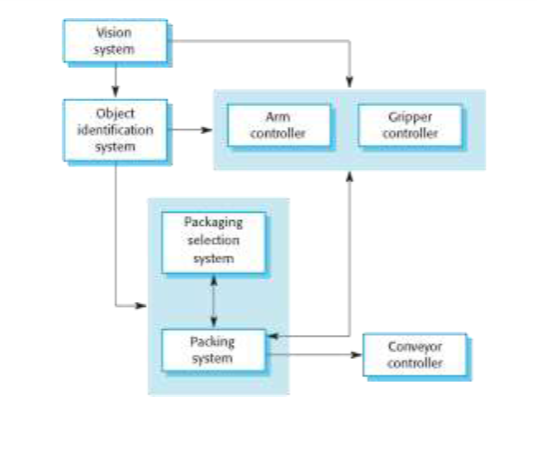
CHAPTER 6

1. Definition and Purpose:

Software architecture involves identifying subsystems, their framework for control, and communication.

Architectural design is an early stage in system design, connecting specification and design processes.



2. Architectural Abstraction:

- Architecture in the small deals with individual program structures.

- Architecture in the large involves complex enterprise systems distributed across multiple computers.

3. Advantages of Explicit Architecture:

- Facilitates stakeholder communication.

- Enables system analysis for non-functional requirements.

- Supports large-scale reuse, including product-line architectures.

4. Architectural Representations:

- Simple block diagrams are commonly used but criticized for lacking semantics.

- Box and line diagrams, though abstract, are useful for communication and project planning.

5. Use of Architectural Models:

- Facilitates discussion and communication during system design.

- Documents the designed architecture, depicting components, interfaces, and connections.

6. Architectural Design Decisions:

- Various decisions impact the non-functional characteristics of the system.

- Decisions include selecting generic application architecture, distribution strategy, architectural styles, structural approach, module decomposition, control strategy, evaluation method, and documentation format.

7. Architecture Reuse:

- Systems in the same domain often share similar architectures.

- Application product lines utilize a core architecture with variants.

- Systems can be designed around architectural patterns or styles.

8. Architecture and System Characteristics:

- Considerations for performance, security, safety, availability, and maintainability.

9. Architectural Views:

- Different views or perspectives are useful for designing and documenting system architecture.

- Multiple views, such as decomposition, runtime processes, and distribution of components, are necessary.

10. 4 + 1 View Model:

- A model with logical, process, development, and physical views, related through use cases or scenarios.

11. Architectural Patterns:

- Patterns are a means of representing, sharing, and reusing knowledge.

- Architectural patterns are stylized descriptions of good design practice.

- Patterns should indicate when they are and are not useful.

12. Model-View-Controller (MVC) Pattern:

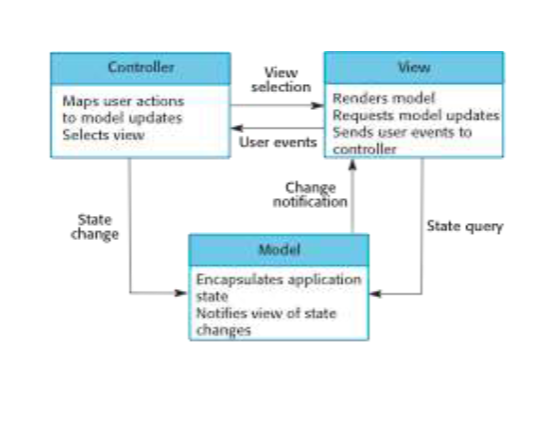
- Separates presentation and interaction from system data.

- Comprises three logical components: Model, View, and Controller.

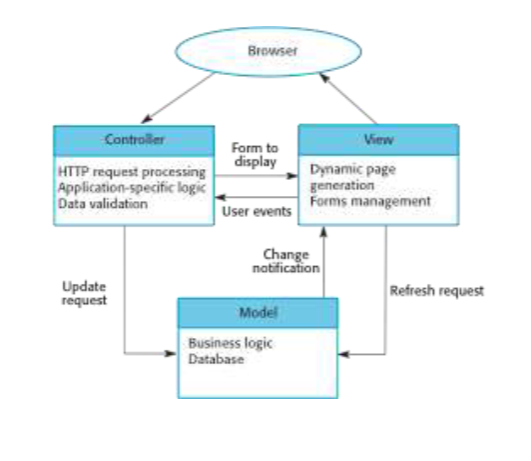
- Used when there are multiple ways to view and interact with data, especially when future requirements are unknown.

- Advantages include independent changes to data and its representation, supporting different presentations of the same data. However, it may involve additional code complexity for simple data models and interactions.

The organization of the Model-View-Controller



Web application architecture using the MVC pattern



The document also includes diagrams illustrating the organization of the Model-View-Controller and a web application architecture using the MVC pattern.

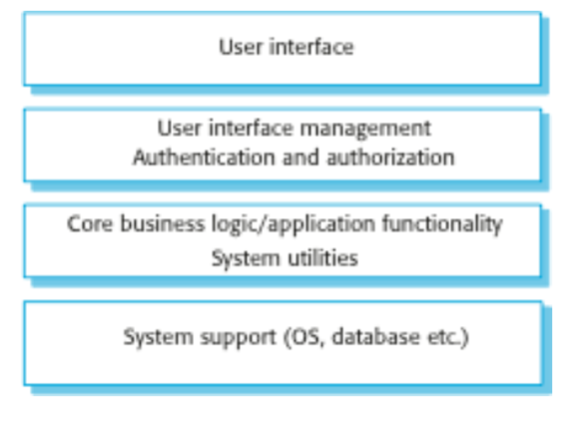
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 21 – 40

14. Layered Architecture:

- Used for modeling subsystem interfaces, organizing the system into layers.

- Supports incremental development, where changes in one layer affect only the adjacent layer.

- May be artificial in practice.

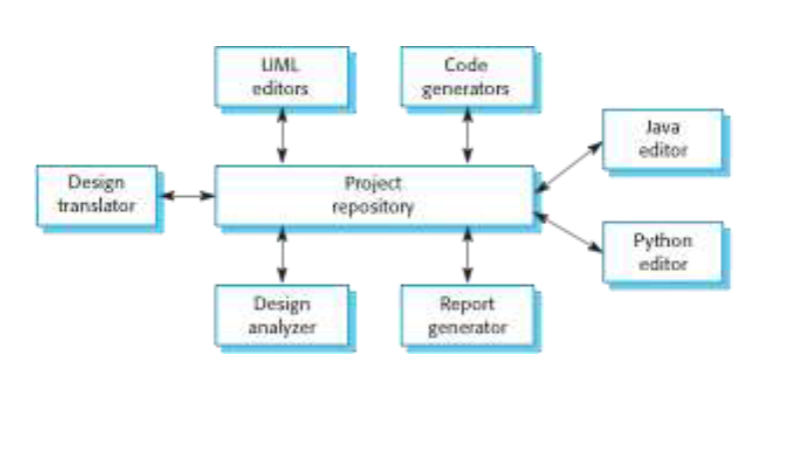


15. Repository Architecture:

- Data exchange can be through a central repository or explicit passing between subsystems.

- Repository pattern manages all system data centrally, allowing components to access it indirectly.

- Suitable for systems generating large volumes of long-term data.

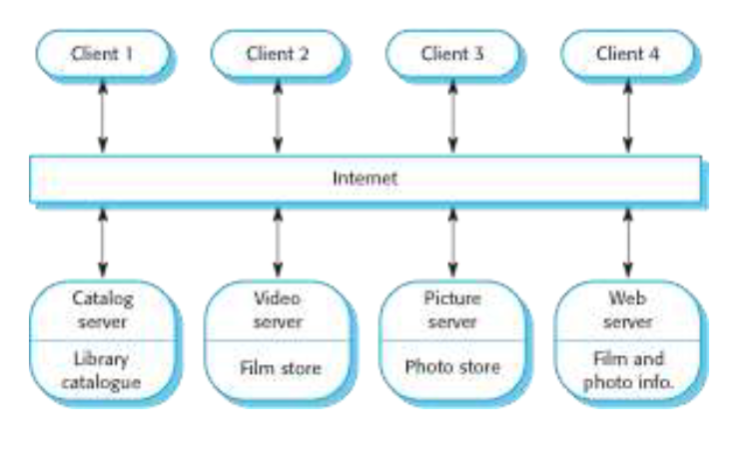


16. Client-Server Architecture:

- Organizes functionality into services delivered from separate servers, accessed by clients.

- Suitable for shared database access or variable system loads.

- Servers can be distributed, but each service is a single point of failure.

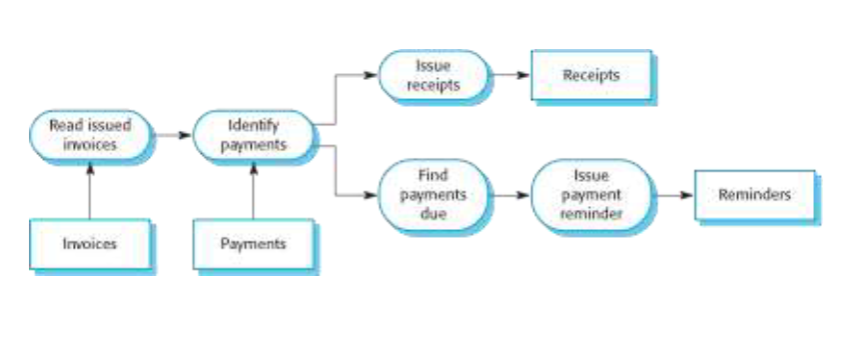


17. Pipe and Filter Architecture:

- Functional transformations are discrete components processing data sequentially.

- Commonly used in data processing applications, both batch- and transaction-based.

- Supports easy understanding and transformation reuse but requires agreed data format between components.



18. Application Architectures:

- Systems designed to meet organizational needs often share common architectures.

- Generic application architectures can be adapted for specific requirements.

- Used as a starting point for design, checklist, team organization, component assessment, and vocabulary for application types.

19. Transaction Processing Systems:

- Handle user requests for information or updates in a database.

- Transactions are coherent sequences of operations satisfying a goal.

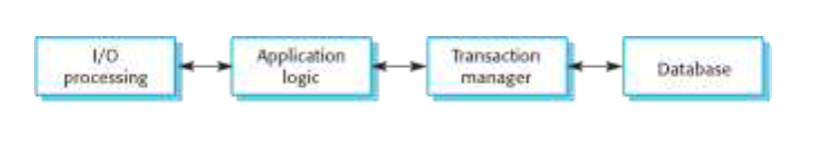
- Users make asynchronous requests processed by a transaction manager.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 41 – 54

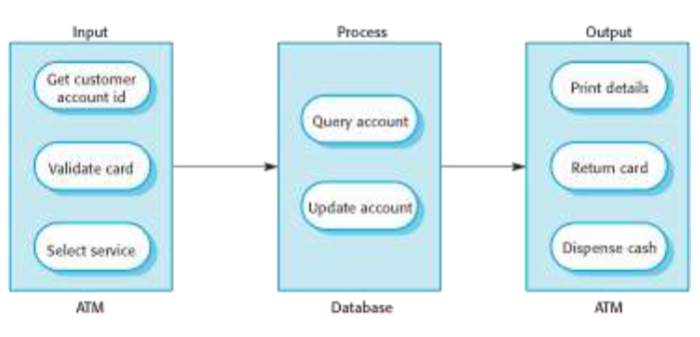
20. Transaction Processing Applications:

- Presents the structure of transaction processing applications.

- Features the software architecture of an ATM system.



The software architecture of an ATM system



21. Information Systems Architecture:

- Information systems typically have a layered architecture, emphasizing transaction-based systems.

- Layers include user interface, user communications, information retrieval, and the system database.

22. Web-based Information Systems:

- Modern information and resource management systems are web-based.

- E-commerce systems are highlighted as examples, implementing a layered structure with a specific application layer for tasks like shopping cart functionality.

23. Server Implementation:

- Describes the implementation of web-based systems using multi-tier client-server architectures.

- Components include web servers for user communications, application servers for logic and data, and database servers for transaction management.

24. Language Processing Systems:

- Examines systems that accept natural or artificial language as input and generate alternative representations.

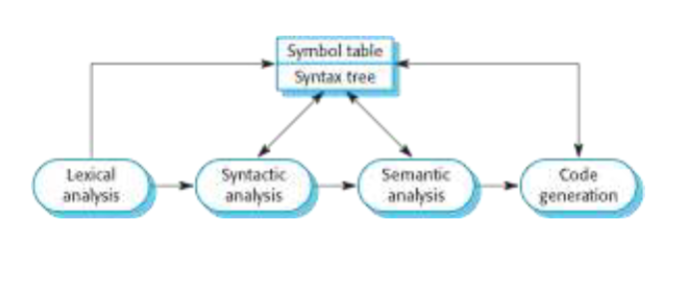
- Meta-case tools are mentioned, which process descriptions and rules to generate tools.

25. Compiler Components:

Discusses the components of a language compiler, including lexical analyzers, symbol tables, syntax analyzers, syntax trees, semantic analyzers, and code generators.

**26. Pipe and Filter Compiler Architecture:**

- Illustrates a pipe and filter architecture for a compiler, emphasizing sequential processing stages.



**27. Key Points:**

- Highlights the importance of application system architecture models for understanding, comparison, validation, and assessing components for reuse.

- Defines transaction processing systems as interactive systems allowing remote database access 0and modification by multiple users.

- Describes language processing systems as tools for translation and execution of instructions in a specified language, including a translator and an abstract machine.