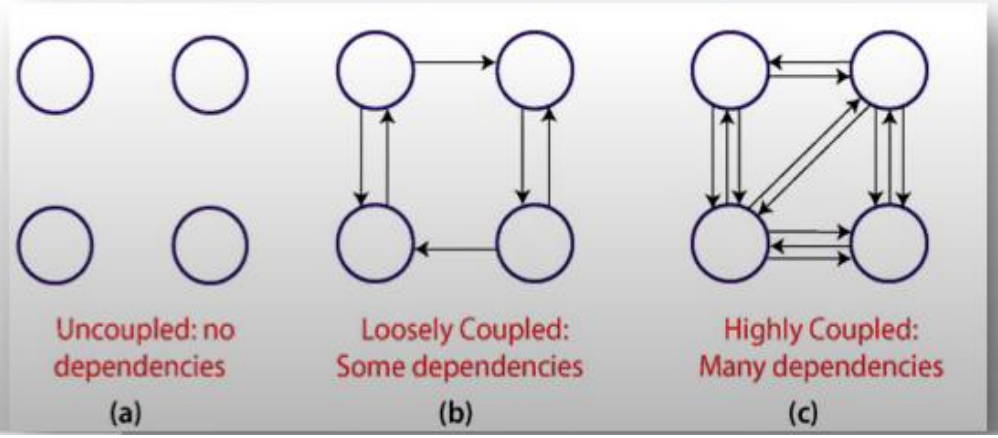
Unit- 4

About Coupling

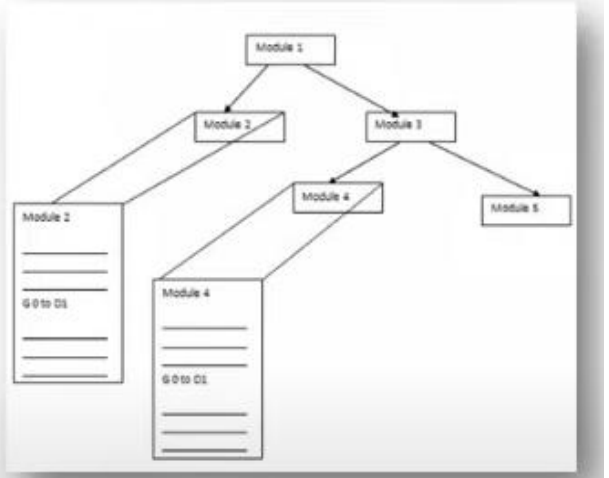
- The coupling is the degree of interdependence or number of relations between software modules.
- Two modules that are tightly coupled are strongly dependent on each other.
- However, two modules that are loosely coupled are not much dependent on each other.
- A **good design** is the one that has Low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



Types of Coupling

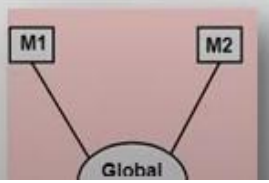
Type 1: Content Coupling

- 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.



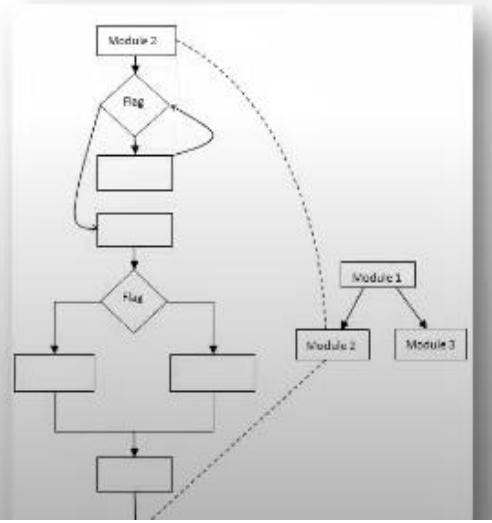
Types of Coupling

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.

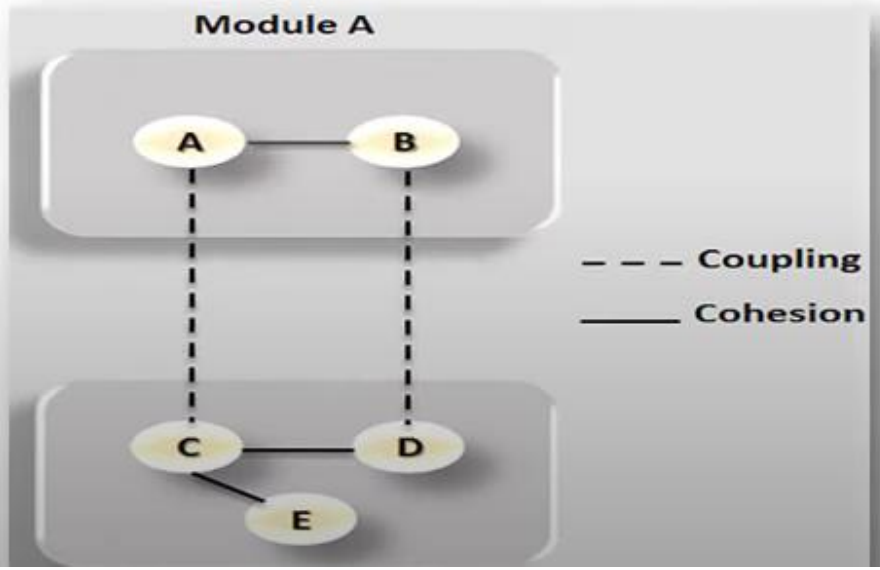
Type 4: Control Coupling

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



About Cohesion

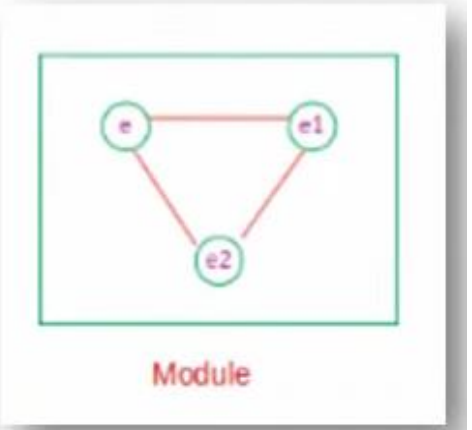
- Cohesion defines to the degree to which the elements of a module belong together or interrelated.
- Thus, cohesion measures the strength of relationships between pieces of functionality within a given module.
- A **good software design** will have high cohesion.



Types of Cohesion

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.

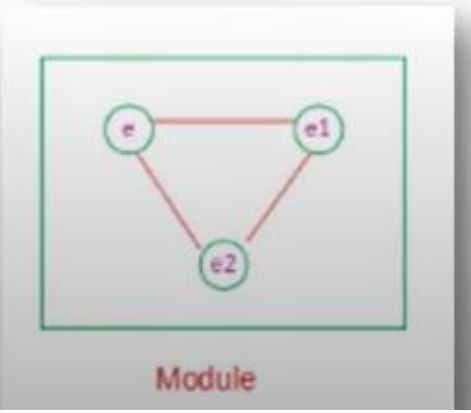
Types of Cohesion

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.



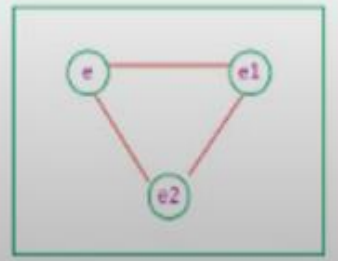
Types of Cohesion

Type 6: Sequence Cohesion

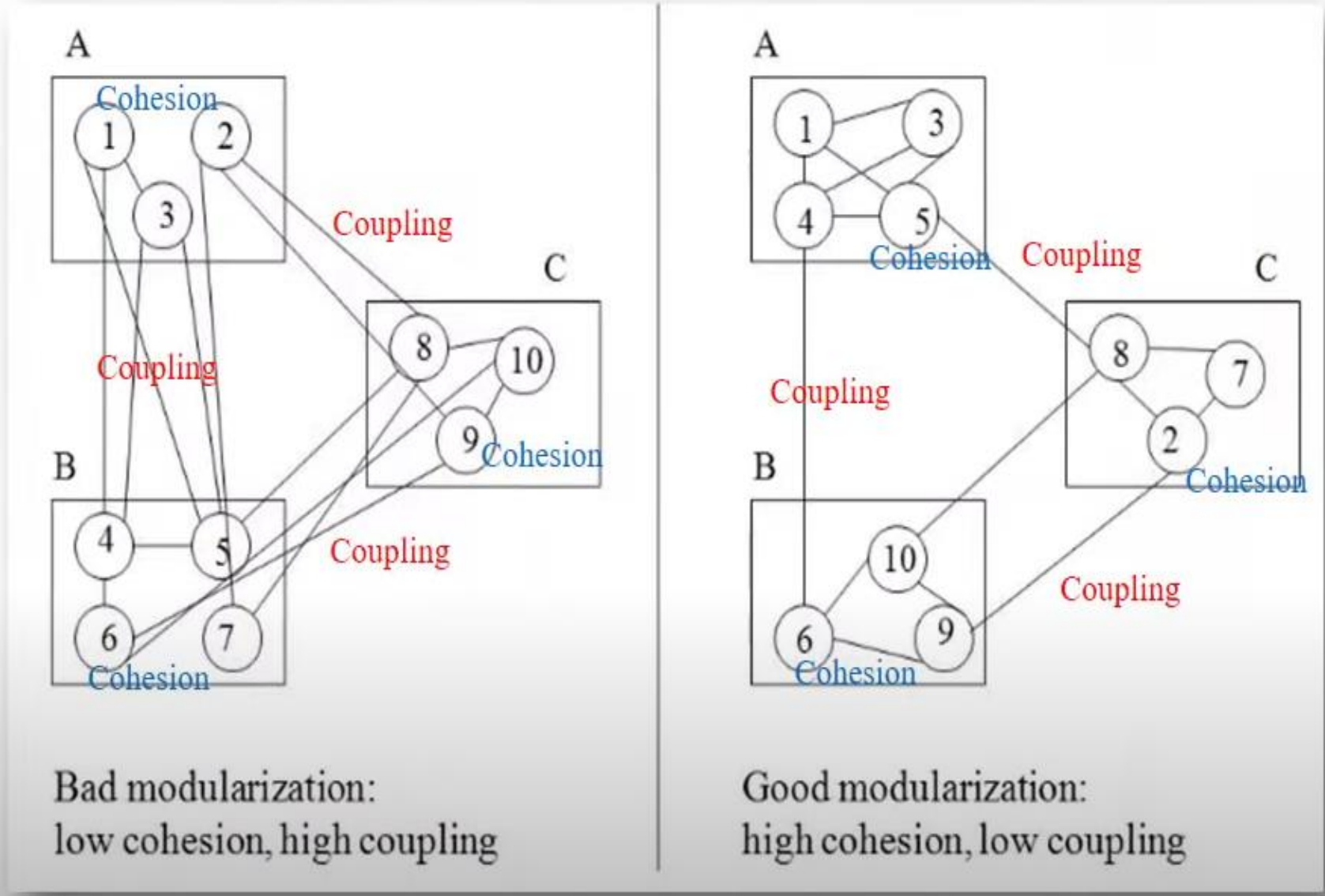
- Output of one element treats as an input to the other elements inside the same module.
- **Example:** Enter the numbers -> Perform Addition of that numbers -> Display Addition.

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



Good & Bad Software System Design



Design models

Design Models

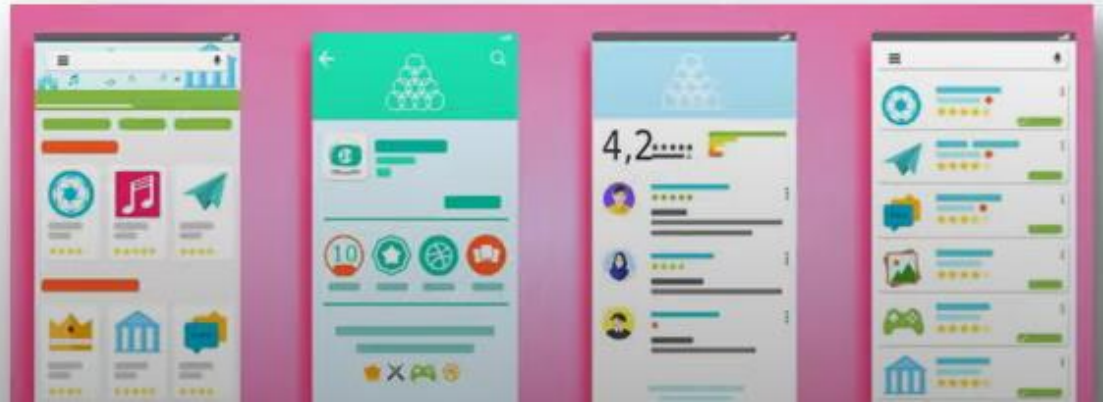
- The design phase of software development, transforming the customer requirements as described in the SRS documents into design forms.
- Designing a model is an important phase and is a multi-process that represent the data structure, program structure, interface characteristic and procedural details.

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:**

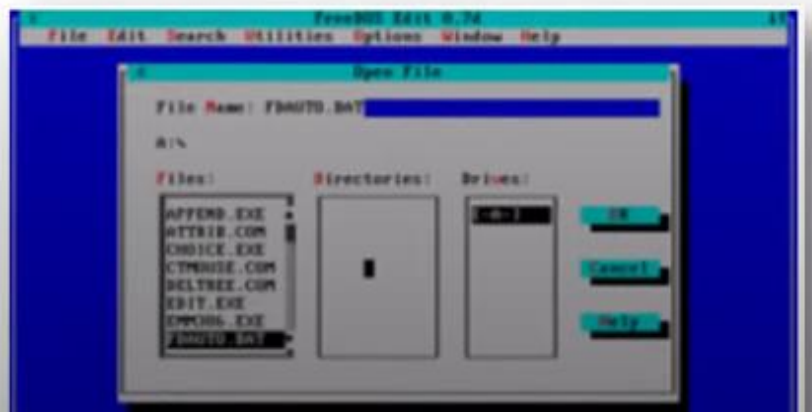
1. Attractive
2. Simple to use
3. Responsive in short time
4. Clear to understand
5. Consistent on all interfacing screens

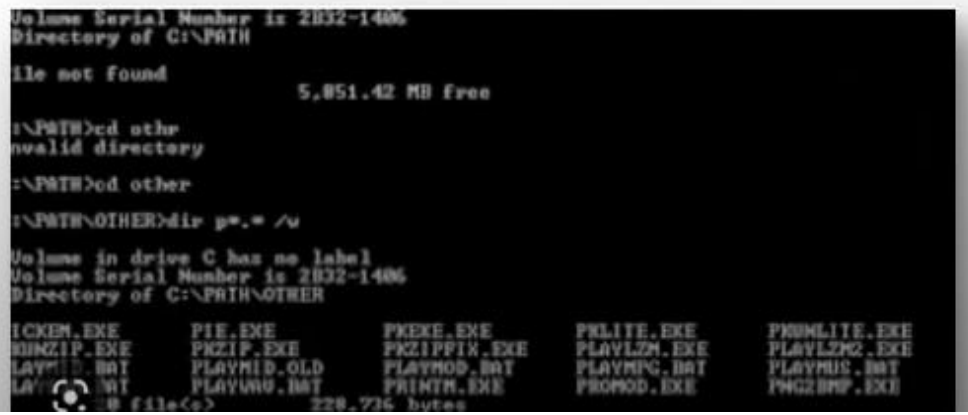


Types of User Interface

Type 1: Text-Based User Interface OR Command Line Interface

- Text based interface primarily used keyboard handling data.
- Command Line Interface provide command prompt or coding tools where the user types the command towards the system.
- The user needs to remember the syntax of command and its use.
- Command Line Interface used by Technical people or Programmers.

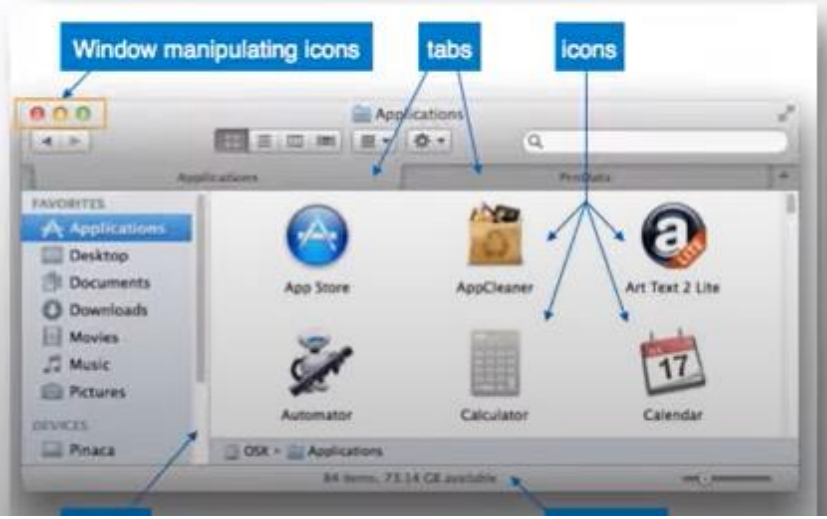




Types of User Interface

Type 2: Graphical User Interface

- Graphical User Interface provides the simple interactive interface to interact with the system.
- GUI can be a combination of both hardware and software.
- Graphical user interfaces are easy to learn as compared to the command-line interface.
- GUI provides multiple windows to the user simultaneously to interact with the system.



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.

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.

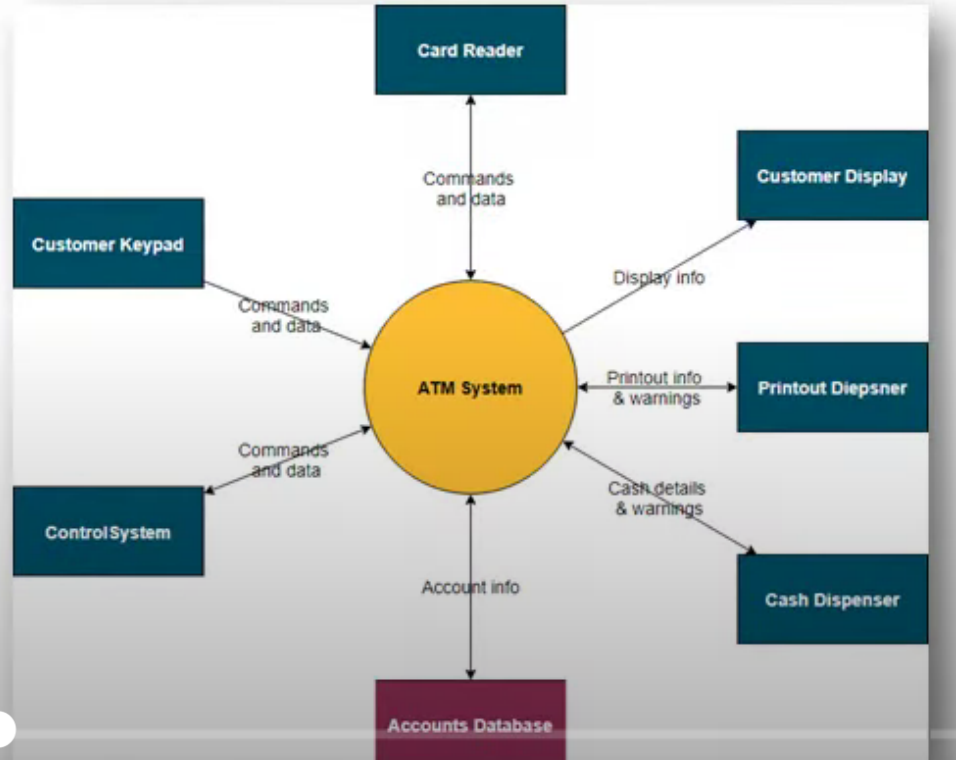
➢**There are some common decisions that should be taken care of in any design process.**

- ❑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?

Taxonomy of Architectural styles

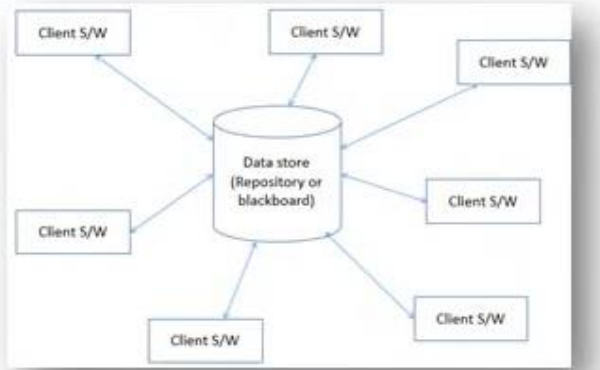
- **Architectural styles is establish a complete structure & components of all over the system.**

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.

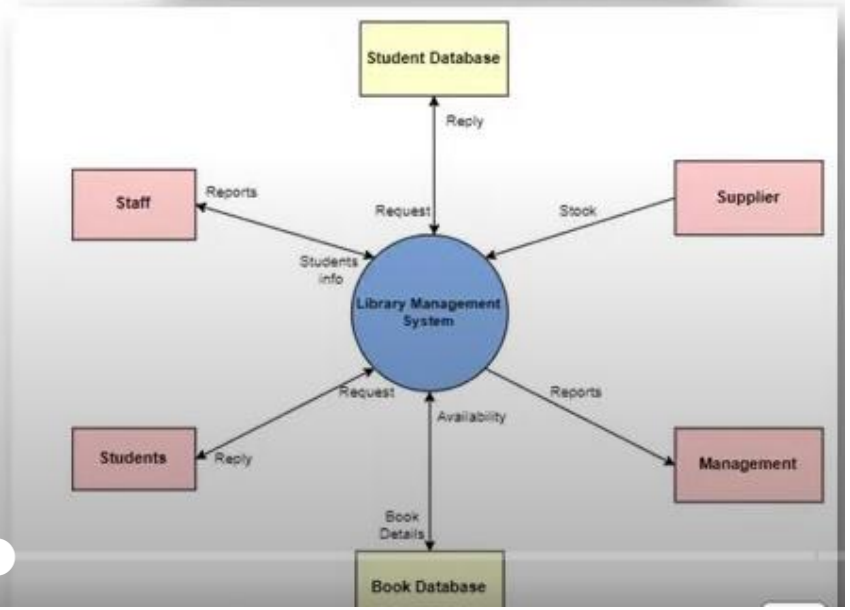


Advantages:

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

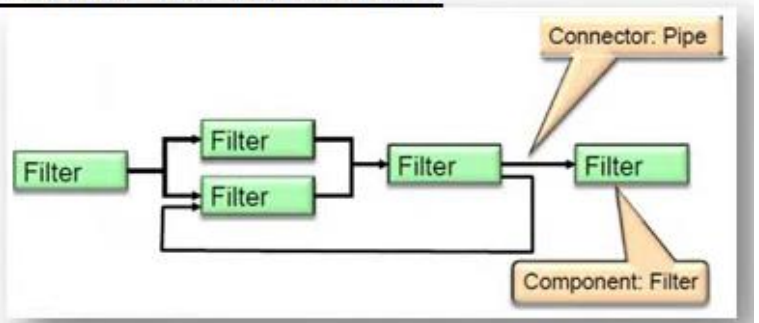
Disadvantages:

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.

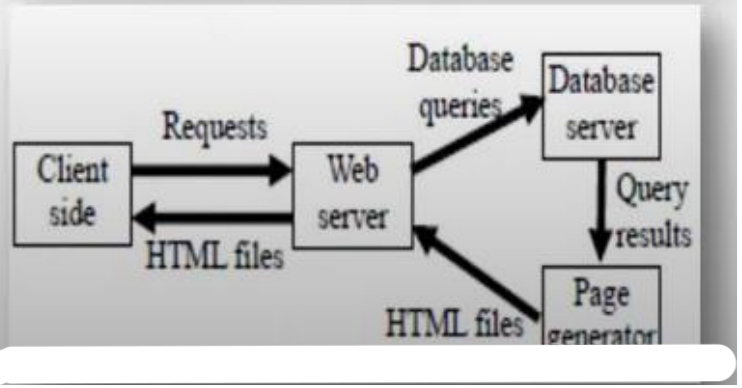


Advantages:

1. With this design, concurrent execution is supported.

Disadvantages:

1. Does not allow greater user engagement.



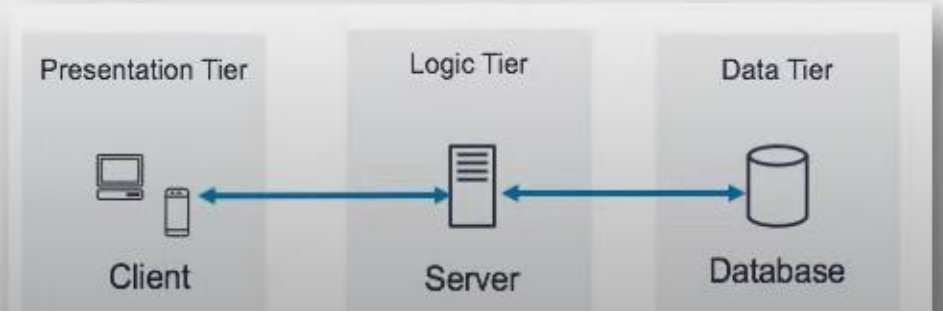
Object Oriented Architecture

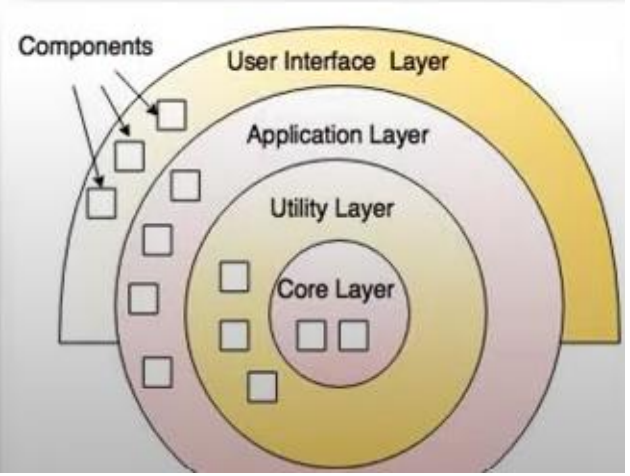
- Objects are the foundational building blocks for all kinds of software applications.

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.

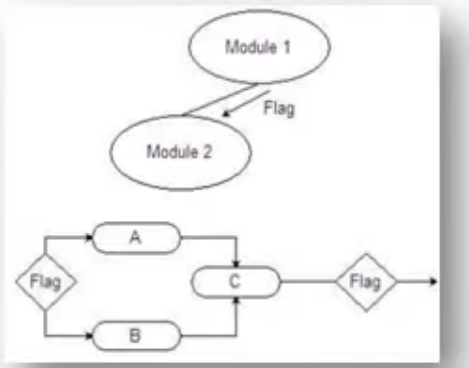


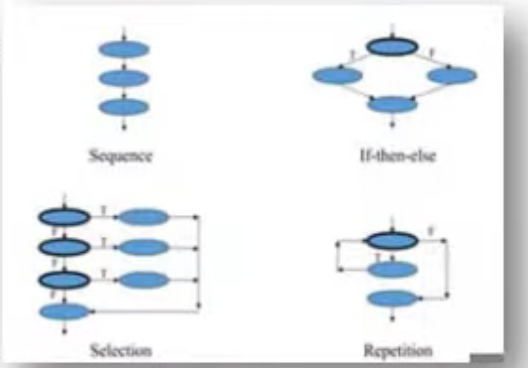


Component Level Design Model

➢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.

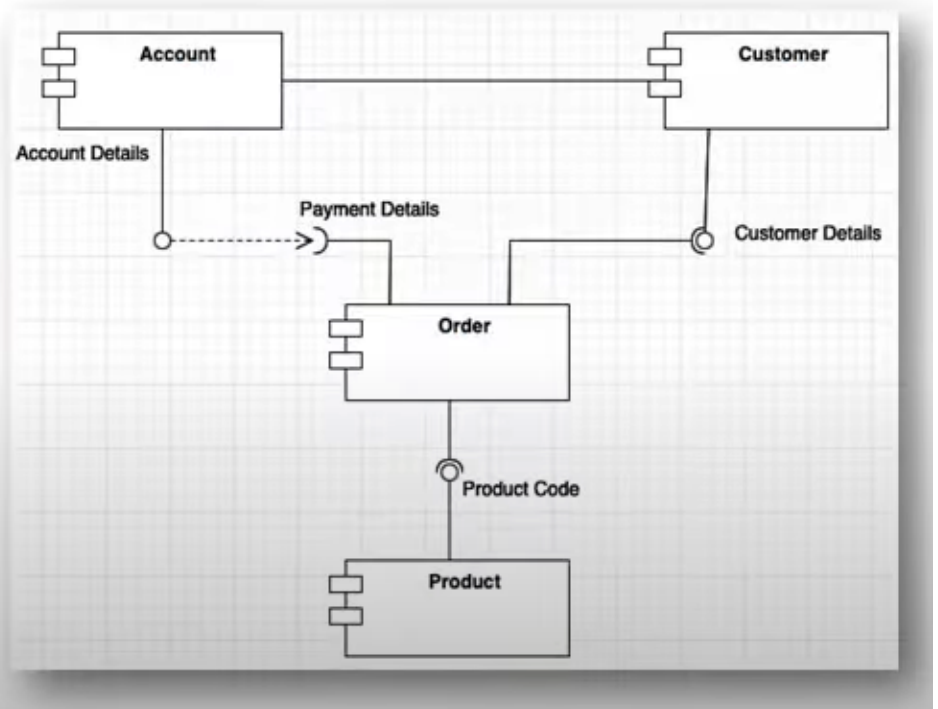
Steps of Component Level Design

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 & there Interactions through Activity & State diagram.
6. Elaborate deployment diagram. **Ex.** Hardware, OS, Server of components.

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.**

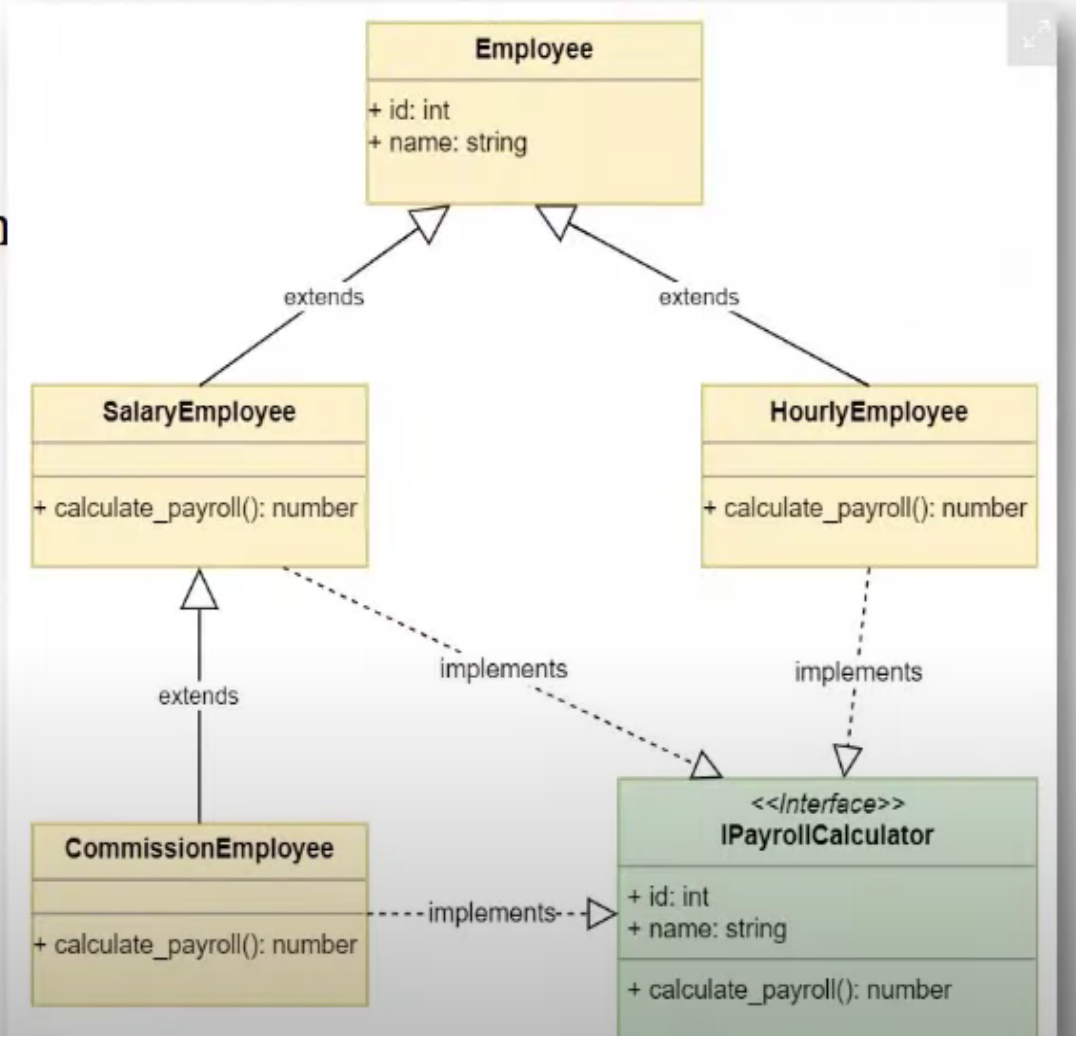
1. Customer will go online site and search product.
2. By selecting product, the customer will place an order.
3. System will store customer & his account information in Order Database.
4. The order would be generated against the customer and stored in the Database.



Component Design View

1. Object Oriented View:

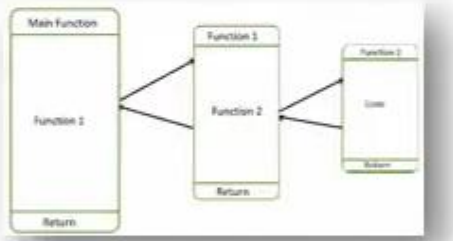
- An object-oriented view is a set of collaborating classes in module mentioned in architectural design
- It view all the objects, methods, properties and functionalities mentioned in classes.
- It also view how classes in each module are connected with each other.
- It detailed out data structure &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

Component Level Design for WebApps

➢**What is Web Apps Component?**

- A well-defined cohesive function that provides data processing for an end user
- A cohesive package of content and functionality that provides the end user with some required capability.
- Web applications include online forms, shopping carts, word processors, spreadsheets, video and photo editing, file conversion, file scanning & emails such as Gmail, Yahoo etc.

➢**Component-level design for WebApps often incorporates elements of**

1. Content design
2. Functional design.
3. Deployment design

Component Level Design for WebApps

1. Content Design at Component Level:

- It focus on objects & there collaboration in packaged for presentation to a WebApp end user.
- It should be tuned to the characteristics of the WebApp to be built.
- Size and complexity between objects their interrelationships grows, it may be necessary to organize content in a way that allows easier reference and design.

2. Functional Design at Component Level:

- Perform localized processing to generate content and navigation capability in a dynamic fashion.
- Provide sophisticated database query and access.
- Establish data interfaces with external corporate systems.

Component Level Design for WebApps

3. Deployment Design at Component Level:

- It indicates how software functionality & subsystem will be allocated within the physical computing environment that will support the software.
- **Examples:** Personal Computer, Internet connectivity, Server, Control Panel like Security etc.