東北大學
Northeastern University

软件工程

张爽

东北大学软件学院

Chapter 6 Object-Oriented Design

- Software System Design
- Object-Oriented Design
- Testing for OOD
- Case Study
- Challenges of OOD

So far, what have we learned?

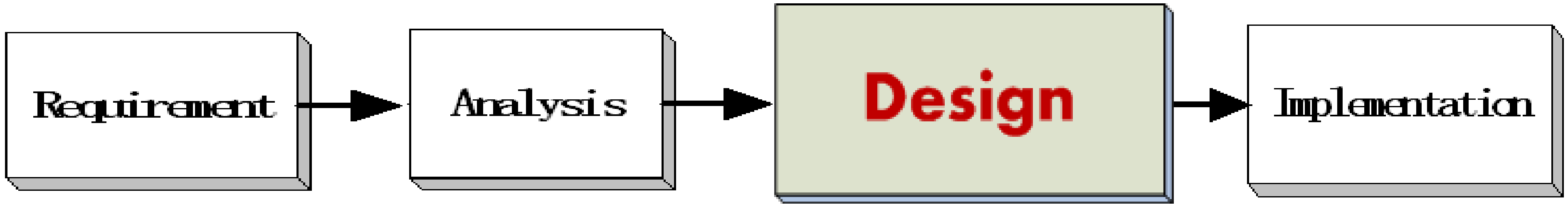
Phase 1. Requirements

Requirements documentation

Phase 2. Analysis

---- Object-Oriented Analysis

- Use case modeling
- Class modeling
- **Dynamic modeling**



What $\longrightarrow$ How

東北大學
Northeastern University

Phase 3. Design

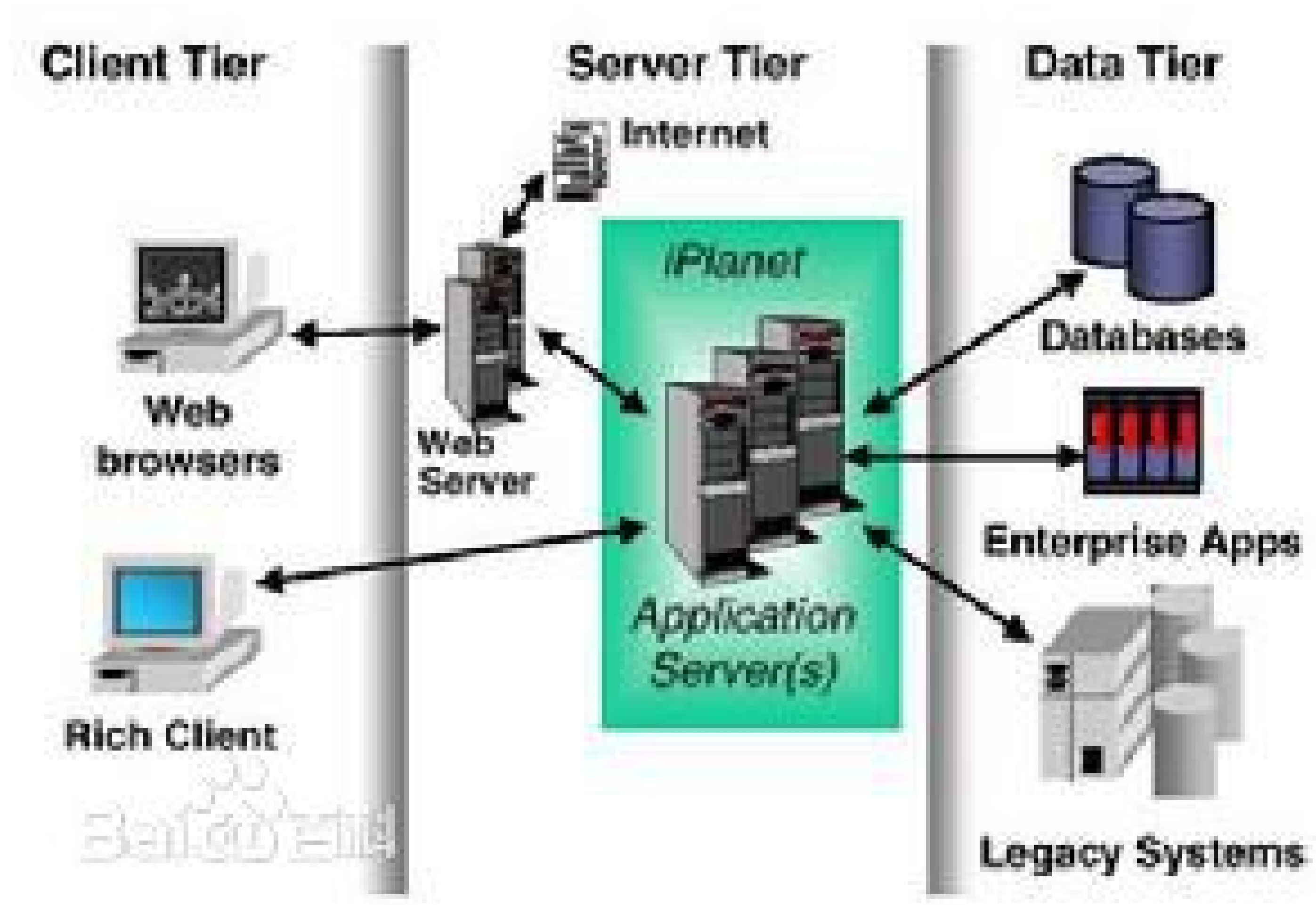
6.1 Software System Design

Architectural Design

Architectural Design

- Architecture ----*skeleton*, the overall structure of the software system, and the ways which provide conceptual integrity for the system.
- A software architecture is a description of how a software system is organized.
- Properties of a system such as performance, security, and availability are influenced by the architecture used.

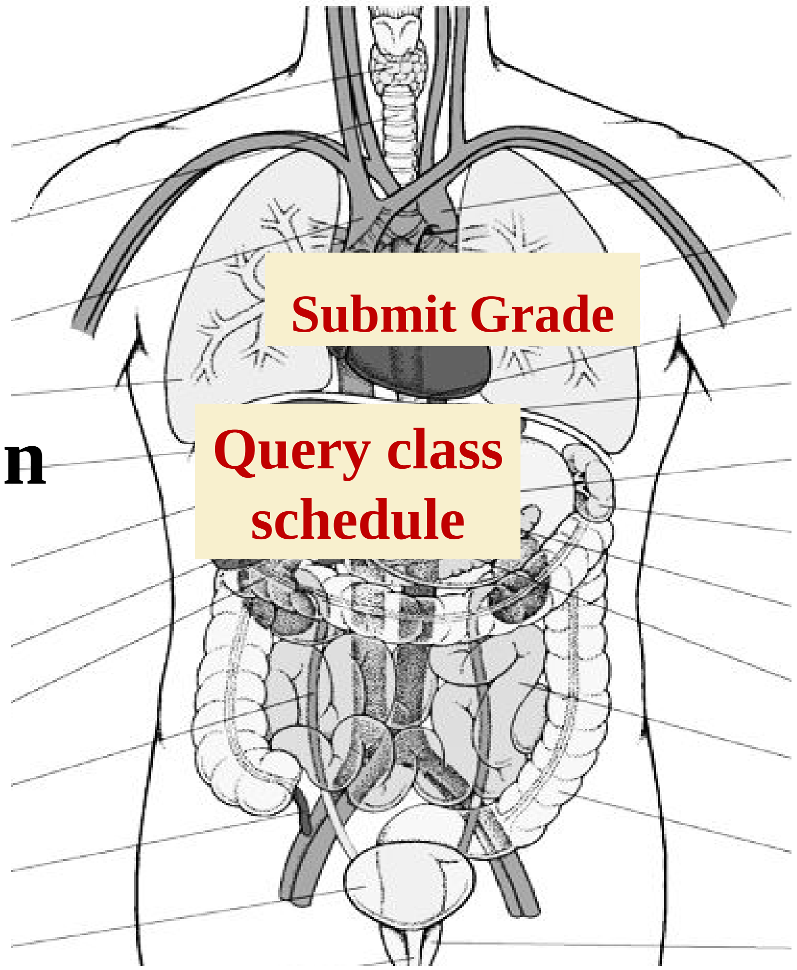
Software Architecture Example



Northeastern University

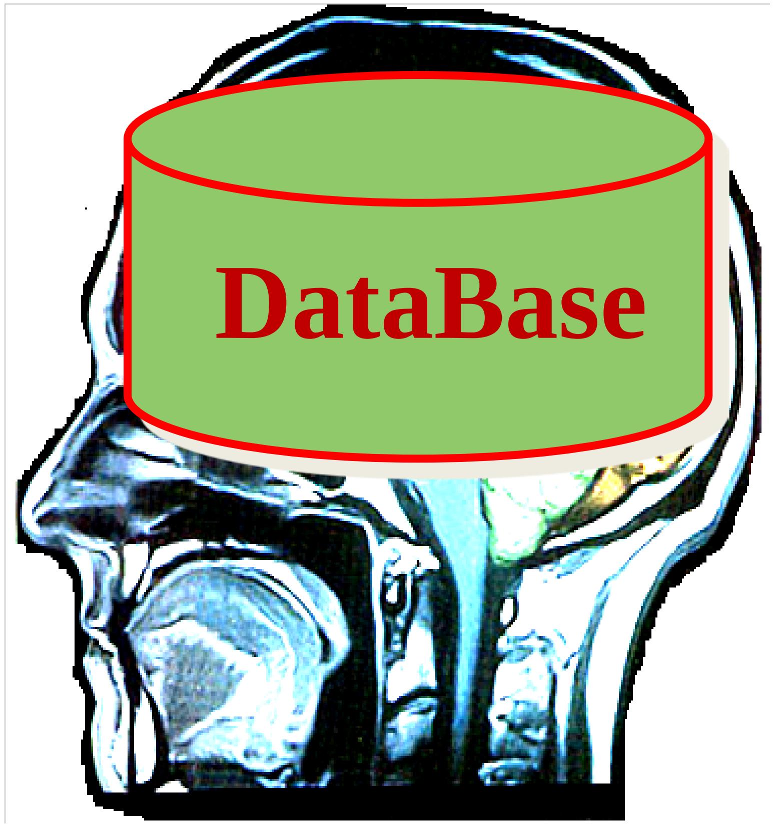
Module Design

- Module ---- *organ*
  - Function-independent
  - Integratable
- Rules for modularization
  - Information hiding
  - High cohesion
  - Low coupling



Database Design

- Database ---- ***brain***
  - **Store data**
  - **Process data**



Database Design

- **Database system ---- the DB server supplied by DB supplier**
  - **Big:** ***Oracle, DB2, Informix, Sybase***
  - **Medium:** ***SQL Server***
  - **Small:** ***mySQL(free), PostGre (free), access***

Database Design

- Design DB
  - Design the tables in DB
    - Data is stored in the tables
    - Table’s structure is data’s store structure
  - Design views, stored procedures
  - Operate / Process the data in DB
    - query, insert, update, delete
- Further course ---- *Database*

Data Structure & Algorithm Design

- Data Structure & Algorithm

  ---- ***nerves & muscles***

- Link all the components together to have them work as a whole system

Data Structure & Algorithm Design

- **Efficient program is based on good data structure and algorithms, rather than on small programming skills.**
- **Further course ----**
  - ***Data Structure***
  - ***Algorithm***

武生

花脸

刀马旦

小生

生

旦

净

丑

武生

青衣

武旦

旦

生

花旦

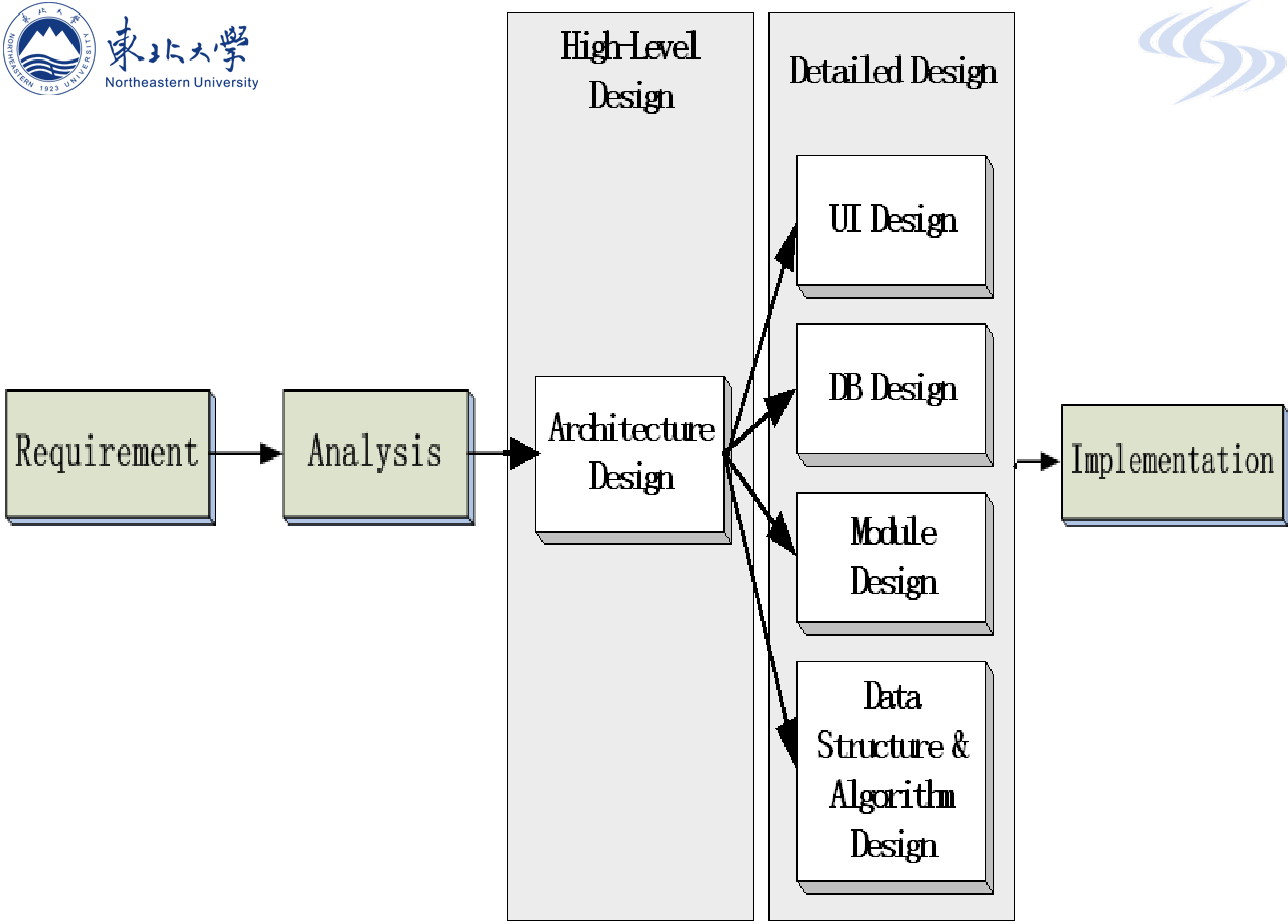
旦

旦

User Interface Design

- User Interface ---- *appearance*
- Criteria
  - Ease to use
  - Beauty
- Further course -----

User Interface Design

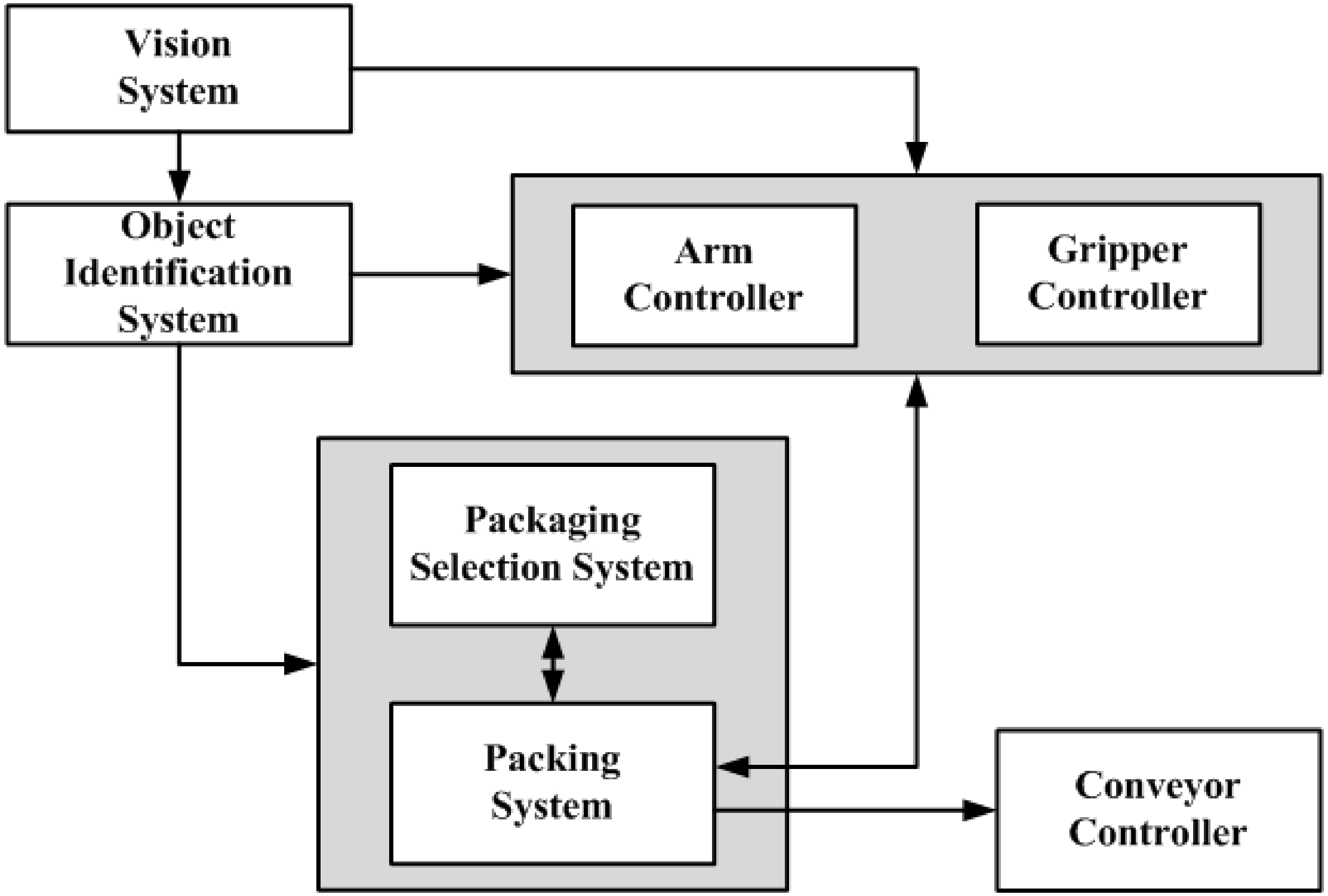

High-Level Design
Detailed Design
Requirement
Analysis
Architecture Design
UI Design
DB Design
Module Design
Data Structure & Algorithm Design
Implementation

Architecture Design

- Architecture is the first stage in the software design process;
- The architecture design of software is important;
- The output of the architecture design process is an architecture model that describes how the system is organized as a set of communicating components.

Architecture Design

- e.g., the architecture of a packing robot control system, showing the components that have to be developed.



Architecture Design

- Software architecture is important because it affects the performance, robustness, distributability, and maintainability of a system.
- Individual components implement the functional system requirements.
- The non-functional requirements depend on the system architecture ---- the way in which these components are organized and communicate.
- In many systems, non-functional requirements are also influenced by individual components, but there is no doubt that the architecture of the system is the dominant influence.

Architecture Design

- Software architecture design decisions include decisions on
  - the type of application,
  - the distribution of the system,
  - the architectural styles to be used,
  - the ways in which the architecture should be documented and evaluated.

Architecture Design Decisions

- Architecture design is a creative process where you design a system organization that will satisfy the functional and non-functional requirements of a system.
- Because it is a creative process, the activities within the process depend on the type of system being developed, the background and experience of the system architect, and the specific requirements for the system.

Architecture Design Decisions

- **During the architectural design process, system architects have to make a number of structural decisions that profoundly affect the system and its development process.**

Architecture Design Decisions

- Based on their knowledge and experience, the architects have to consider the following fundamental questions about the system:
  - Is there a generic application architecture that can act as a template for the system that is being designed?
  - How will the system be distributed across a number cores or processors?
  - What architectural patterns or styles might be used?

Architecture Design Decisions

- Based on their knowledge and experience, the architects have to consider the following fundamental questions about the system:
  - What will be fundamental approach used to structure the system?
  - How will the structural components in the system be decomposed into subcomponents?
  - What strategy will be used to control the operation of the components in the system?

Architecture Design Decisions

- Based on their knowledge and experience, the architects have to consider the following fundamental questions about the system:
    - What architectural organization is best for delivering the non-functional requirements of the system?
    - How will the architectural design be evaluated?
    - How should the architecture of the system be documented?

Architecture Design Decisions

- Although each software system is unique, systems in the same application domain often have similar architectures that reflect the fundamental concepts of the domain.
- Because of the close relationship between non-functional requirements and software architecture, the particular architectural style and structure that you choose for a system should depend on the non-functional system requirements.

Architecture Design Decisions

1. Performance

If performance is a critical requirement, the architecture should be designed to localize critical operations within a small number of components, with these components all deployed on the same computer rather than distributed across the network. This may mean using a few relatively large components rather than small, fine-grain components, which reduces the number of components communications.

Architecture Design Decisions

2. Security

If security is a critical requirement, a layered structure for the architecture should be used, with the most critical assets protected in the innermost layers, with a high level of security validation applied to these layers.

Architecture Design Decisions

3. Safety

If safety if a critical requirement, the architecture should be designed so that safety-related operations are all located in either a single component or in a small number of components. This reduces the costs and problems of safety validation and makes it possible to provide related protection systems that can safely shut down the system in the event of failure.

Architecture Design Decisions

4. Availability

If availability is a critical requirement, the architecture should be designed to include redundant components so that it is possible to replace and update components without stopping in the event of failure.

Architecture Design Decisions

5. Maintainability

If maintainability is a critical requirement, the system architecture should be designed using fine-grain, self-contained components that may readily be changed. Producers of data should be separated from consumers and shared data structures should be avoided.

Architecture Views

- **It is impossible to represent all relevant information about a system's architecture in a single architectural model, as each model only shows one view or perspective of the system.**
- **Krutchen (1995), in his well-known 4+1 view model of software architecture, suggests that there should be 4 fundamental architectural views:**

Architecture Views

1. A logical view, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.
2. A process view, which shows how, at run-time, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.

Architecture Views

3. A development view, which shows how the software is decomposed for development, that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful of software managers and programmers.
4. A physical view, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

Architecture Patterns

- The architectural patterns, well-tried ways of organizing system architectures, which can be reused in system designs
- The idea of patterns as a way of presenting, sharing, and reusing knowledge about software systems is now widely used.
- An architecture pattern should describe a system organization that has been successful in previous systems.

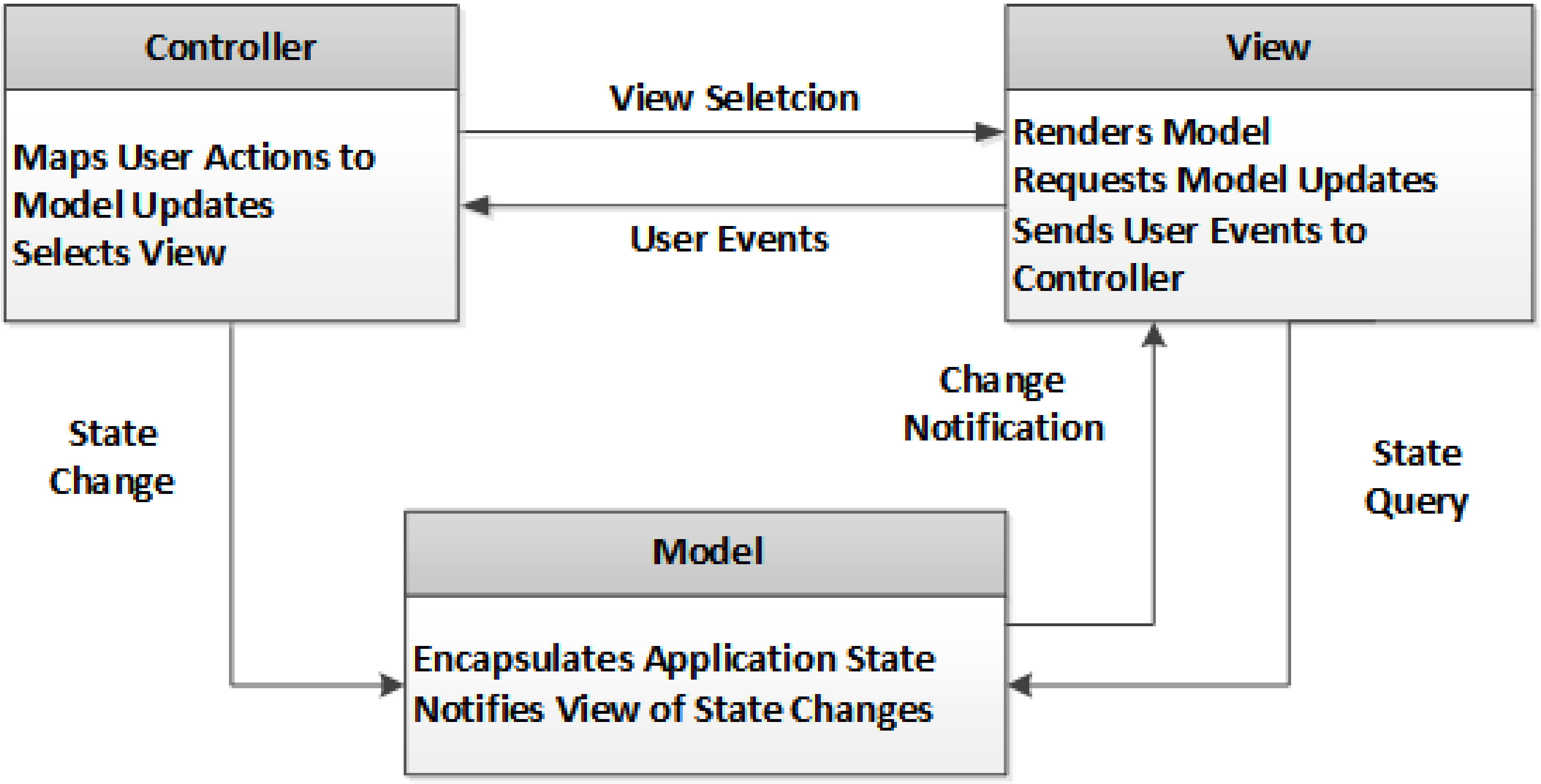
Architecture Patterns

- **Model-View-Controller pattern**

  The well-known Model-View-Controller pattern, the basis of interaction management in many web-based systems. The MVC pattern separates elements of a system, allowing them to change independently.

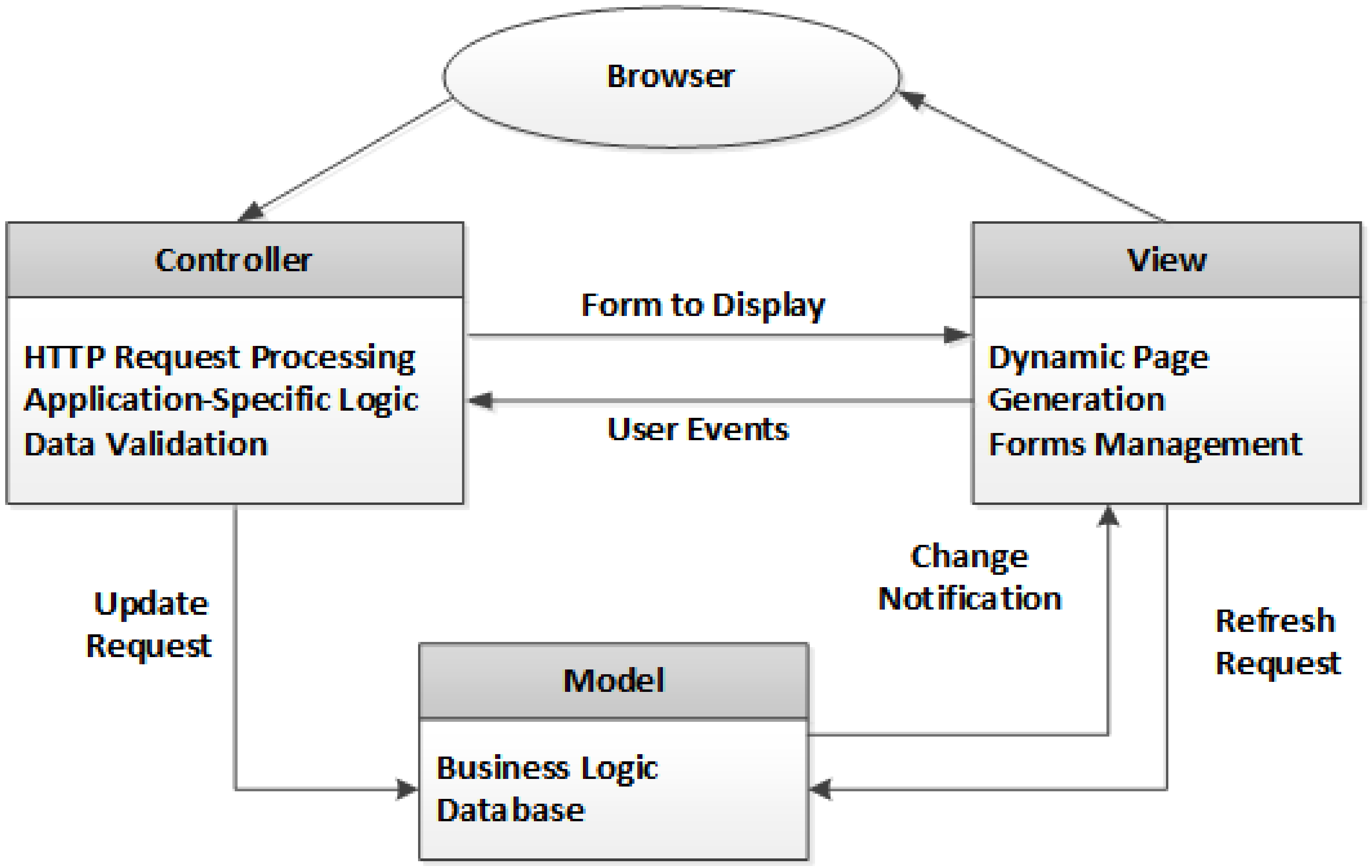
Architecture Patterns

- Model-View-Controller pattern



Architecture Patterns

- **Web application architecture using the MVC pattern**



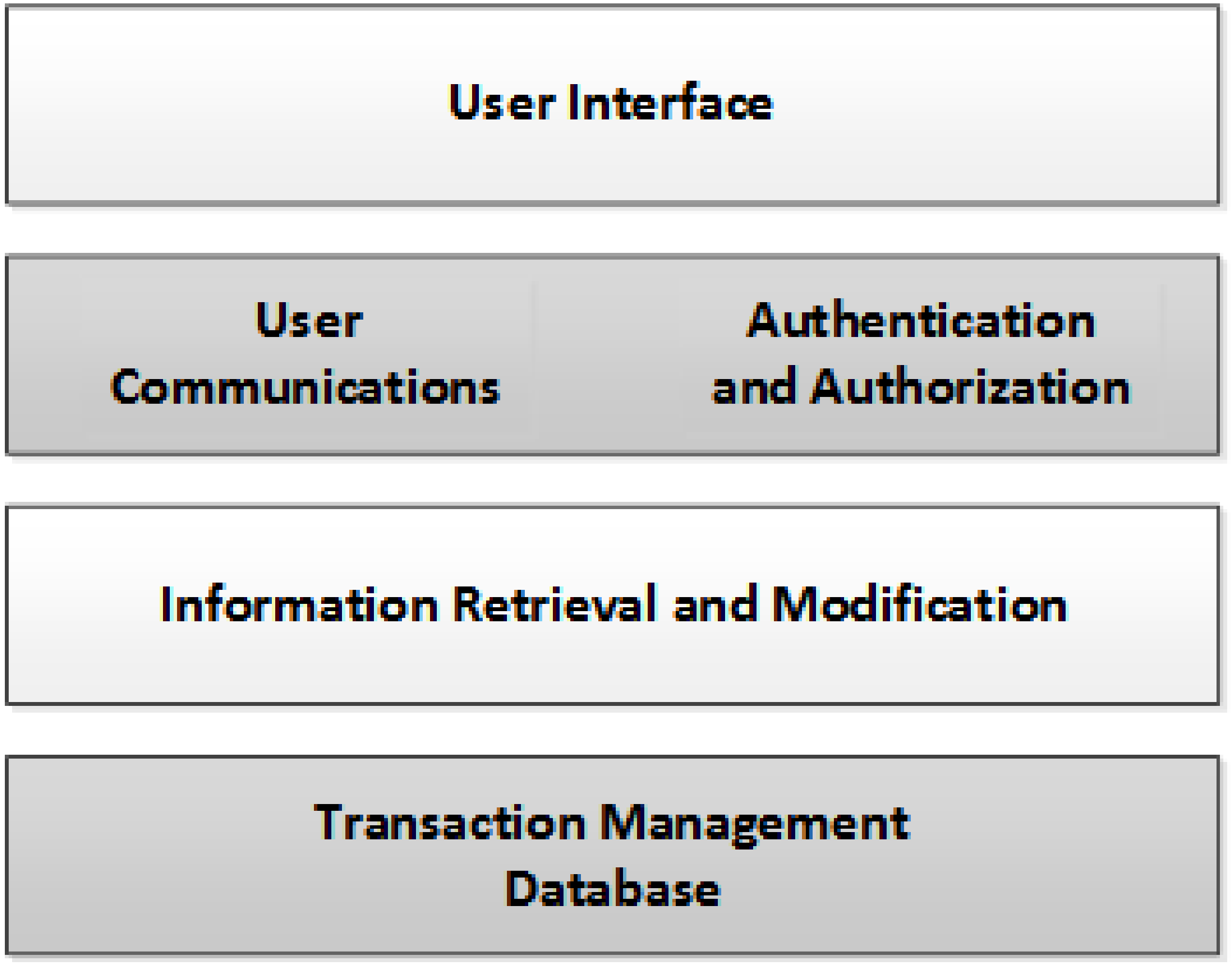
Architecture Patterns

- Layered architecture
  - Layered architecture supports the incremental development of systems.
  - The system functionality is organized into separate layers, and each layer only relies on the facilities and services offered by the layer immediately beneath it.

Architecture Patterns

- Layered architecture

e.g., layered information system architecture


User Interface
User Communications
Authentication and Authorization
Information Retrieval and Modification
Transaction Management Database

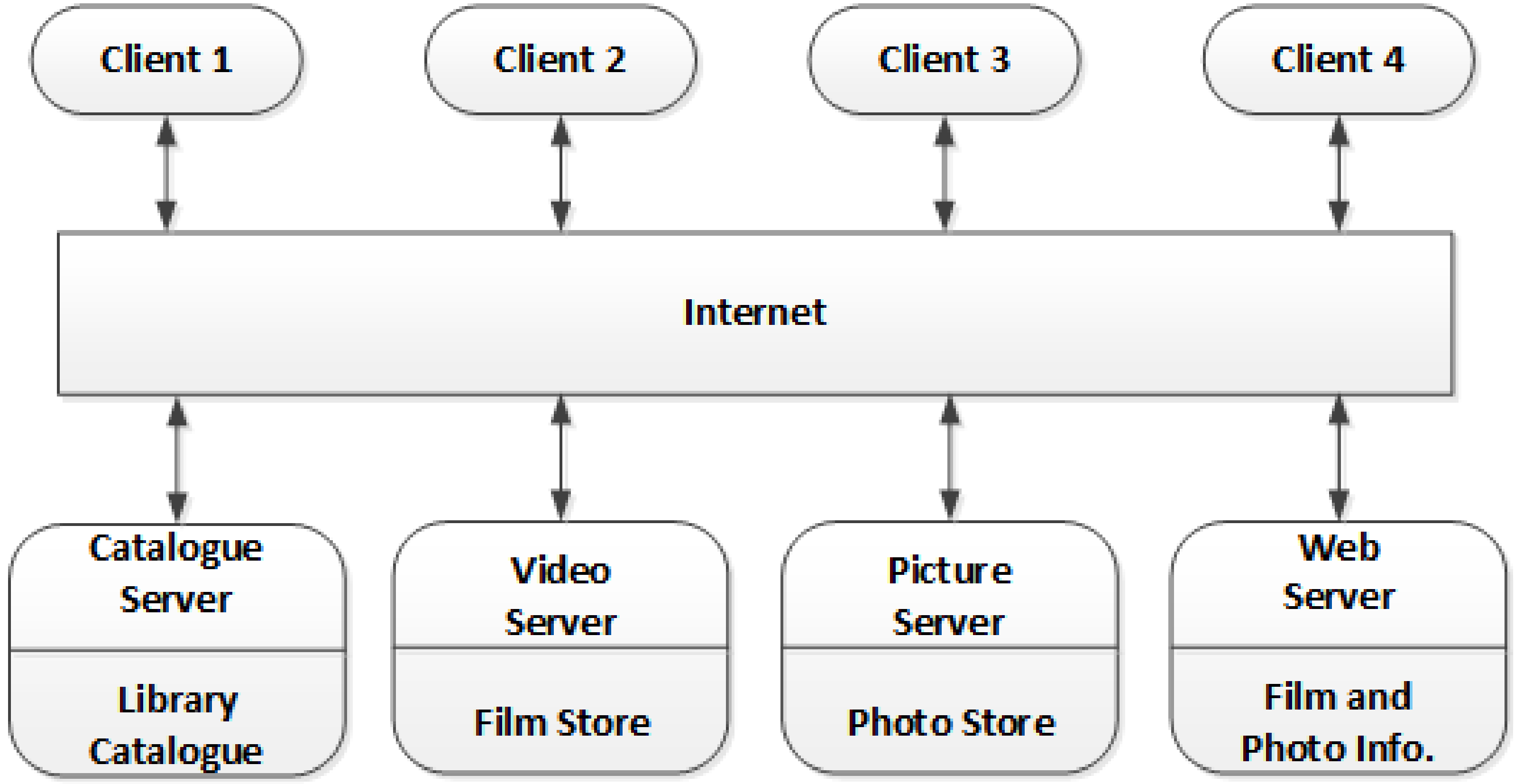
Architecture Patterns

- **Client-Server Architecture**
  - **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.**
  - **Client-server architectures are usually thought of as distributed systems architectures but the logical model of independent services running on separate servers can be implemented on a single computer.**

Architecture Patterns

- **Client-Server Architecture**

  e.g., a client-server architecture for a film library



Architecture Patterns

- **Client-Server Architecture**

  The major components of this model are:

  - **A set of servers that offer services to other components. Examples of servers include print servers that offer printing services, file servers that offer file management services, and a compile server, which offers programming language compilation services.**

Architecture Patterns

- **Client-Server Architecture**

  The major components of this model are:

  - **A set of clients that call on the services offered by servers. There will normally be several instances of a client program executing concurrently on different computers.**
  - **A network that allows the clients to access these services. Most client-server systems are implemented as distributed systems, connected using Internet protocols.**

Application Architecture

- Application systems are intended to meet a business or organizational need.
- Consequently, the application systems used by these businesses also have much in common.
- The application architecture may be re-implemented when developing new systems. But for many business systems, application reuse is possible without re-implementation.

Application Architecture

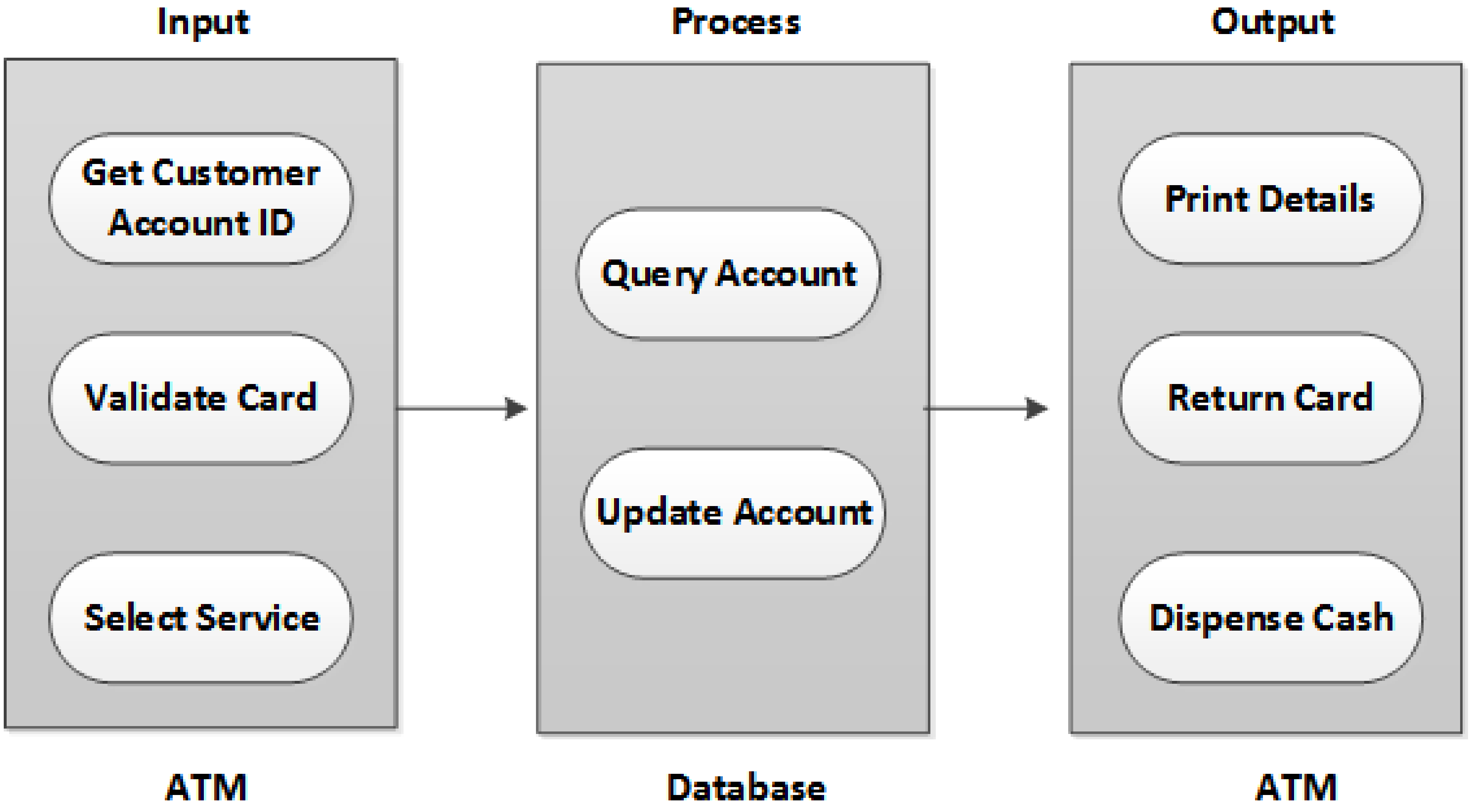
- We see this in the growth of Enterprise Resource Planning (ERP) systems from companies such as SAP and Oracle. In these systems, a generic system is configured and adapted to create a specific business application. E.g., a system for supply chain management can be adapted for different types of suppliers, goods, and contractual arrangements.
- There are many types of application system, and in some cases, they may seem to be very different.

Application Architecture

- However, many of these superficially dissimilar applications actually have much in common, and thus can be represented by a single abstract application architecture.
- Let’s discuss the architecture of Transaction Processing Applications.

Application Architecture

- E.g., the software architecture of an ATM system



To be continued with

6.2 Object-Oriented Design