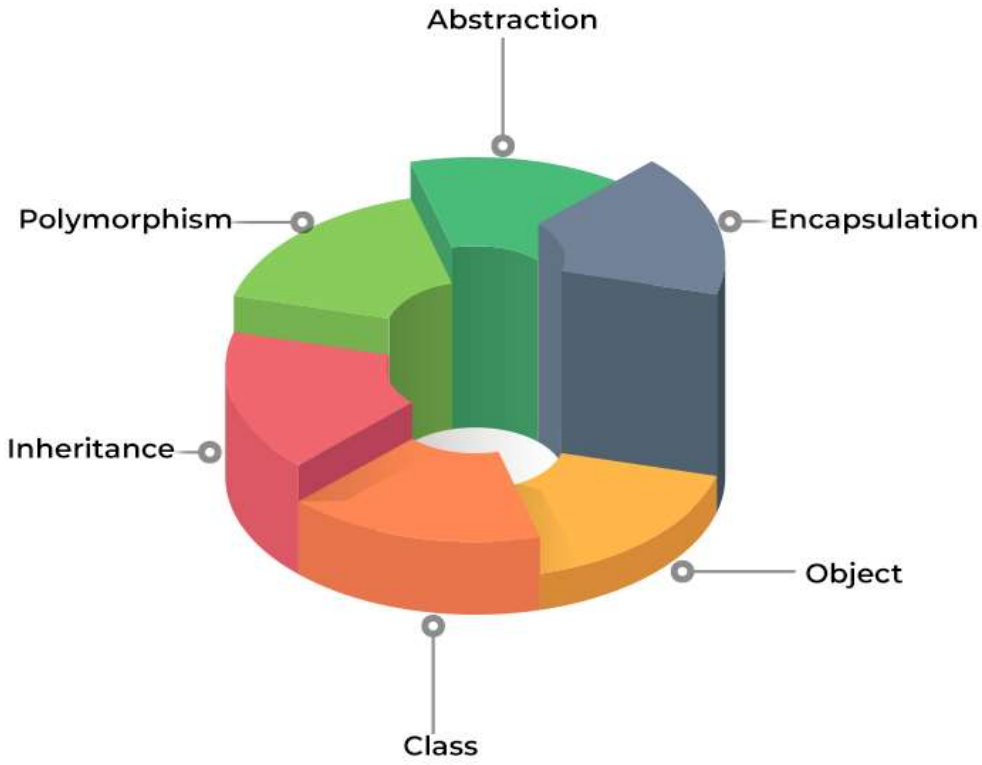
INPUT
Get number 1 (Integer)
Get number 2 (Integer)
While (Invalid Number)
EXIT

8. Refactoring

- Refactoring is the process of changing the internal software system in a way that it does not change the external behavior of the code still improves its internal structure.
- When software is refactored, the existing design is examined for redundancy, unused design elements, unnecessary design algorithms, poorly constructed data, inappropriate data structure or any other design failure that can be corrected for better design.

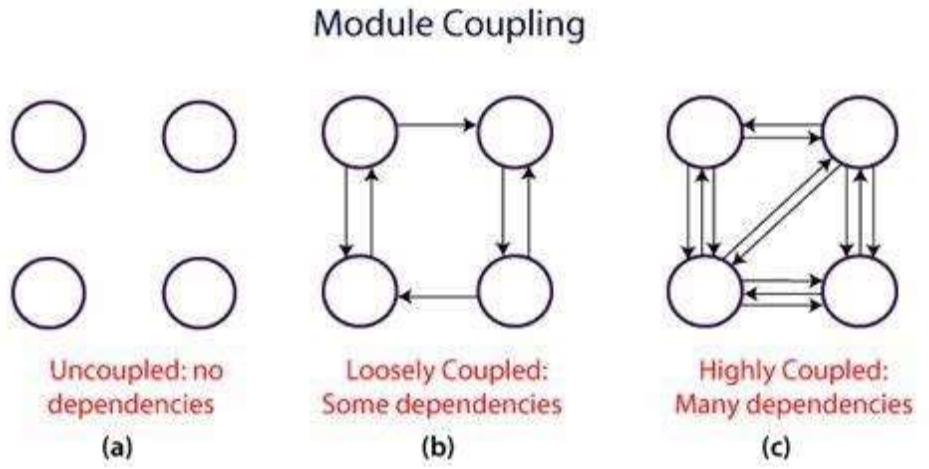
9. Object Oriented Design Concepts

- Object Oriented is a popular design approach for analyzing and designing an application.
- Advantage is that faster, low-cost development and creates a high-quality software.

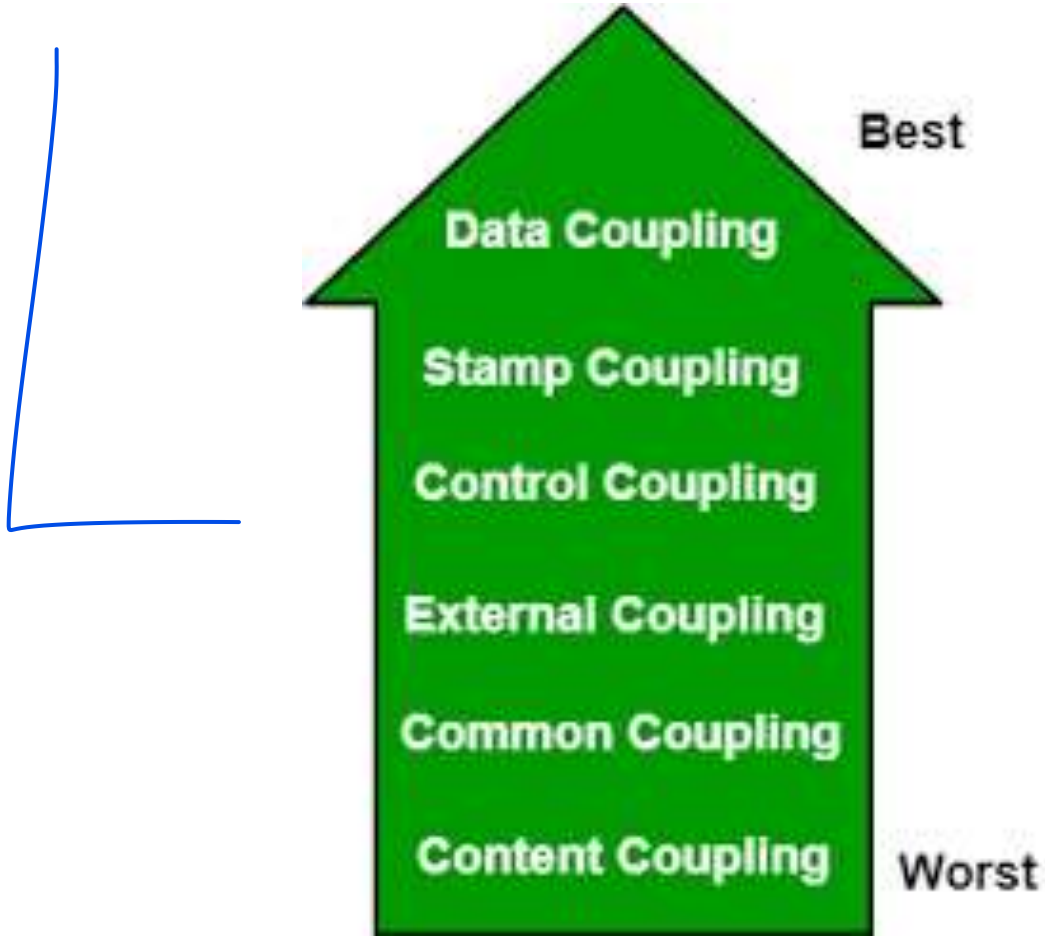


Coupling

- The coupling is the degree of interdependence or number of relations between software modules.
- Two modules that are tightly coupled are strongly dependent on each other.
- However, two modules that are loosely coupled are not much dependent on each other.
- A good design is the one that has Low coupling.
- High coupling generates more errors because they shared large number of data.



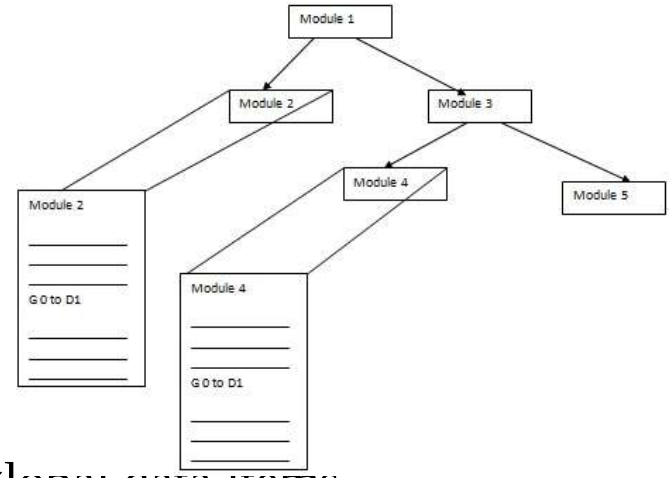
Types of Coupling



Types of Coupling

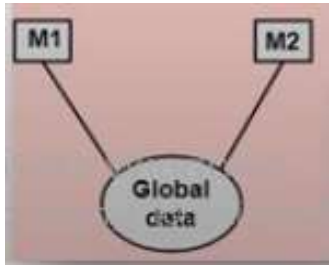
Type 1: Content Coupling

- Here, Two modules are connected as they share the same content like functions, methods.
- When a change is made in one module the other module needs to be updated as well.



Type 2: Common Coupling

• Two modules are common coupled if they share information through some global data items.



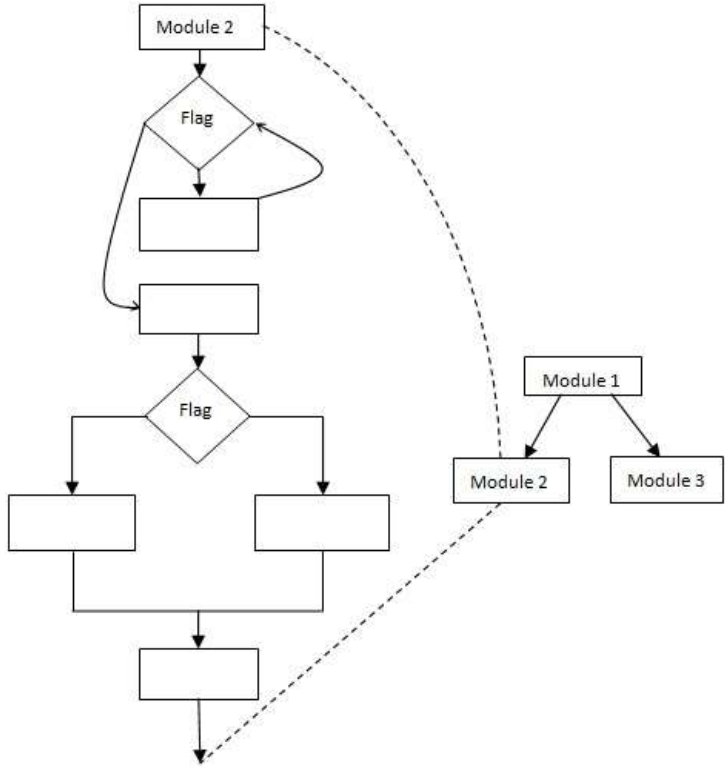
Types of Coupling

Type 3: External Coupling

- When two modules share an externally import data format, communication protocols or device interface.
- This is related to communication to external tools and devices.

Type 4: Control Coupling

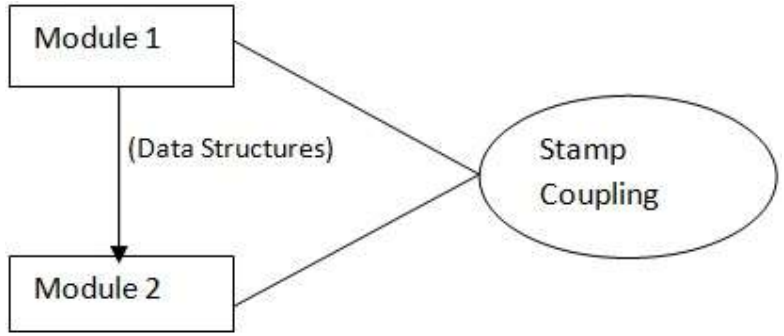
- Control coupling handle functional flow between software modules.
- Example: Module 1- Set Flag = 1 then only Module 2 perform action.



Types of Coupling

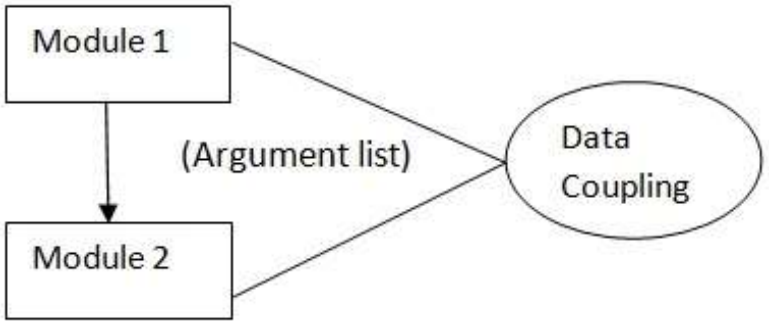
Type 5: Stamp Coupling

- Two modules are stamp coupled if they communicate using composite data items such as Complete Data Structure & objects.
- No junk or unused data shared between the two coupling modules.



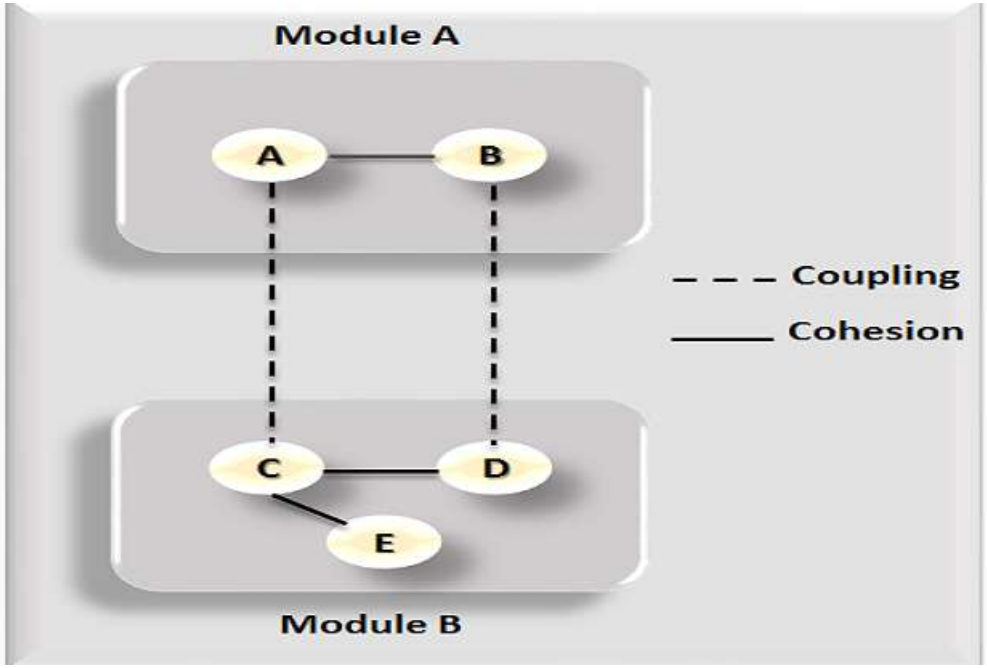
Type 6: Data Coupling

- When data are passed from one modules to another module via argument list or parameters through functional blocks.

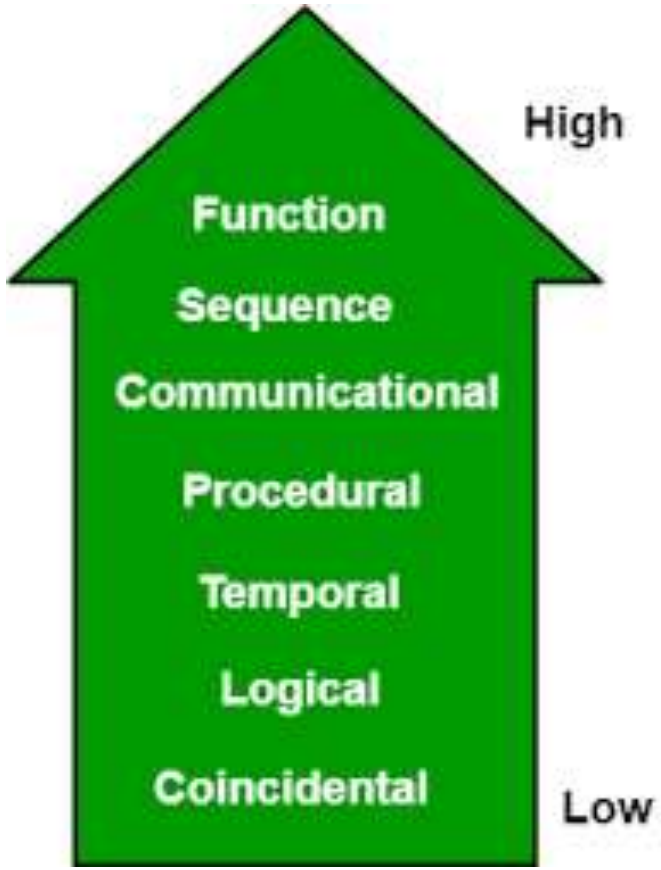


Cohesion

- Cohesion defines to the degree to which the elements of a module belong together or interrelated.
- Thus, cohesion measures the strength of relationships between pieces of functionality within a given module.
- A good software design will have high cohesion.



Types of Cohesion



Types of Cohesion

Type 1: Coincidental Cohesion

- It performs a set of tasks that are associated with each other very loosely.
- Example: Calculator: ADD, SUB, MUL, DIV

Type 2: Logical Cohesion

- If all the elements of the module perform a similar operation.
- Example: Error handling, Sorting, If Type of Record Student, then Display Student Record.

Type 3: Temporal Cohesion

- The activities related in time, Where all methods executed at same time. Temporal cohesion is found in the modules of initialization and termination. Example: Counter = 0, Open student file, Clear(). Initializing the array etc.

Types of Cohesion

Type 4: Procedural Cohesion

- All parts of a procedure execute in particular sequence of steps for achieving goal.
- Example: Calling one function to another function, Loop statements, Reading record etc.

Type 5: Communicational Cohesion

- If all the elements of a module are working on the same input & output data and are accessing that data through the same data structures.
- Example: Update record in the database and send it to the printer.

Type 6: Sequence Cohesion

- Output of one element treats as an input to the other elements inside the same module.
- Example: Enter the numbers -> Perform Addition of that numbers -> Display Addition.

Type 7: Function Cohesion

- If a single module aims to perform all the similar types of functionalities through its different elements. The purpose of functional cohesion is single minded, high, strong and focused.
- Example: Railway Reservation System

Design Models

- The design phase of software development, transforming the customer requirements as described in the SRS documents into design forms.
- Designing a model is an important phase and is a multi-process that represent the data structure, program structure, interface characteristic and procedural details.
- **There are four types of design elements/models**

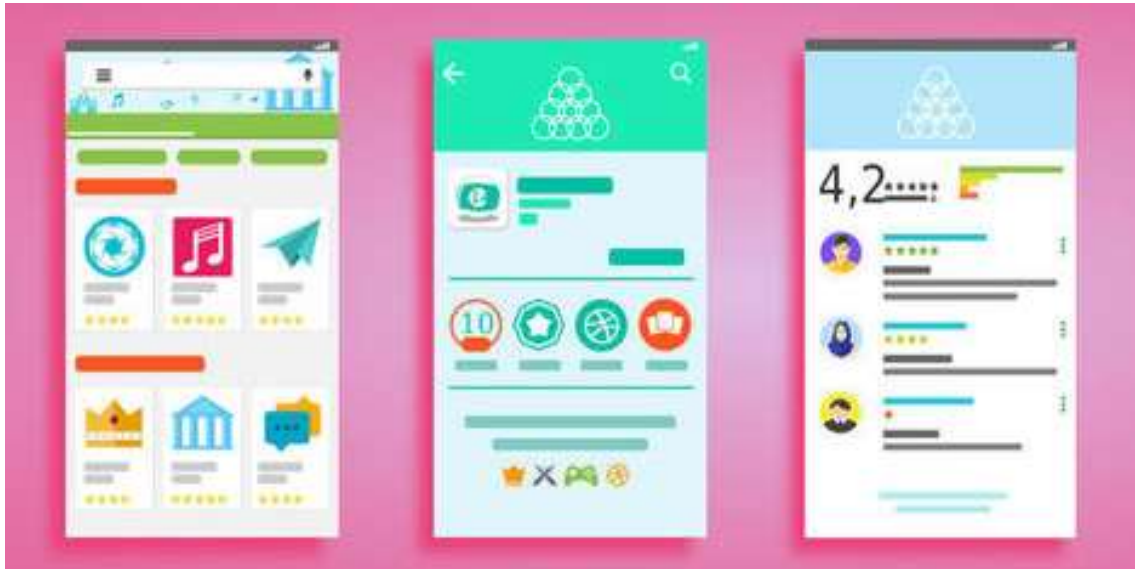
1. Data Design Element / Model
2. User Interface Design Element / Model
3. Architectural Design Element / Model
4. Component Level Design Element / Model

User Interface Design Model

- User interface is the front-end application view to which user interacts with the software.
- It determines how commands are given to the computer or the program and how data is displayed on the screen.

➤**The software becomes more popular if its user interface is:**

1. Attractive
2. Simple to use
3. Responsive in short time
4. Clear to understand
5. Consistent on all interfacing screens



Architectural Design Model

- Architecture Design Model serves as a blueprint for a system.
- The architecture focuses on the early design decisions.

Architectural Design Styles describe:

- Set of hardware & software components that will perform a function required by the system. Eg. Database, Modules, Frameworks etc.
- Set of connectors will help in coordination & communication between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

Architecture Models

- Software architecture involves the high-level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability.
- A software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.

A software architecture can be defined in many ways :

- **UML (Unified Modeling Language)** − UML is one of object-oriented solutions used in software modeling and design.
- **Architecture View Model (4+1 view model)** − Architecture view model represents the functional and non-functional requirements of software application.
- **ADL (Architecture Description Language)** − ADL defines the software architecture formally and semantically.

Importance of Software Architecture

1. Security: The system is secured against malicious users by encryption or any other security measures due to layered software architecture.

2. Performance: It handle request and response of the page in minimum time.

3. Maintainability: Architectural design process uses easily modifiable and replaceable components which is easy to change them over time according to the new requirements.

4. Safety: Avoid critical functionalities in small components & improve communication of the system.

5. Availability: Architectural design process includes corresponding components, functionalities for handling the occurrence of any type of errors.

Decisions of Architectural Design

The architectural design process differs as the system differs depending upon the type of system being developed.

➤There are some common decisions that should be taken care of in any design process.

❑How can the system be distributed across the network?

❑Which approach can be used to structure the system?

❑Which architectural styles are suitable for the proposed system?

❑How can software architecture be documented?

❑How can the system be decomposed into modules?

❑What control strategy must be used to control the operation of the components in the system?

❑ How can architectural design be analyzed?

UML

- UML stands for Unified Modeling Language.
- It is a pictorial language used to make software blueprints. UML was created by Object Management Group (OMG).
- It serves as a standard for software requirement analysis and design documents which are the basis for developing a software.
- UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document a software system.
- UML is generally used to model software system; it is not limited within this boundary. It is also used to model non software systems such as process flows in a manufacturing unit.
- The elements are like components which can be associated in different ways to make a complete UML picture, which is known as a **diagram**.
- So, it is very important to understand the different diagrams to implement the knowledge in real-life systems.
- We have two broad categories of diagrams, and they are further divided into sub-categories i.e., **Structural Diagrams** and **Behavioral Diagrams**.

Structural Diagrams

Structural diagrams represent the static aspects of a system. These static aspects represent those parts of a diagram which forms the main structure and is therefore stable.

These static parts are represented by classes, interfaces, objects, components and nodes. Structural diagrams can be sub-divided as follows :

1. **Class diagram:** Represents the object orientation of a system. Shows how classes are statically related.
2. **Object diagram:** Represents a set of objects and their relationships at runtime and also represent the static view of the system.
3. **Component diagram:** Describes all the components, their interrelationship, interactions and interface of the system.
4. **Deployment diagram:** Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.
5. **Package diagram:** Describes the package structure and organization. Covers classes in the package and packages within another package.
6. **Composite structure:** Describes inner structure of component including all classes, interfaces of the component, etc.

Behavioral Diagrams

Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspects are basically the changing/moving parts of a system.

UML has the following types of behavioral diagrams :

1. **Use case diagram:** Describes the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.
2. **Sequence diagram:** Visualizes the sequence of calls in a system to perform a specific functionality.
3. **Communication diagram:** Same as sequence diagram, except that it focuses on the object's role. Each communication is associated with a sequence order, number plus the past messages.
4. **State chart diagram:** Represents the event driven state change of a system. It basically describes the state change of a class, interface, etc. Used to visualize the reaction of a system by internal/external factors.
5. **Activity diagram:** Describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched.
6. **Interaction overview:** Combines activity and sequence diagrams to provide a control flow overview of system and business process.
7. **Time sequence diagram:** Describes the changes by messages in state, condition and events.

Architecture View Model

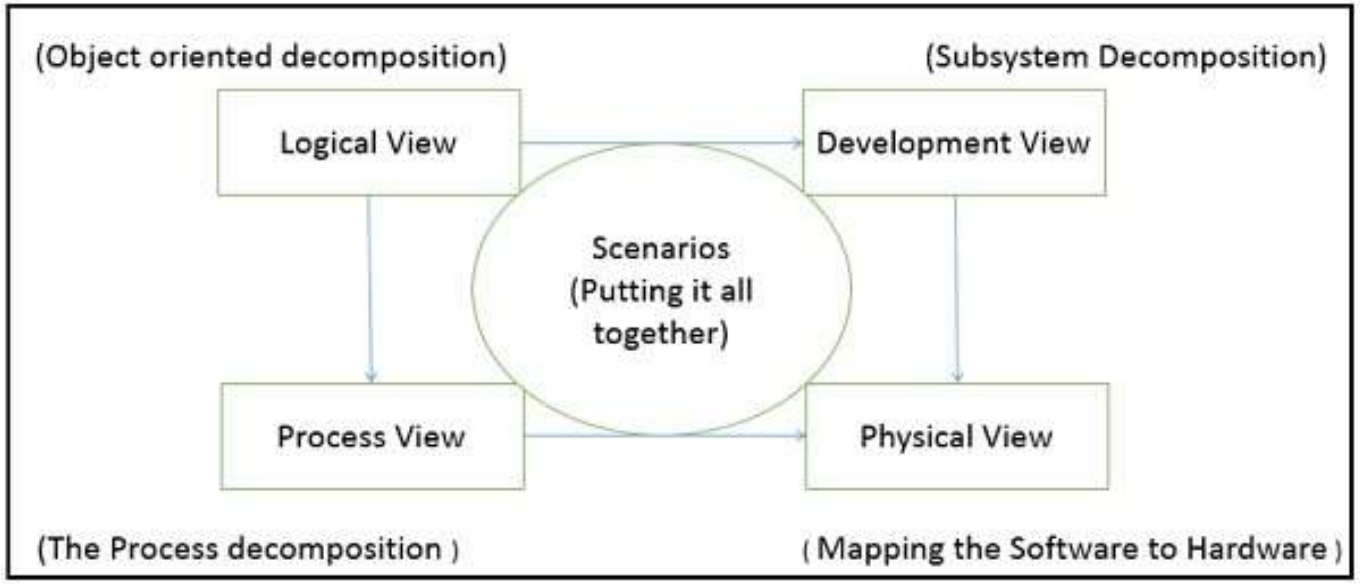
- A model is a complete, basic, and simplified description of software architecture which is composed of multiple views from a particular perspective or viewpoint.
- A view is a representation of an entire system from the perspective of a related set of concerns.
- It is used to describe the system from the viewpoint of different stakeholders such as end-users, developers, project managers, and testers.

4+1 View Model

- The 4+1 View Model was designed by Philippe Kruchten to describe the architecture of a software–intensive system based on the use of multiple and concurrent views.
- It is a **multiple view** model that addresses different features and concerns of the system. It standardizes the software design documents and makes the design easy to understand by all stakeholders.
- It is an architecture verification method for studying and documenting software architecture design and covers all the aspects of software architecture for all stakeholders.
- **It provides four essential views :**
  - **The logical view or conceptual view** − It describes the object model of the design.
  - **The process view** − It describes the activities of the system, captures the concurrency and synchronization aspects of the design.
  - **The physical view** − It describes the mapping of software onto hardware and reflects its distributed aspect.
  - **The development view** − It describes the static organization or structure of the software in its development of environment.

4+1 View Model

- This view model can be extended by adding one more view called **scenario view** or **use case view** for end-users or customers of software systems.
- It is coherent with other four views and are utilized to illustrate the architecture serving as “plus one” view, (4+1) view model.
- The following figure describes the software architecture using five concurrent views (4+1) model.



Why is it called 4+1 instead of 5?

- The **use case view** has a special significance as it details the high-level requirement of a system while other views details ,how those requirements are realized.
- When all other four views are completed, it’s effectively redundant. However, all other views would not be possible without it.

Architecture Description Languages (ADLs)

- An ADL is a language that provides syntax and semantics for defining a software architecture.
- ADLs must support the architecture components, their connections, interfaces, and configurations which are the building block of architecture description.

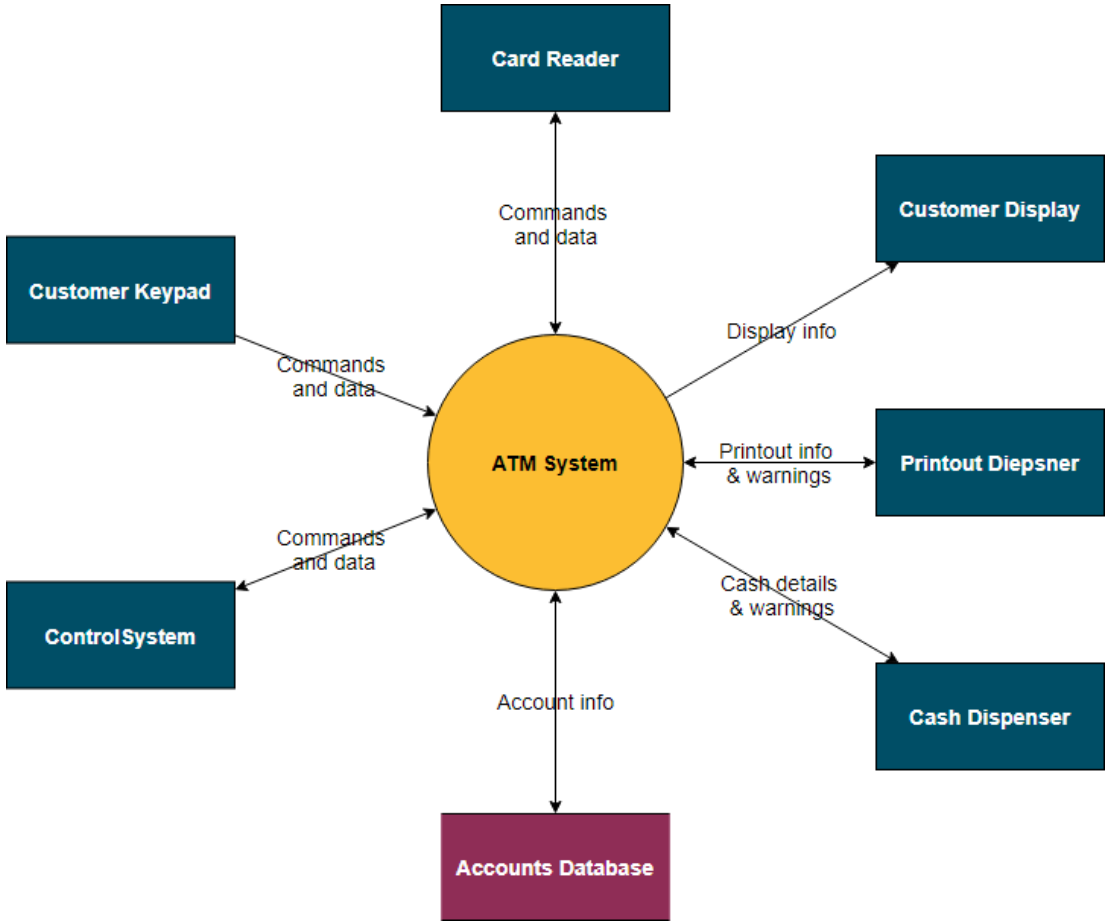
following requirements for a language to be classified as an ADL :

- It should be appropriate for communicating the architecture to all concerned parties.
- It should be suitable for tasks of architecture creation, refinement, and validation.
- It should provide a basis for further implementation, so it must be able to add information to the ADL specification to enable the final system specification to be derived from the ADL.
- It should have the ability to represent most of the common architectural styles.
- It should support analytical capabilities or provide quick generating prototype implementations.

Taxonomy of Architectural styles

Architectural styles is establishing a complete structure & components of all over the system.

1. Data Centered Architecture
2. Data Flow Architecture
3. Object Oriented Architecture
4. Layered Architecture



Data Centered Architecture

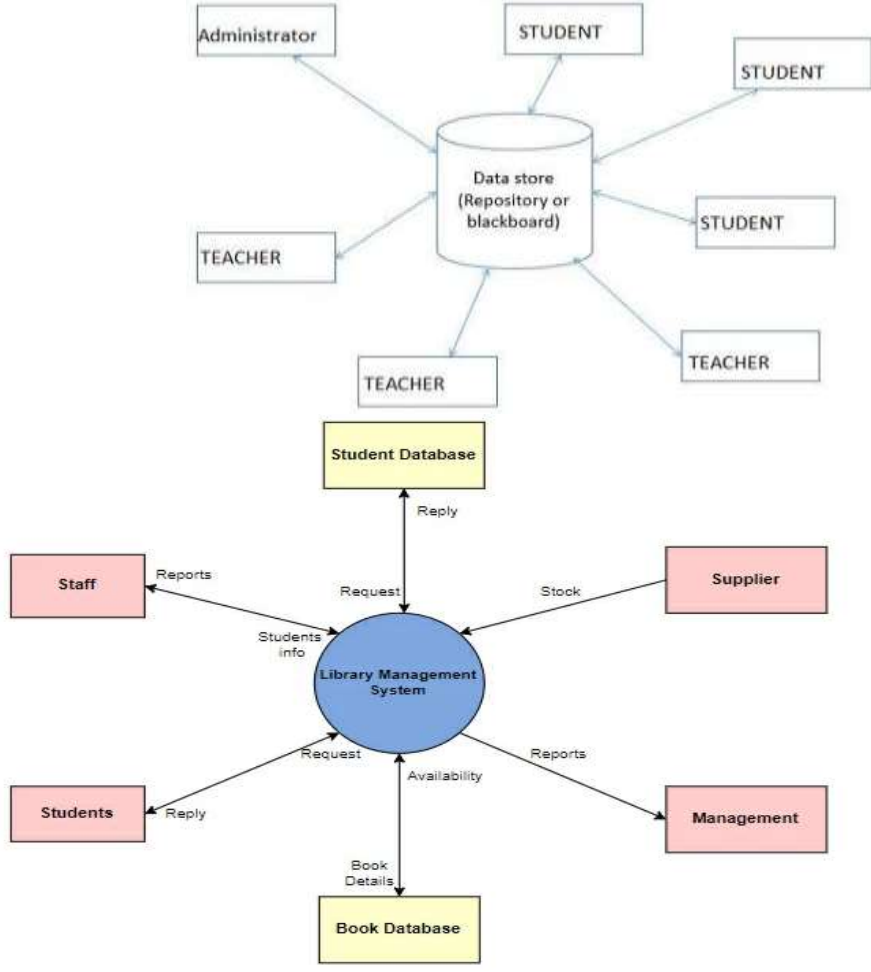
- Data store at the center of this architecture and is accessed frequently by everyone.
- Update, add, delete or modify the data present within the store.
- It is widely used in DBMS, Library Information System etc.

Advantages:

- 1. Repository of data is independent of clients
- 2. It may be simple to add additional clients.
- 3. Modification can be very easy.

Disadvantages:

- 1. Data replication or duplication is possible.
- 2. Changes in data structure highly affect the clients.



Data Flow Architecture

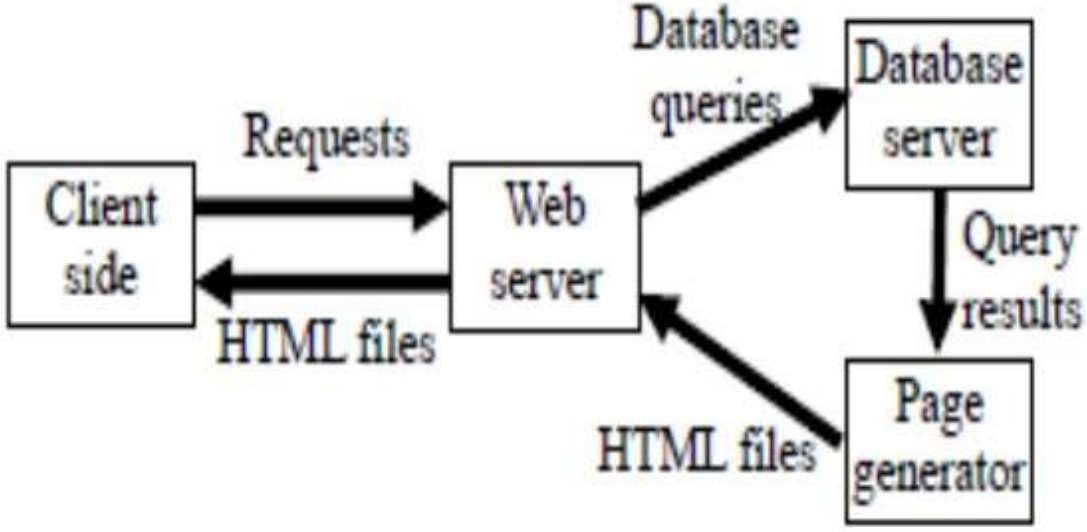
- This architecture is used when input data to be transformed into output data through a series of computational manipulative components.
- **Pipe** is a connector which passes the data directionally from one filter to the next.
- **Filter** is a component reads the data from its input pipes and performs its function.
- This data and places the result on all output pipes.

Advantages:

- 1. With this design, concurrent execution is supported.

Disadvantages:

- 1. Does not allow greater user engagement.



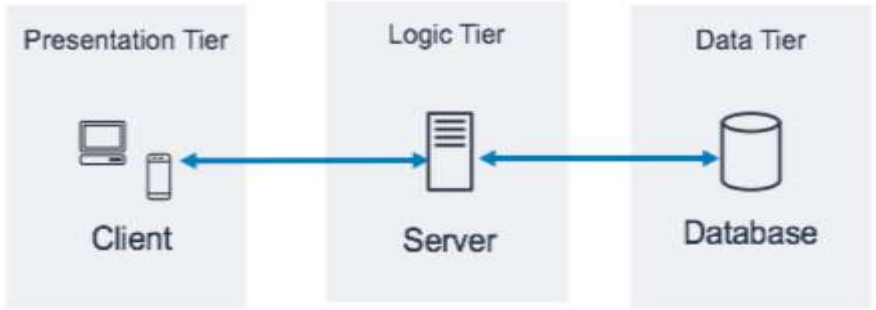
Object Oriented Architecture

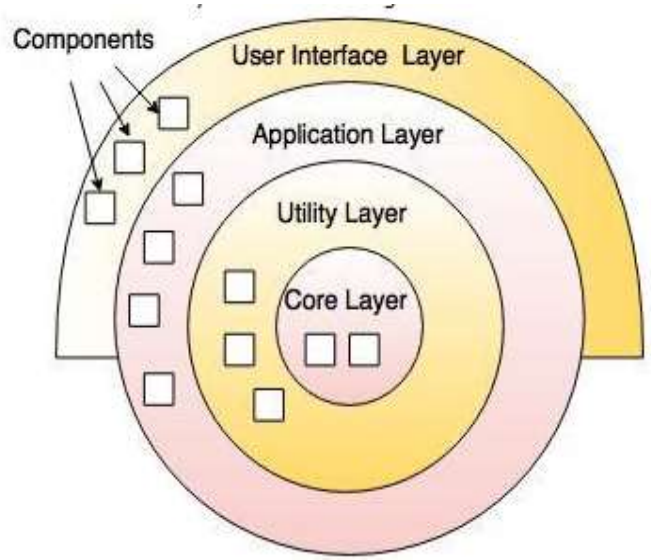
- Objects are the foundational building blocks for all kinds of software applications.

1. **Object:** Object is an instance of a class. Example Student S, Person;
2. **Class:** It defines all attributes, methods, which represents the functionality of the object.
3. **Encapsulation:** It is the process of binding similar types of elements of an abstraction.
4. **Abstraction:** It is the removal of irrelevant essentials from users.
5. **Inheritance:** It deriving a new class from existing one. It increases code reusability.
6. **Polymorphism:** It has multiple forms. Ex: draw graphic objects circle, rectangle, triangle
7. **Message Passing:** Sending and receiving data among objects through function parameters.

Layered Architecture

- Data moves from one level to another level for processing is called layered architecture.
- Number of different layers are defined every layer performing well-defined set of operations.
- **Outer layer** components manage the user interface operations.
- **Inner layer** components will perform the operating system interfacing.
- **Intermediate layers** provide utility services and application software functions.
- Example: E-commerce web applications development like Amazon.





Architectural Views

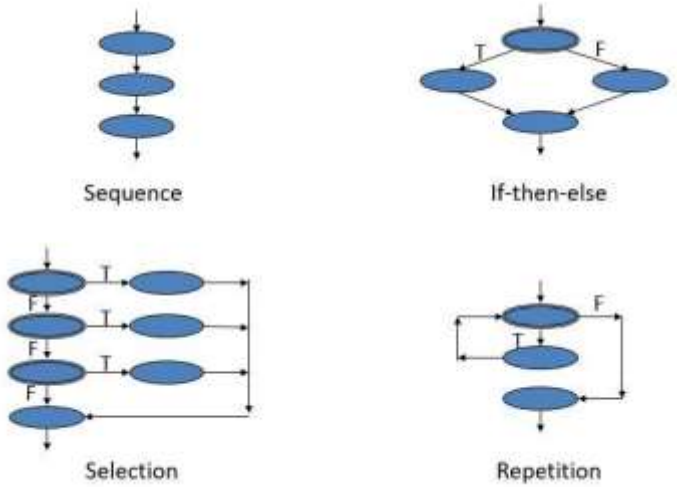
- It is generally used to represent entire architecture that is useful and meaningful to one or more stakeholders in system.

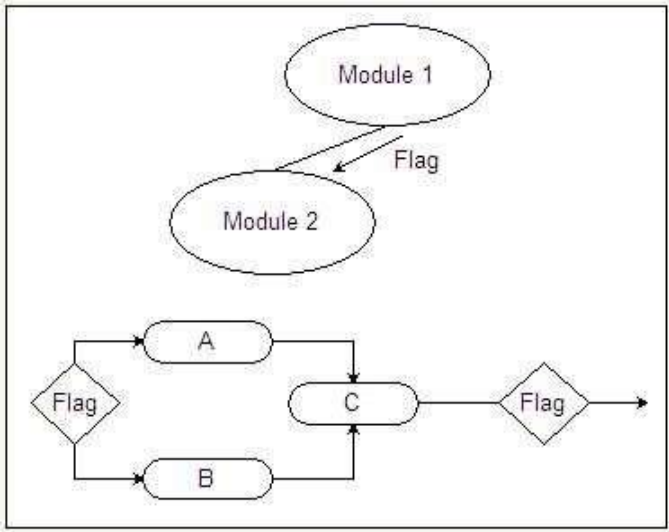
1. **Use case View:** Users view to handling product as per requirement. Use case diagram used.
2. **Design View:** Organize design information, significant features, entities and attributes.
3. **Process View:** Describe communication, behavior & synchronization aspects of the design.
4. **Implementation View:** It address source code integrators and developers of project.
5. **Deployment View:** It describes & explains environment within system runner & executed.

Component Level Design Model

➤What is Component?

• A component is a modular, portable, replaceable and reusable set of well-defined functionalities.





➤What is Component Level Design?

The component-based architecture focuses on breaking the software design into small individual modules.

The component design describes communication, interfaces, algorithms & functionalities of each component regarding the whole software design.

Component Level Design notations are Activity Diagram, Data Flow Diagram, Conditional Notations & Tabular forms.

Benefits of Component Design

1. The major benefit of using the Component design is that it makes every module reusable.
2. It also reduces the cost as every module is reusable.
3. It is more reliable as the client gets its interaction with the system with each module, so it increases the reliability of the entire system.
4. The component design makes the system easy to maintain as we don't need to make changes in the entire system.
5. The component design describes each component's functionality more clearly.

Steps of Component Level Design

1. Identify all design classes that corresponds to problem domain.

2. Identify all design classes that corresponds to infrastructure domain. **Ex**. GUI Class, Database management class, OS Communication class etc.

3. Elaborate all design classes that are not acquired as reusable component. **Ex**. Coupling & Cohesion methods like message details between classes, data type, data structure & data flow between classes.

4. Describe persistent data source & identify classes require to manage them. **Ex**. Classes which data from databases.

5. Develop behavioral representation for a component & class. **Ex**. Classes, Operations & their Interactions through Activity & State diagram.

6. Elaborate deployment diagram. **Ex**. Hardware, OS, Server of components.

7. Factors every component level design representations. **Ex**. Diff. design solutions that provide good characteristics.

Component Design Example

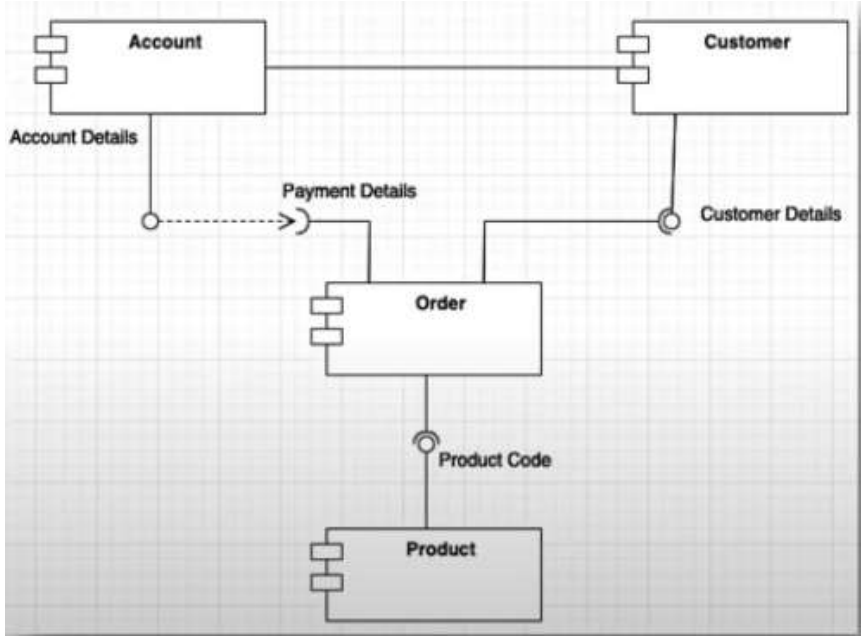
- **It has Account, Customer, Order & Product this Four Components.**
- **Every object is an individual component but is related to each other at the same time.**

1. Customer will go online site and search product.

2. By selecting product, the customer will place an order.

3. System will store customer & his account information in Order Database.

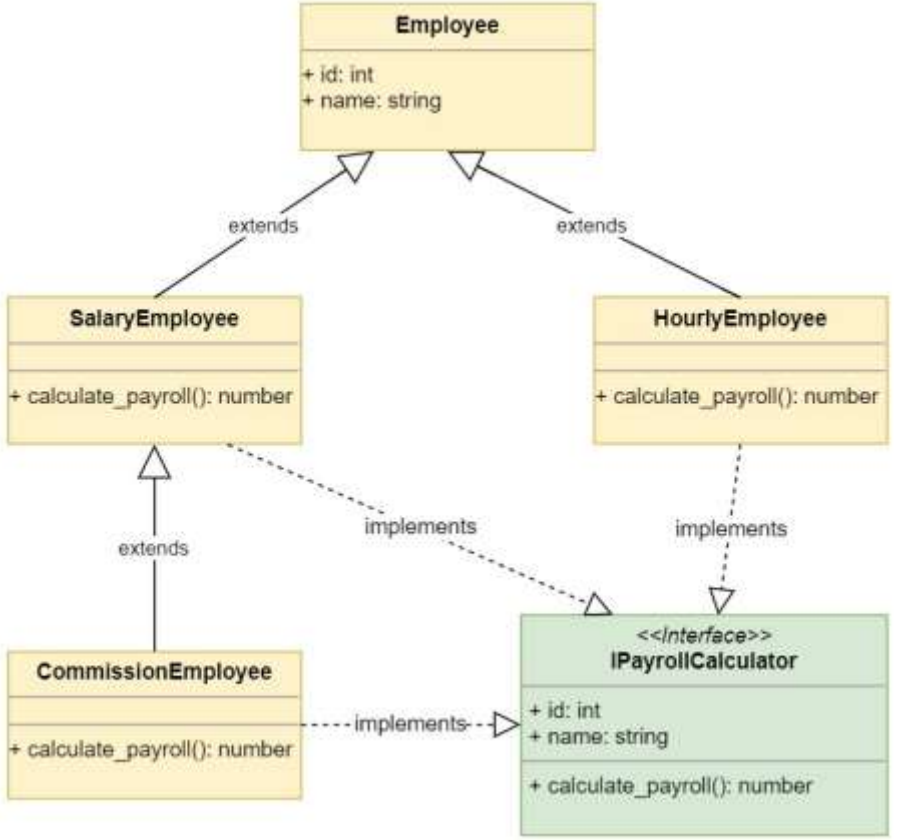
4. The order would be generated against the customer and stored in the Database.



Component Design View

1. Object Oriented View:

- An object-oriented view is a set of collaborating classes in module mentioned in architectural design
- It view all the objects, methods, properties and functionalities mentioned in classes.
- It also view how classes in each module are connected with each other.
- It detailed out data structure & algorithms required for processing.



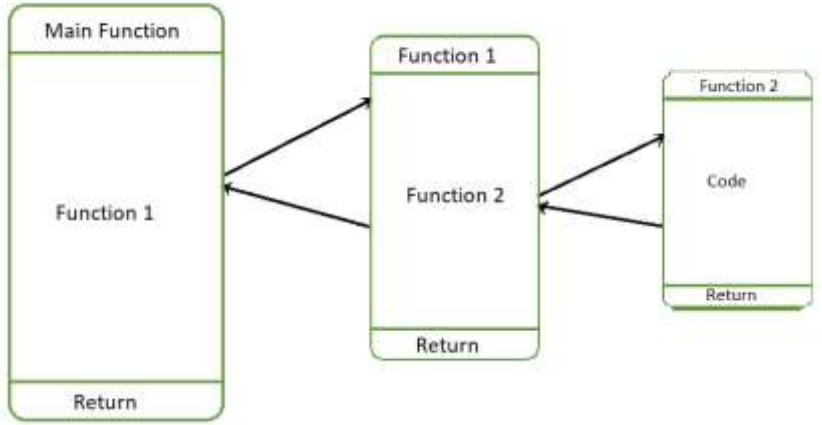
Component Design View

2. Conventional View:

- It is viewed functional element of a program that integrates the all logical & internal operations performed in class
- It also view calling and coordinating the request like calling functions, parameter passing, data passing between the of other modules.
- They analyze complete view of data flow model & state diagram.

3. Process-related View:

- The process-related view majorly focuses on the reusability feature of the component design.
- Databases and libraries are used to store pre-existed modules.
- This pre-existed data used for creating new modules.



Project Planning

Before starting a software project, it is essential to decide which tasks to be performed and how to manage all tasks involved in the software development.

✓Project planning is an organized process from requirement gathering to testing and support.
✓Focuses on all activities required for successful completion of the project.

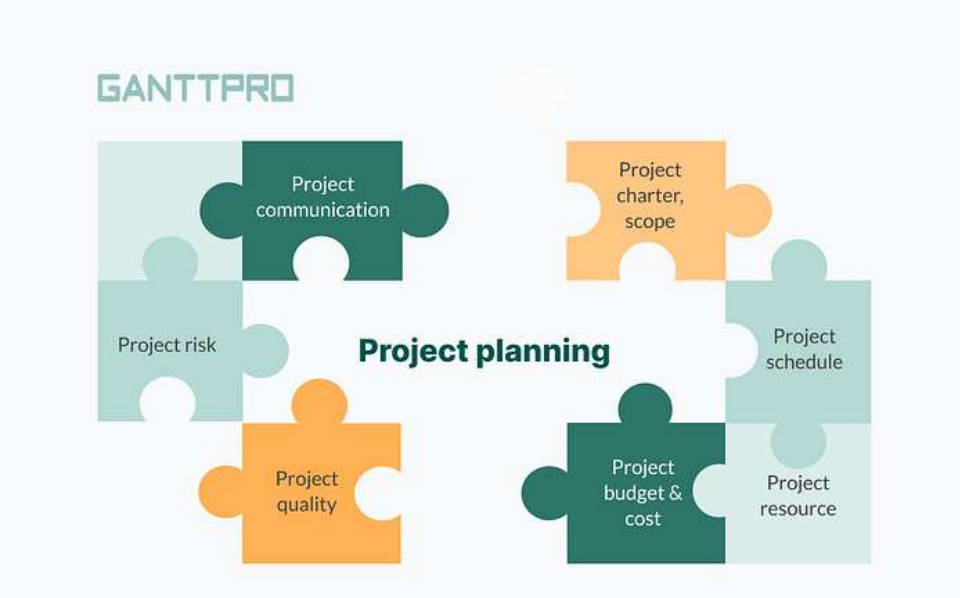
✓Prevents obstacles that arise in the project such as changes in projects, organization's objectives, non-availability of resources and so on.

✓Helps in better utilization of resources and optimal usage of the allotted time for a project.
✓Defines roles and responsibilities of the project management team members.

Software Project Manager

- **Managing People :**
  - Act as Project Leader.
  - Contact with all stakeholders.
  - Managing human resources.

- **Managing Project :**
  - Defining and setting up project scope.
  - Managing project management activities.
  - Monitoring progress and performance.
  - Risk analysis at every phase.
  - Take necessary step to avoid or come out of problems.



Project Planning Process

Project Planning Process

1. Identify Stakeholders Need:

- A project must meet the expectations of stakeholders & meet all of their needs to be effective.
- To consider their needs a project planner might identify the characteristics and qualities of the project at hand.

2. Identify Project Objectives:

- Project objectives must be specific, measurable, achievable, realistic and time-bound in order to be used to validate project success.
- All team members must contribute ideas and compromise on collective goals for the project.

Project Planning Process

3. Identify Deliverables & Due Date:

- A due date is a scheduled fixed date and time that an objective is due within a project.
- Deliverables are defined as the products, services or results that project quality, size, length, or any other standard that proves to be important in the success of project.

4. Identify Project Schedules:

- This stage is created through different tasks that label a project's start and end date.
- Schedules are used to make sure projects stay on task and are completed in a timely manner.

Project Planning Process

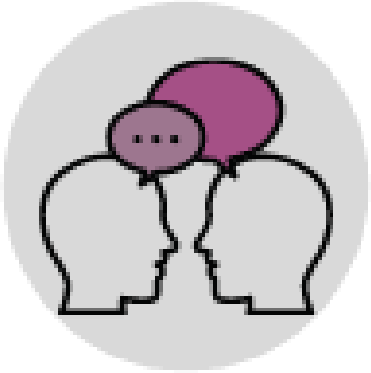
5. Provide Roles & Responsibilities:

- This stage allows for effective communication between everyone involved in the projec
- This segment includes identifying who is involved and what their individual tasks are.
- Everyone understand expected objectives, goals & output of project.

6. Identify Project Budget:

- Project budget are calculating the anticipated cost, changing budget cost and monitoring the budget.
- Therefore, it is crucial to prepare for unexpected alterations of project costs.

Project Planning Process



7. Identify Communication Plan:

- A communication plan is a tool that able to effectively communicate about a project to your client, team and other stakeholders.
- Communication plan has clear guidelines about how information will be shared & who is responsible within the project.

8. Provide Tracking & Management:

- It is an effective way to deliver projects on time and organize tasks.
- Features of this element include planning/scheduling, collaboration, documentation and evaluation.
- Management tools are track productivity and growth of project.

Recourses Used in Project Development

Resource Pyramid

Project resources simply mean resources that are required for successful development and completion of project on time and on budget.

Type 1: Human Recourses:

- In software industry, manager, software developer, software testing and so on.
- These positions are according to their skills and specialty.

Type 2: Reusable Components:

- Managing budget for project is one of most important tasks that all project managers. Resource Pyramid
- The reusable resources are very helpful as they help in reducing overall cost of development.

Type 3: Hardware and Software tools:

- It should be planned before starting development of project otherwise it way causes problems.
- These are actually material resources that are part of project.

Project Scope Management

- Scope refers to the detailed set of deliverables or features of a project.
- This includes all the objectives, activities, process, output, deadline need to be done in order to make a deliverable software product.
- Scope management is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done.

Real Life Example of Project Scope Management

- 'I need a new house', & no one has any idea about functionality of the house at this point.
- Should it be Apartment or an Independent house?
- How many rooms, swimming pool, car park, security, air conditioning the scope gets elaborated as the project progresses.
- Scope which has major impact on the architecture, budget of the project need to be fixed before we commence the detailed planning.
- The purpose of 'Project scope management' is to ensure that the project includes all the work required and exclude all the unwanted work during the beginning till the end of the project.

Steps of Project Scope Management

Project Scope Management

Step 1: Plan Scope Management

- Documentation & guidelines of Project Scope, Product Scope, Project Life Cycle, Requirement management Plan & Scope management plan etc.
- This process provides guidance and direction for managing scope across the project.
- How to define, Validate and control the project scope.

Step 2: Collect Requirements

- Collect requirements from all stakeholders who you have identified as impact on the project.
- Collect Business, Stakeholder, Product, Transition, Quality requirements.
- Collect requirements from Interviews. Brainstorming, Questionnaires', Surveys, Group Discussion, Voting & Prototype etc.

Project Scope Management

Step 3: Defining Scope

- Define the project scope by identifying Project objectives, Goals, Sub-phases. Tasks. Budget Resources & Schedule.
- Identify Human Recourses like Inter-personnel and team skills.
- This ensures that the client, stakeholders, senior management, project manager & team members are all aware of what is expected.

Step 4: Create WBS

- The Work Breakdown Structure involves subdividing project deliverables into smaller units.
- Break down project into phases, including the priority tasks required in order to complete each phase.

Project Scope Management

Step 5: Verify Scope

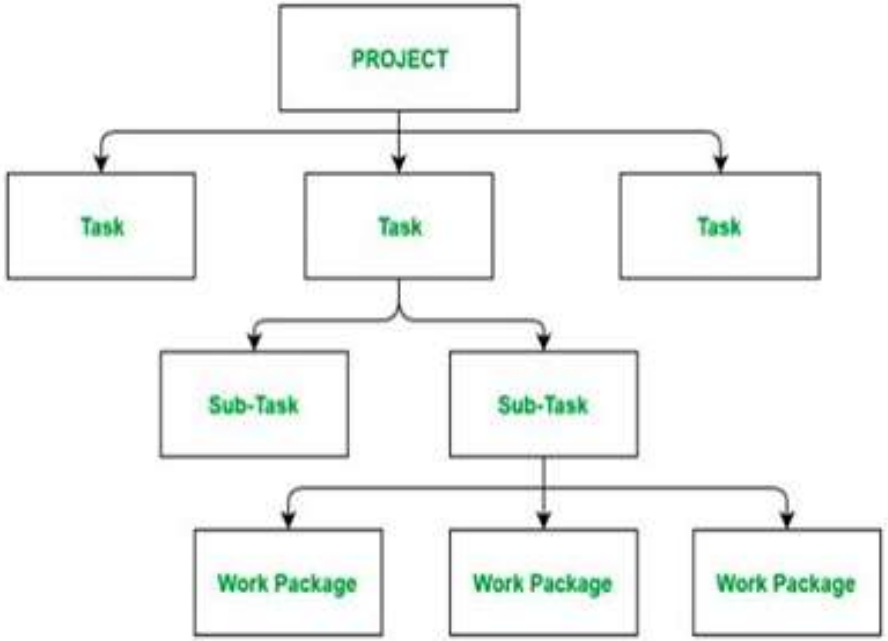
- The Validate Scope process focuses mainly on customer acceptance.
- It is when the project customer formally accepts all project deliverables & end of each phase.
- During the process, the customer gives feedback on the work that was performed.

Step 6: Control Scope

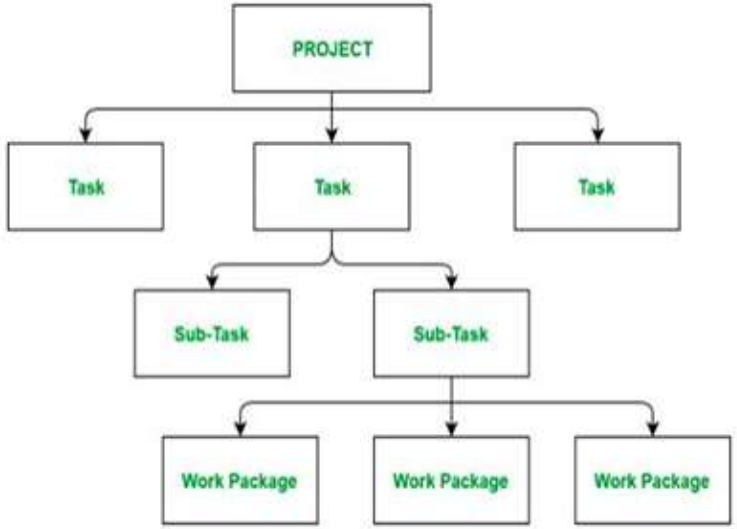
- It involves monitoring the status of the project and managing changes in scope.
- Efficient in dealing with Time & Cost management.
- Calculate how does it impact the project and its process.

Work Breakdown Structure

- A Work Breakdown Structure includes dividing a large and complex project into simpler, manageable and independent tasks.
- It used to organize and plan projects, programs and portfolio.
- It follows a Top-Down & Hierarchical view approach.
- The root of this tree is labelled by the Project name itself.
- Perform by Project Manager & Subject Matter Expert.



Working of Work Breakdown Structure



- **Step 1:** Project managers decide project name at top/root of WBS.
- **Step 2:** Project managers identifies the main deliverables of the project.
- **Step 3:** These main deliverables are broken down into smaller higher-level tasks.
- **Step 4:** This complete process is done recursively to produce much smaller independent tasks.
- **Step 5:** Choose Task Owner. They need to get the job done.

- Depends on project manager that up to which level of detail they want to break down project.
- Lowest level tasks are simplest & independent tasks it takes less than weeks of work.

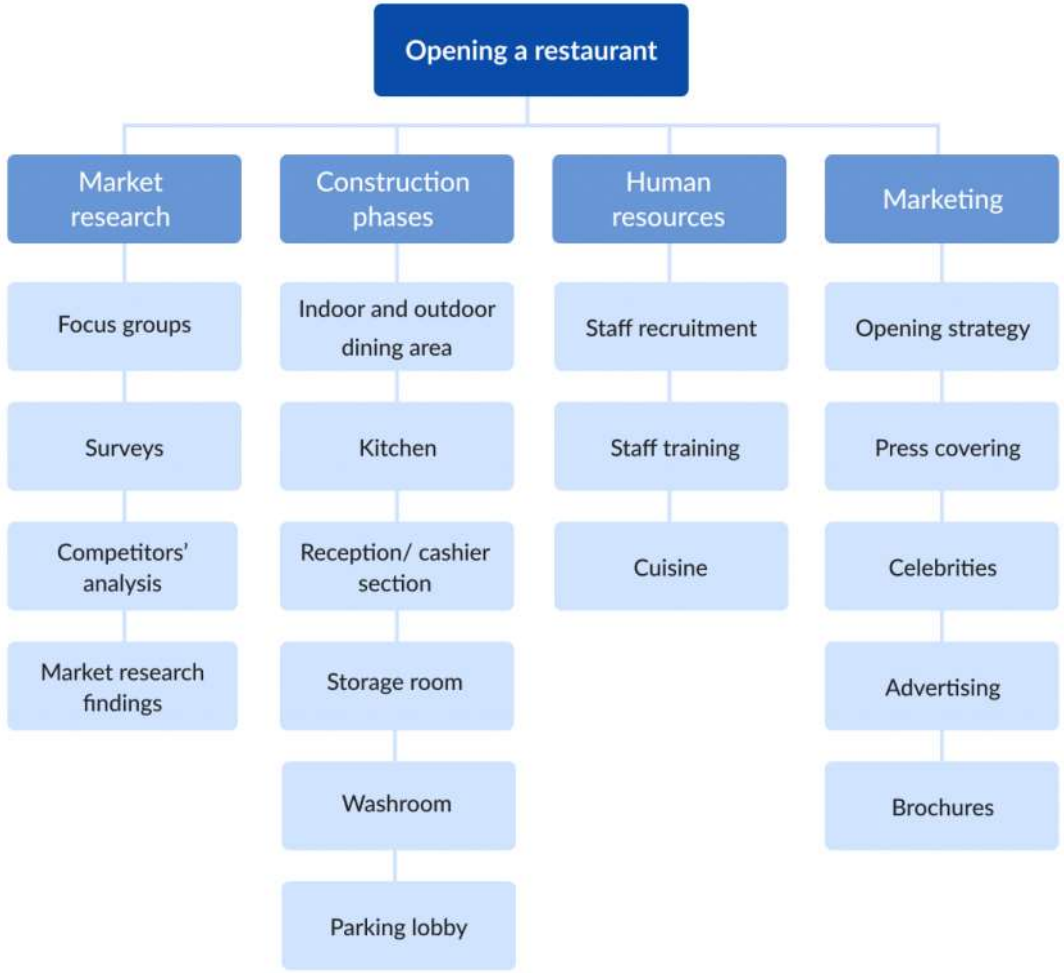
Components of Work Breakdown Structure

1. **WBS Dictionary:** Document that defirres the various WBS elements.
2. **WBS Levels:** Determines the hierarchy level of a WBS element.
3. **Task:** Main deliverable tasks. Status, Description. Task owner, Dependencies and Duration.
4. **Sub Task:** Divided tasks of Main deliverable tasks.
5. **Work Packages:** Lowest level of WBS. Small group of tasks.
6. **Control Account:** Group work packages and measure their status.
7. **Project Deliverables:** Desired outcome of project tasks and work packages.

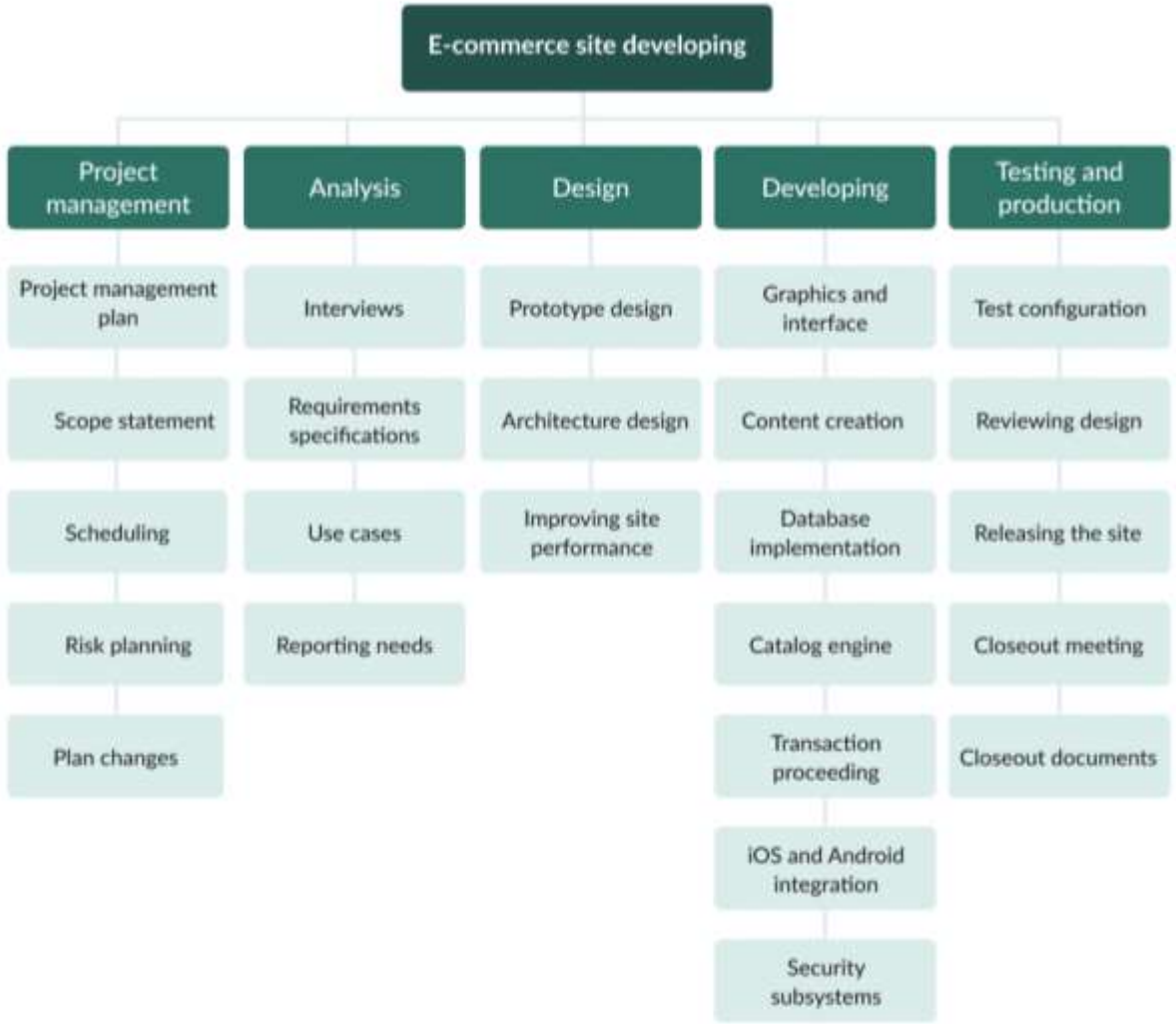
Why Work Breakdown Structure?

1. It allows easy management of the project.
2. It helps in proper organization of the project by the top management.
3. Giving visibility to important activities.
4. Giving visibility to risky activities.
5. Illustrate the correlation between the activities and deliverables.
6. It allows to do a precise cost estimation of each activity.
7. It allows to estimate the time that each activity will take more precisely.
8. Better communication with your team members regarding tasks
9. The efficiency of a work breakdown structure can determine the success of a project.

Example 1: Work Breakdown Structure of Restaurant System



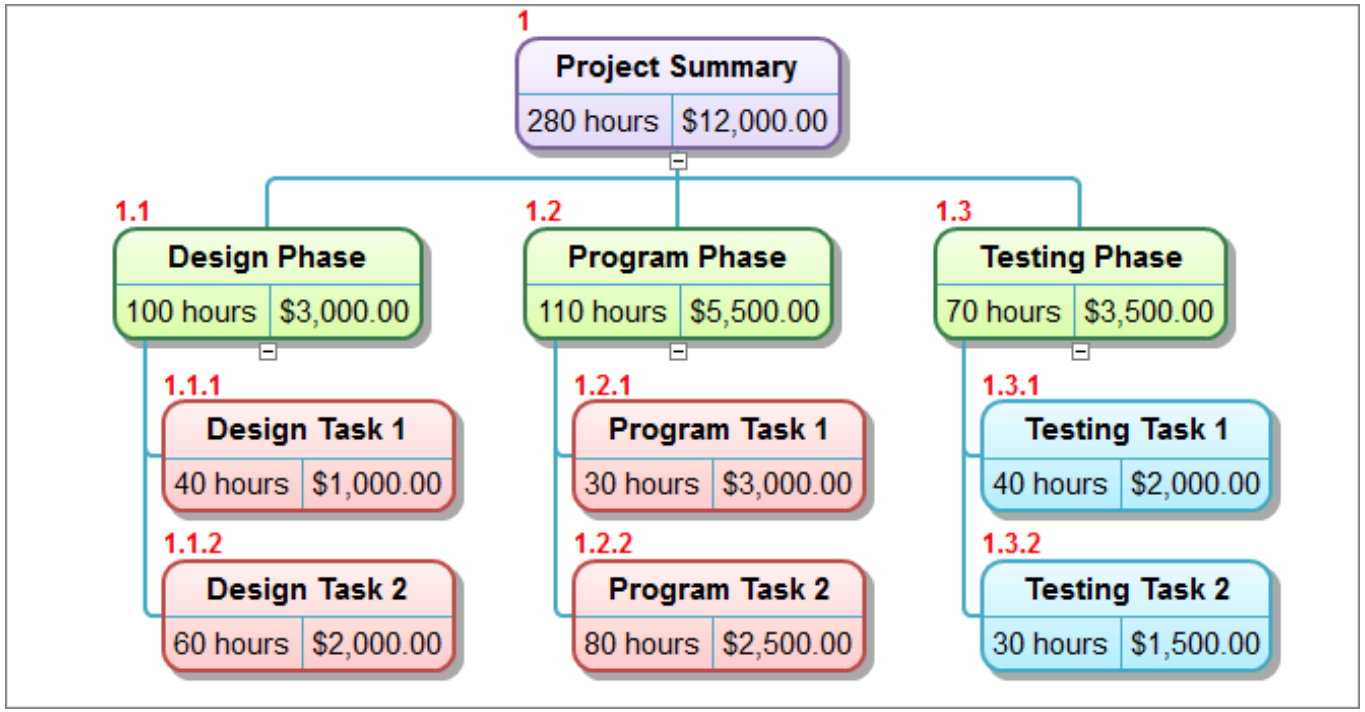
Example 2: Work Breakdown Structure of E-Commerce site



WBS Software Tools

- Many free and paid software tools can help you create WBS for your projects.
- **Common WBS Software tools are:**

1. EdrawMax
2. Lucidchart
3. SmartDraw
4. Visual Paradigm
5. MindView
6. Creately



Issues in Project Management

➤Project is not completed on schedule.

➤Changing customer requirements affect on schedule.

➤Technical difficulties are generate.

➤Miscommunication among project management. Essential software & hardware may be delivered late.

➤In large project, Software engineer perform multiple tasks parallel.

➤Tasks interdependencies are in project.

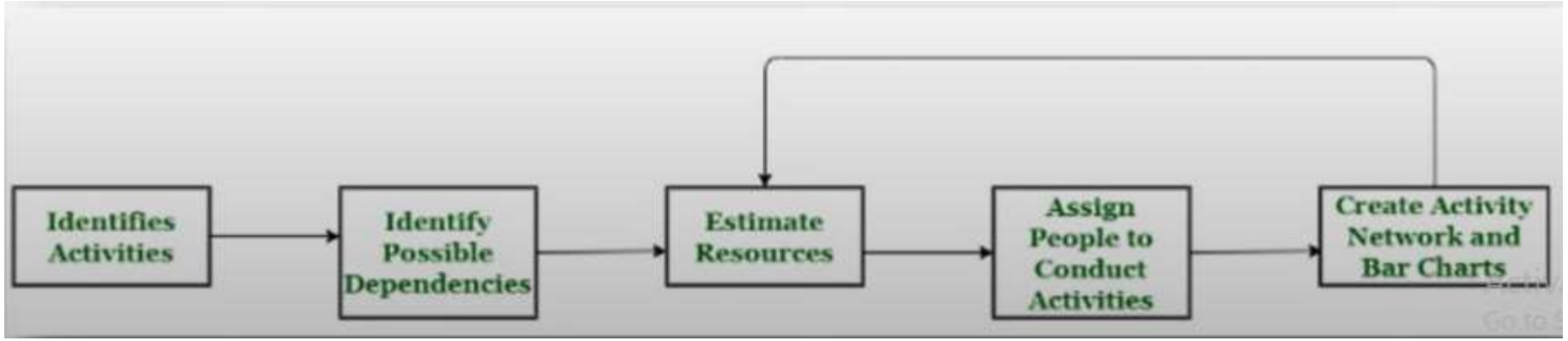
➤Risk is not considered at beginning of project.

Project Scheduling

- Project scheduling is responsible activity of project manager.
- Project schedule is a mechanism that is used to communicate and know about that tasks are needed and has to be performed in project.
- Project manager separate total work task in project into different activities. i.e. WBS
- Project Manager estimate time & recourses required to complete activities & organize them into coherent sequence.
- Effective project scheduling leads to success of project, reduced cost and increased customer satisfaction. Project

Project Scheduling Process

1. Identify all the functions/modules required to complete the project.
2. Break down large functions into small activities i.e. Develop WBS structure.
3. Determine the dependency among various activities.
4. Allocate resources to activities.
5. Assign people to conduct different activities.
6. Plan the beginning and ending dates for different activities.
7. Create Activity Network & Bar or Gantt charts.



Basic Principles of Project Scheduling

1. Compartmentalization: Project divide into number of manageable activities & tasks.

2. Interdependency: Certain tasks occur in sequence whereas other tasks occur in parallel.

3. Time Allocation: Each task has to be assigned specific time period i.e a start date & a completion date.

4. Effort Validation: PM ensure that allocated number of people work on given task.

5. Defined Responsibilities: Each task is assigned to specific member of the software team.

6. Defined Outcomes: Each task has a defined outcome. i.e., Work product

7. Defined Milestones: Every task should be associated with a project milestone. A milestone is accomplished when one or more work products has been reviewed for quality.

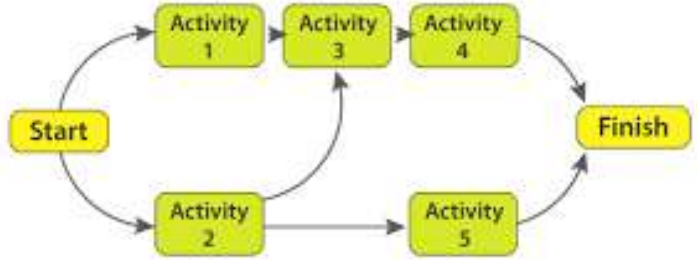
Project Scheduling Techniques

1. Critical Path Method:

- It help you determine both longest and shortest possible time take to complete a project.
- There are three essential elements: The tasks required to complete the project, Which tasks depend on the completion of others, A time estimate for each activity

Example:

- There are four tasks in the project - A, B, C. and D.
- Task B and D can only begin after task A completes, whereas task C has no such restriction.
- Task A will be time-sensitive as any delay in its completion can delay in task B & D.
- Called as Critical Task.
- This helps in identifying and separating the independent variables
- Finally, it adds milestones to the project.



Project Scheduling Techniques

2. Program Evaluation and Review Technique (PERT):

It is a way to schedule flow of tasks in a project and estimate total time taken to complete it.

PERT charts offer a visual representation of the major activities (and dependencies) in a project

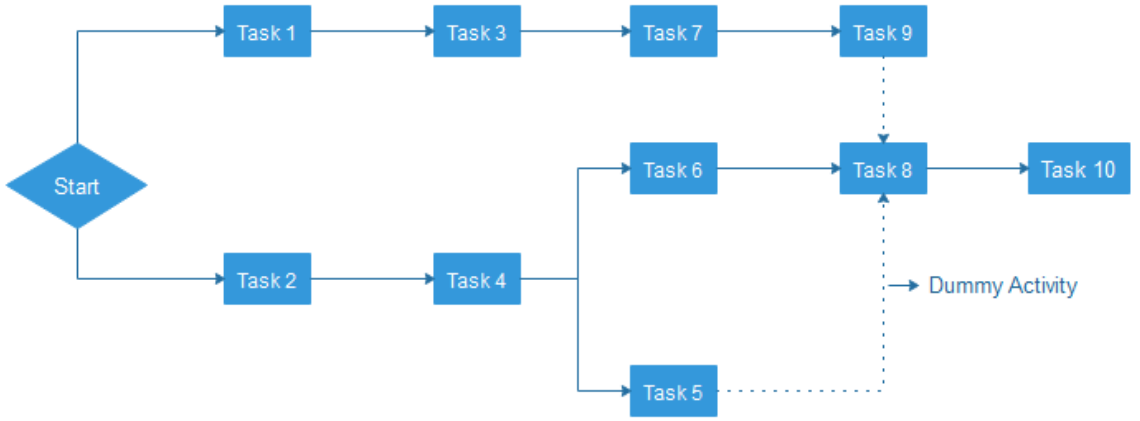
It calculate:

Optimistic time (O): Quickest time you can complete a project

Pessimistic time (P): Longest time it'll take to complete your project

Most likely time (M): How long it'll take to finish your project if there are no problems.

(O+4M+P)/6



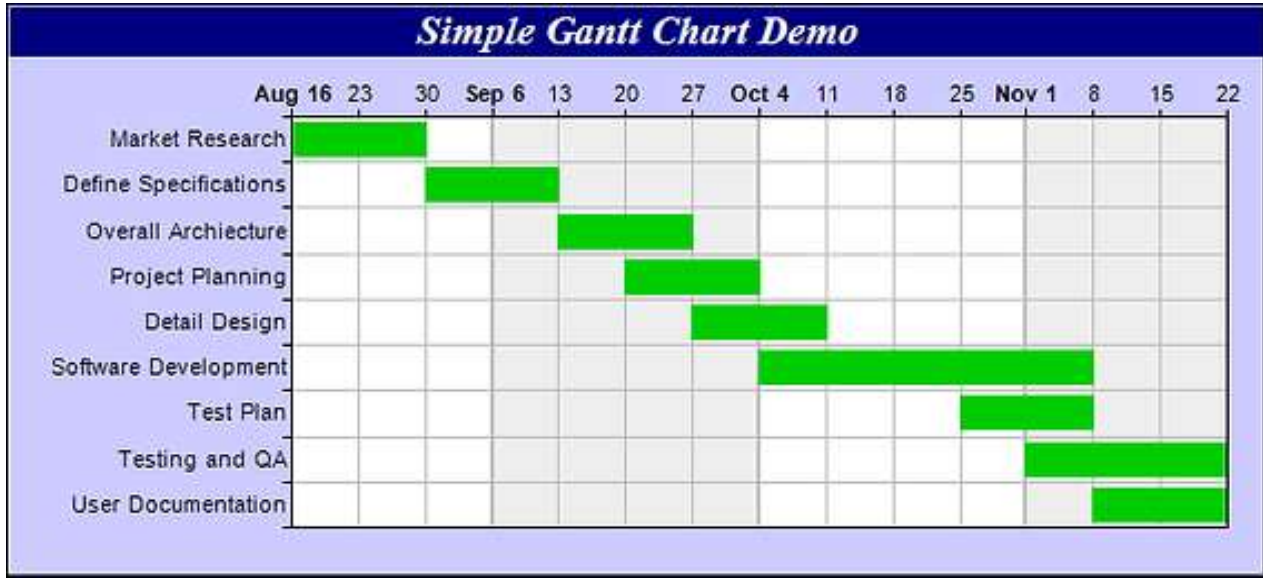
Project Scheduling Techniques

3. Gantt Chart:

• A Gantt chart is a type of bar graph that project managers use for planning and scheduling in complex project.

• It represent each task horizontally on a bar chart, which shows the start and end dates & they frequently include deadlines & dependencies of tasks.

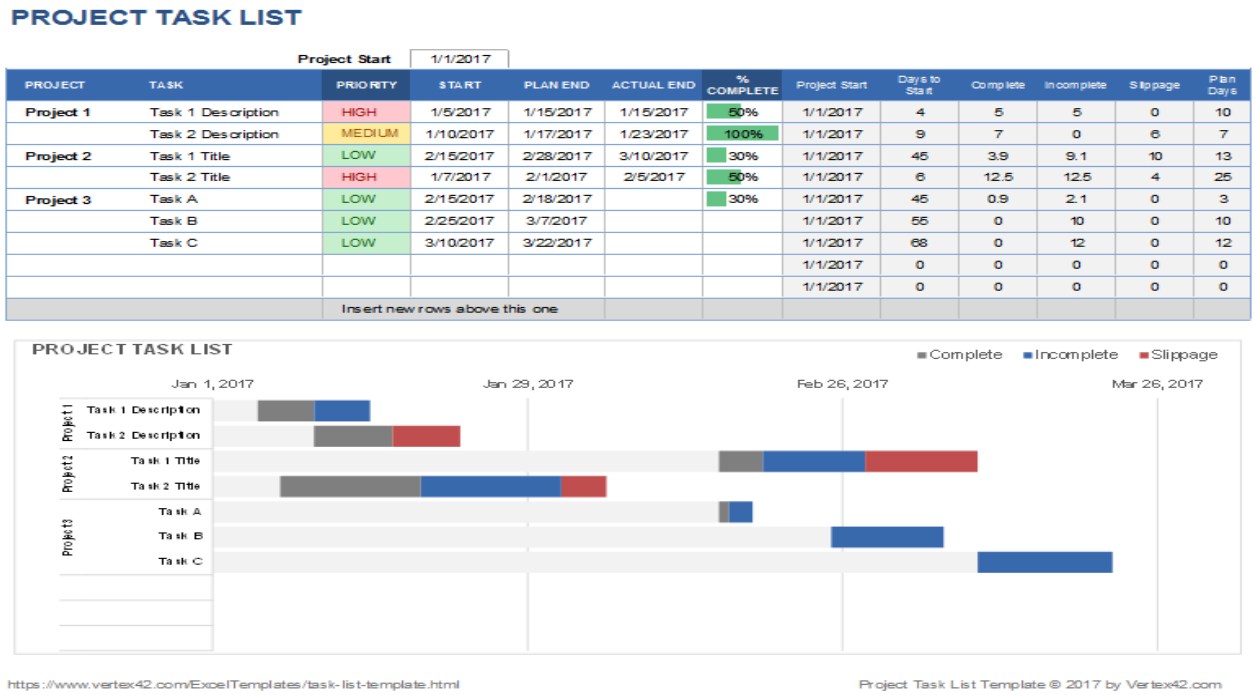
• It easier to visualize the progress of a project and see how different tasks interact with one another.



Project Scheduling Techniques

4. Task List:

- One of the simplest project scheduling techniques is the creation of a task list.
- Create task list using a word processor or spreadsheet software.
- It create a list of tasks and include important information like the task manager, start date, deadline & completion status.



PROJECT TASK LIST

Project Start 1/1/2017

PROJECT	TASK	PRIORITY	START	PLAN END	ACTUAL END	% COMPLETE	Project Start	Days to Start	Complete	Incomplete	Slippage	Plan Days
Project 1	Task 1 Description	HIGH	1/5/2017	1/15/2017	1/15/2017	50%	1/1/2017	4	5	5	0	10
	Task 2 Description	MEDIUM	1/10/2017	1/17/2017	1/23/2017	100%	1/1/2017	9	7	0	6	7
Project 2	Task 1 Title	LOW	2/15/2017	2/28/2017	3/10/2017	30%	1/1/2017	45	3.9	9.1	10	13
	Task 2 Title	HIGH	1/7/2017	2/1/2017	2/5/2017	50%	1/1/2017	6	12.5	12.5	4	25
Project 3	Task A	LOW	2/15/2017	2/18/2017		30%	1/1/2017	45	0.9	2.1	0	3
	Task B	LOW	2/25/2017	3/7/2017			1/1/2017	55	0	10	0	10
	Task C	LOW	3/10/2017	3/22/2017			1/1/2017	68	0	12	0	12
							1/1/2017	0	0	0	0	0
							1/1/2017	0	0	0	0	0
		Insert new rows above this one										

Project Scheduling Techniques

5. Fast Tracking:

• In Fast Tracking, Project is being implemented by either simultaneously executing many tasks or by overlapping many tasks to each other.

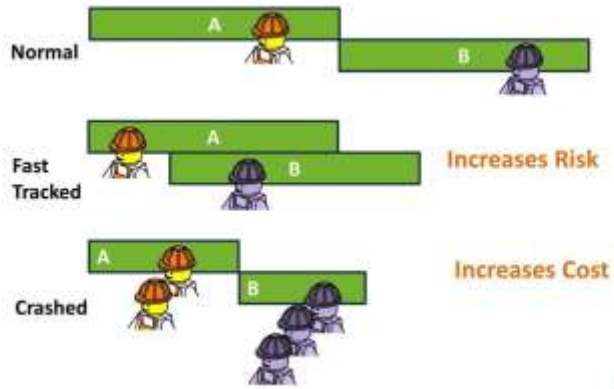
• **Example:** In software development project, designing and development can be taken up in parallel. Once design of essential features is ready and approved, development team can work on it. Meanwhile, the designing team will work on the remaining elements and functions.

6. Crashing:

• Crashing deals with involving more resources to finish the project on time.

• **Example:** Add more developer in project, Paying overtime to employee.

• Crashing can only be applied when it fits your project budget.

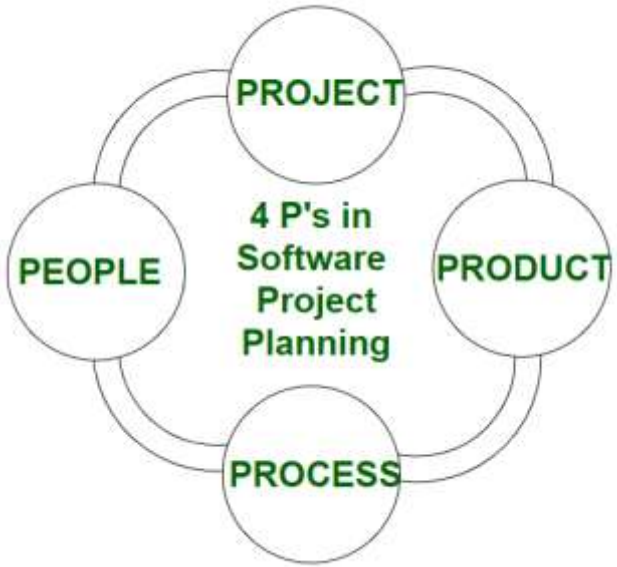


The Management Spectrum

- In software engineering, the management spectrum describes the management of a software project.
- For properly building a product, there's a very important concept handle by Project Manager.
- These components play a very important role in your project that can help your team meet its goals and objective

➤**Effective software project management focuses on the four P's:**

1. People
2. Product
3. Process
4. Project.



1. People

- The most important contribution in software project is not made by system or tool, It is done by people. i.e. Human Resources.
- The success of project depend on selecting right kind of people with right talent.

➤**Depending on there roles & responsibilities, following are the main categories:**

1. Senior Manager: Define the business issues & have significant influence on the project.

2. Project Manager: Plan, Motivate, Organize & Control the project work. Has problem solving skills & manageable team abilities.

3. Software Engineer: Who deliver technical skills which are necessary in project.

4. Customer: Who specify the requirement for the software project.

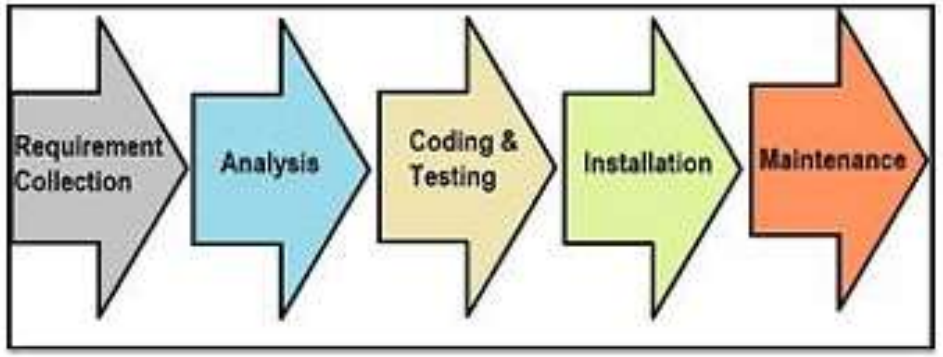
5. End Users: Who interact with the software product.

2. Product

- The product is the ultimate goal of the project.
- This is any types of software product that has to be developed or deliverable.
- Before a product can be planned, its objectives and scope should be established, alternate solutions should also be considered & technical & management constraints should be identified.
- Lack of these information, it is impossible to define reasonable and accurate estimation of the cost, identify possible risk & define manageable project schedule.

3. Process

- Project managers and team members should have a methodology and plan that complete project as per customer requirements.
- Without a clearly defined process, team members will not know what to do and when to carry out project activities.
- Using the right process will increase the project execution success rate that meets its original goals and objectives.
- The Process has several steps involved like, Documentation phase, Designing phase, Implementation phase, software configuration management, Deployment phase and Interaction phase.



4. Project

- The project is the complete software project that includes requirement analysis, development, delivery, maintenance and updates.
- The project manager plays a critical role for completing project.
- They are responsible to guide the team members to achieve the project's target and objectives.
- They helping & assisting them with issues, checking on cost and budget and making sure that the project stays on track with the given deadlines.
- They manage complete project activities to avoid project failure.

W5HH Principle

- Barry Boehm provided philosophy that prepares easy and manageable designs for software projects.
- The W5HH questions help project managers to guide objectives, timelines, responsibilities, management styles, technical approaches and resources in project.
- These method used for efficient software project management & development.

➤WWWWWHH Questions:

1. Why?
2. What?
3. When?
4. Who?
5. Where?
6. How?
7. How Much?

W5HH Questions

1. Why the system is going to be developed?

- This focus on the business reasons & problem statements for developing the software.
- Identify whether the project's purpose will justify the cost, time spent on it by people?

2. What is activities are needed to be done in this?

- It help team to establish project schedule by identifying key project tasks that are required by the customer.
- It mentioned guidelines & principle for determining tasks that need to be complete.

3. When will it be completed?

- When project tasks will be started and when they enter into the final stage to reach the goal.

W5HH Questions

4. Who is responsible for each activities?

- The role and responsibility of each member of the software team are defined.

5. Where are they organizationally located?

- Not all roles and responsibilities reside within the software team itself.
- The customer, users and other stakeholders also have responsibilities.

6. How will the job be done technically and managerially?

- Once the product scope is established, management and technical strategy for the project must be defined.

7. How much of each resource is needed?

- It figure out how much resources necessary to complete the project as per customer requirement & budget of the project.