UNIT-6

Architectural Design

⊛ **Introduction:** Architectural design is concerned with understanding how a software system should be organized and designing the overall structure of that system. It identifies the main structural components in a system and the relationships between them. Architectural design is the first stage in the software design process. It is the link between design and requirements engineering.

⊛. Architectural design decisions:

During the architectural design process, system architects have to make a number of structural decisions. Based on their knowledge and experience, they have to consider the fundamental questions as:

→ Is there a generic application architecture that can be used?
→ What approach will be used to structure the system?
→ What architectural styles are appropriate?
→ How will the system be distributed?
→ How should the architecture be documented?
→ How will the architectural design be evaluated?

The architectural design of the system affects the performance, dependability, maintainability etc. of the system. The particular architectural design and ~~choosen~~ structure choosen for an application depends on the non-functional system requirements: Maintainability, Performance, Availability, Security, and Safety.

⊛. Architectural Views:

A view is a representation of an entire system from the perspective of a related set of concerns. It describe the system from the viewpoint of different stakeholders such as end-users, developers, project managers and testers. It provides four essential views:

1) **Physical view:** It shows the system how hardware and software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

2) **Logical view:** It shows the key abstractions in the system as objects or object classes. It will be possible to relate the system requirements to entities in this view.

3) **Process view:** It shows how the system at runtime is composed of interacting processes. This view is useful for making judgements about non-functional system characteristics such as performance and availability.

4) **Development view:** It shows how the software is decomposed for development. This view is useful for software managers and programmers.

⊛ Architectural Patterns:

Architectural Patterns are a ways of presenting, sharing, and reusing knowledge about software systems that has been adopted in a number of areas of software engineering. Architectural pattern is a stylized, abstract description of good practice, which has been tried and tested in different systems and environments. So, a architectural pattern should describe a system organization that has been successful in previous systems. It should include information on when ~~it~~ is appropriate and when not to use that pattern, and details on the patterns strengths and weaknesses.

There are many generic patterns that can be used in software development. Some examples of patterns that are widely used and that capture good architectural design principles are as follows:

1> Model-view-controller pattern:

Model-view-controller pattern in short is called as mvc pattern. MVC pattern separates presentation and interaction from the system data. The system is structured into three logical components that interact with each other.

Model component: It manages the system data and associated operations on that data.

View component: It defines and manages how the data is presented to the user.

Controller component: It manages user interaction (e.g, key press, mouse click etc.) and passes these interactions to the View and the model.

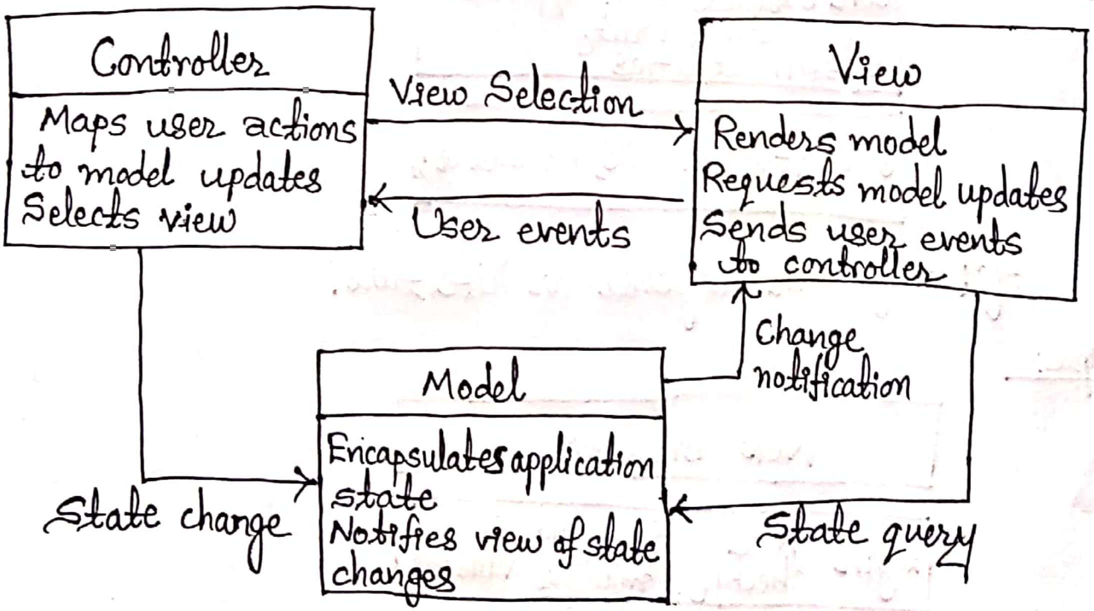


Fig: The organization of model view controller.

Advantages:

→ It allows data to change independently of it's representation and vice-versa.

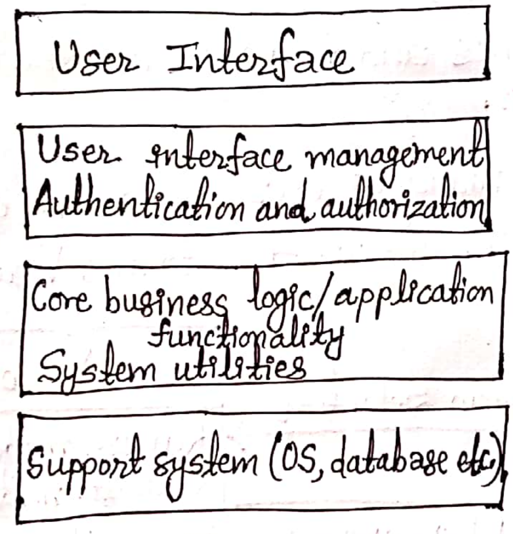
→ Support Presentation of the same data in different ways with changes made in one representation shown in all of them.

Disadvantages:

→ It can involve additional code and code complexity when the data model and interactions are simple.

2) Layered Architecture pattern:

In layerd architecture pattern, the system functionality is organized into separate layers, and each layer only relies on the facilities and services offered by the layer immediately below it. It organizes the system into layers, with related functionality associated with each layer. A layer provides services to the layer above it, so the lowest level layers represent core services that are likely to be used throughout the system.



User Interface

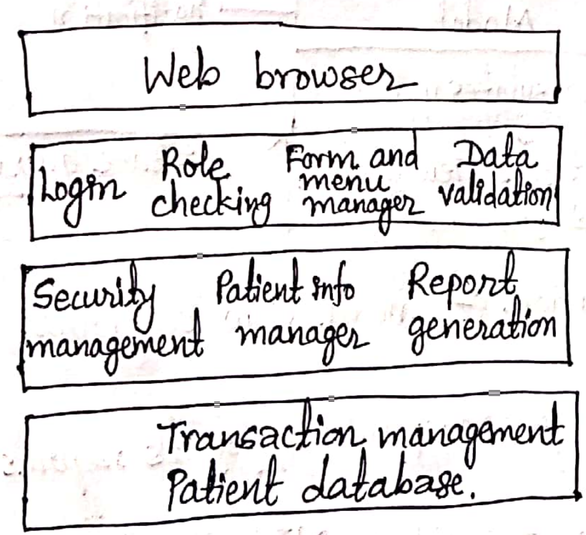
User interface management
Authentication and authorization

Core business logic/application functionality
System utilities

Support system (OS, database etc)

Fig: A generic layered architecture.

Example:



Web browser

Login | Role checking | Form and menu manager | Data validation

Security management | Patient info manager | Report generation

Transaction management
Patient database.

Fig: the layered architecture of a health care system.

3> Repository architecture pattern:

The repository pattern, describes how a set of interacting components can share data. All the data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository.

This model is suited to applications in which data is generated by one component and used by another. Examples of this type of system include command and control systems, management information systems, CAD systems, and interactive development environments for software.

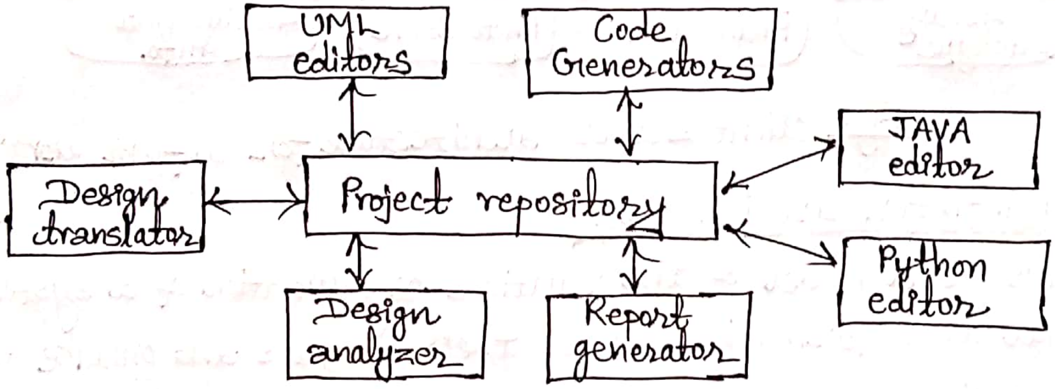


Fig: a repository architecture for IDE

Advantage: Components can be independent, they do not need to know of the existence of other components. Changes made by one component can be propagated to all components.

Disadvantage: The repository is a single point of failure so problems in the repository affect the whole system.

4> Client-server architecture pattern:

It is a system that follows the Client-server pattern is organized as a set of services and associated servers, and clients that access and use the services. The major components of this model are:

Clients: that call on the services offered by servers. There will normally be several instances of a client program executing concurrently on different computers.

Servers: that offer services to other components. Examples of servers include print servers that offer printing services, file servers that offer file management services etc.

Network: that allows clients to access these services. Client-server systems are usually implemented as distributed systems, connected using Internet protocols.

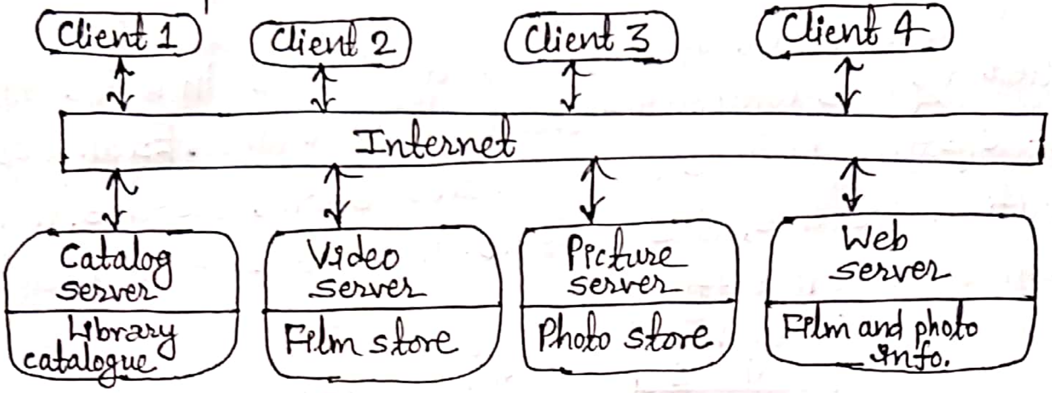


Fig: client server architecture for a film library.

5) Pipe and filter architecture:

This is a model of the runtime organization of a system where functional transformations process their inputs and produce outputs. The processing of the data in a system is organized so that each processing component is discrete and carries out one type of data transformation. The data flows from one component to another for processing.

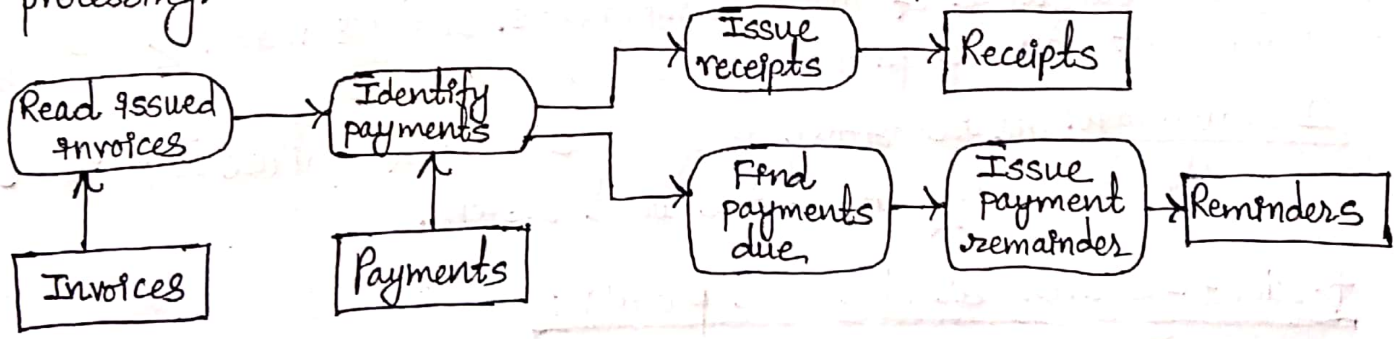


Fig: An example of the pipe and filter architecture.

⊛Application Architectures:

Application architecture describes the patterns and techniques used to design and build an application. Application Architecture is the process of defining the framework of an organization's application solutions against business requirements. It helps to ensure that applications are scalable and reliable. Application architecture encapsuilates the principal characteristics of a class of system and designers can use of models of application architecture in number of ways:

→While developing any software system if the designer is unfamiliar with the type of application that he/she is developing then initial design can be made using generic application architecture.

→We can use application architecture as means of judging components for reuse.

→Application architecture can be used to compare the applications of same type.

Q. Why modular decomposition is used in architectural design?

Ans: Modular decomposition is a process of decomposing subsystems into modules. After decomposition of the system into subsystems, subsystems must be decomposed into modules. Two modular decomposition models are used for:

Object model: where the system is decomposed into interacting objects.

Data-flow model: where the system is decomposed into functional modules which transform inputs to outputs. Also known as pipeline model.

Q. What are the activities of architectural design process?

Ans: Architectural design process includes following activities:

System structuring: The system is decomposed into major sub-systems and communication mechanisms are identified.

Control modelling: A model of the control relationships between the sub-systems is established.

Modular decomposition: The identified sub-systems are decomposed into lower-level modules (components, objects etc.)