UNIT-3(ARCHITECTURAL DESIGN) COVERS

- Architectural design decisions
- System organization
- Modular decomposition styles
- Control styles
- Reference architecture
- Multiprocessor architecture
- Client-server architectures
- Inter-organizational distributed computing

ARCHITECTURAL DESIGN DECISIONS

- Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system.
- In the model of the software development process architectural design is the first stage in software design process.
- It is the critical link between design and requirement engineering as it identifies the main structural components in a system and relationship between them.
- The output of the architectural design process an architectural model that describes how the system is organized as a set of communicating component.

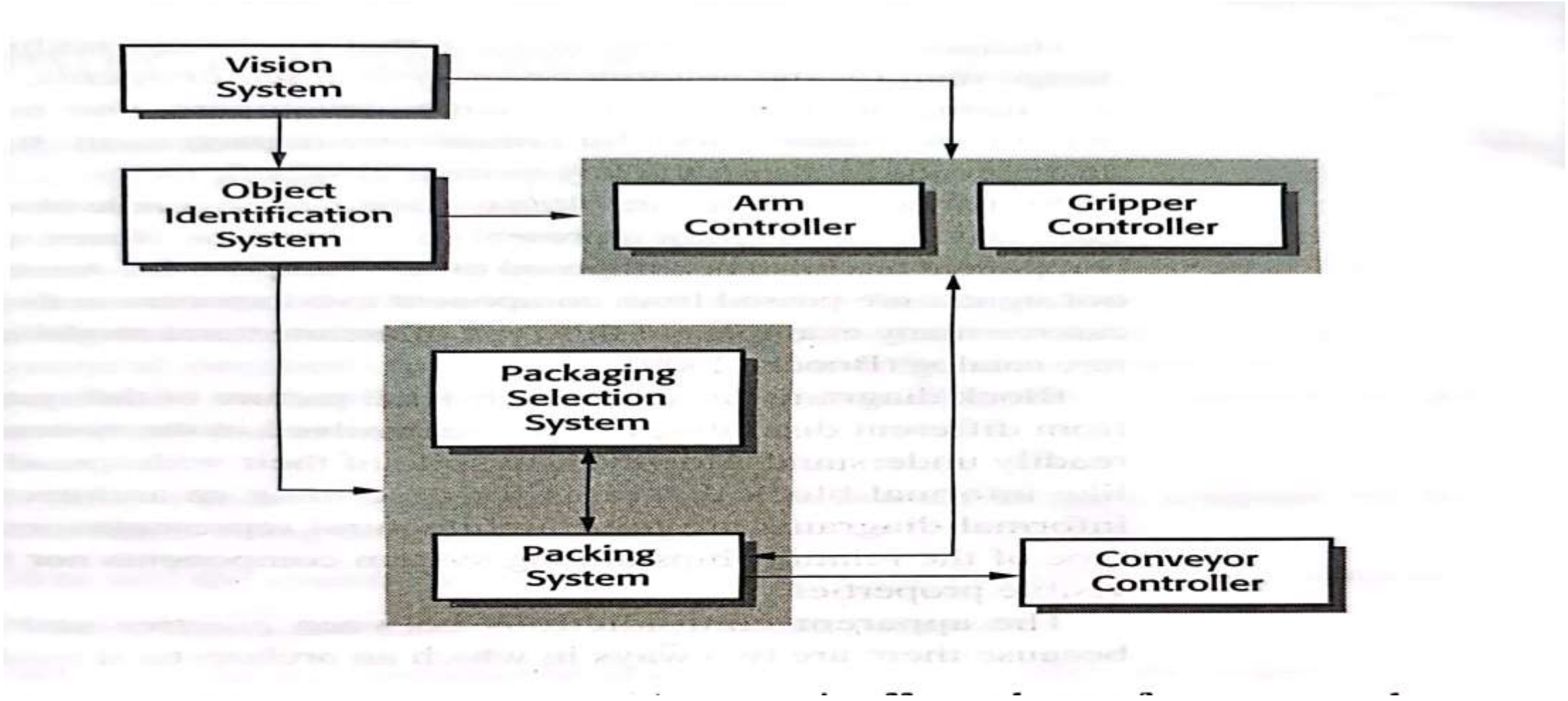
ARCHITECTURAL DESIGN DECISIONS

- Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system.
- In the model of the software development process architectural design is the first stage in software design process.
- It is the critical link between design and requirement engineering as it identifies the main structural components in a system and relationship between them.
- The output of the architectural design process an architectural model that describes how the system is organized as a set of communicating component.

ARCHITECTURAL DESIGN DECISIONS

- To better understand let us consider an example of packing robot system that shows an abstract model of the architecture .
- This robotic system can pack different kinds of object.
- It uses a vision component to pick out objects on a conveyor, identify the types of object, and select the right kind of packaging.
- The system then moves objects from the delivery conveyor to be packaged.
- It places packaged objects on another conveyor.
- The architectural model shows these components and the links between them.

ARCHITECTURAL DESIGN DECISIONS



ARCHITECTURAL DESIGN DECISIONS

- You can design software architectures at two levels of abstraction which I call architecture in the small and architecure in the large.
- Architecture in the small is concerned with the architecture of individual programs.
- At this level, we are concerned with the way that an individual program is decomposed into components.
- Architecture in the large is concerned with the architecture of complex enterprise systems that include other systems,programs, and program components.

ARCHITECTURAL DESIGN DECISIONS

Advantages of explicitly designing and documenting software architecture:

1. **Stakeholder communication:**

The architecture is a high level presentation of the system that may be used as a focus for discussion by a range of different stakeholders.

2. System analysis:

Making the system architecture explicit at an early stage in the system development requires some analysis. Architectural design decisions have a profound effect on whether or not the system can meet critical requirement such as performance , reliability and maintainability

ARCHITECTURAL DESIGN DECISIONS

3. Large –scale reuse:

A model of a system architecture is a compact, manageable description of how a system is organized and how the components interoperate.

ARCHITECTURAL DESIGN DECISIONS

- Architectural design is a creative process where you design a system organization that will satisfy the functional and non-functional requirements of a system.
- Therefore it is useful to think of architectural design a series of decisions to be made rather than a sequence of activities.
- Based on their knowledge and experience, they have to consider the following fundamental questions about the system:

1. Is there a generic application architecture that can act as a template for the system that is being designed?
2. How will the system be distributed across a number of cores ofs processors?

ARCHITECTURAL DESIGN DECISIONS

3. What architectural patterns or styles might be used?

4. What will be the fundamental approach used to structure the system?

5. How will the structural components in the system be decomposed into sub components?

6. What strategy will be used to control the operation of the components in the system?

7. What architectural organization is best for delivering the non-functional requirement of the system?

8. How will the architectural design be evaluated?

9. How should the architecture of the system be documented?

System organization

- Reflects the basic strategy that is used to structure a system.
- Three organisational styles are widely used:
- A shared data repository style;
- A client/server style;
- An abstract machine or layered style.

SYSTEM ORGANIZATION

1. **Shared repository style/architecture:**

All 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.

Example: An example of IDE where the components use a repository of system design information. Each software tool generates information which is then available for use by other tools.

When used: you should use this pattern when you have a system in which large volumes of information are generated that has to be stored for a long time.

You may also used in data-driven systems where the inclusion of data in the repository triggers an action or tools.

SYSTEM ORGANIZATION

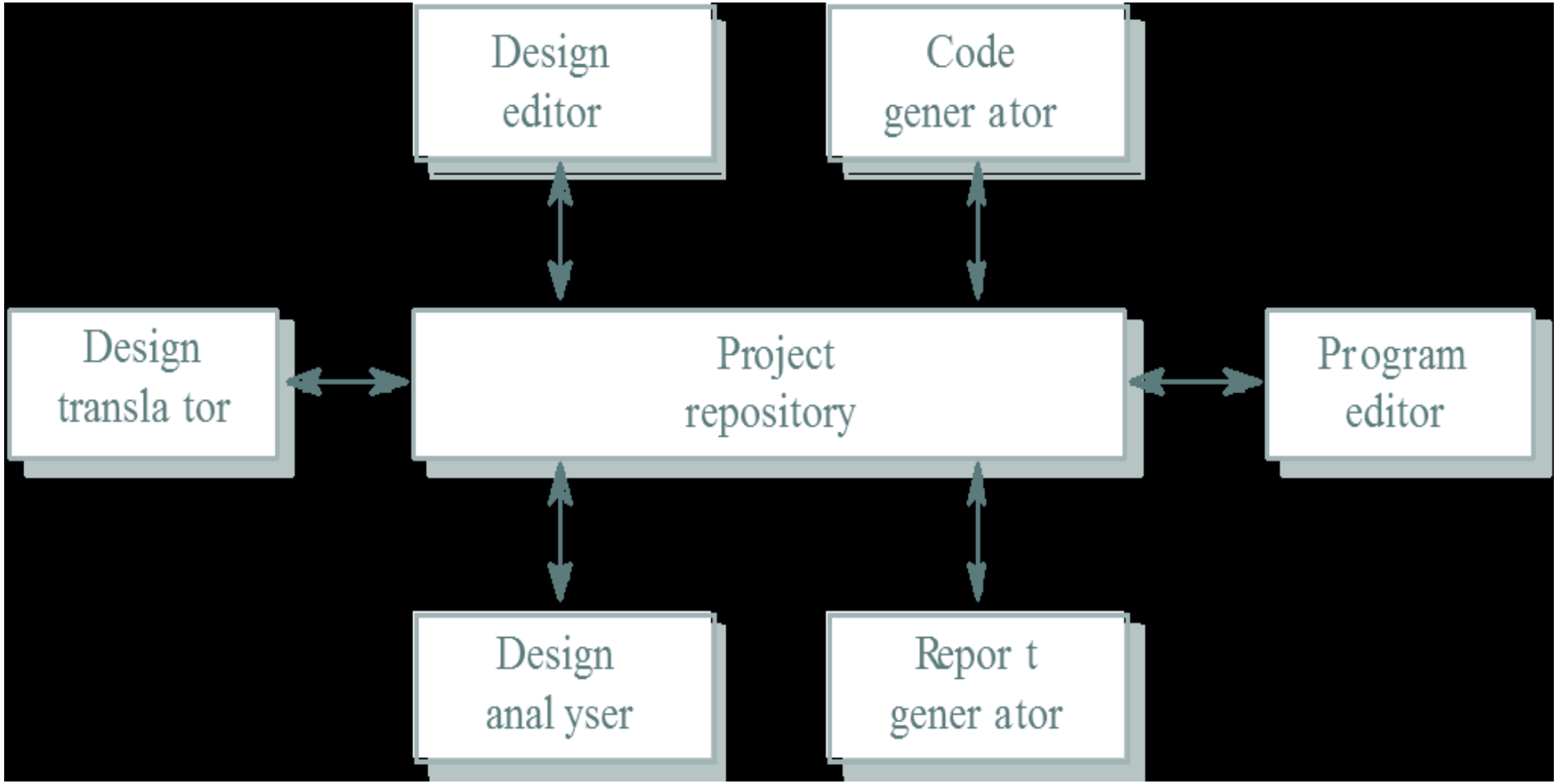
Advantages:

- Components can be independent-they donot need to know of the existence of other components.
- Changes made by one component can be propagated to all components
- All data can be managed consistently(i.e backup done at the same time) as it is all in one place.

Disadvantages:

- The repository is a single point of failure so problems in the repository affect the whole system.
- May be inefficiencies in organizing all communication through repository.
- Distributing the repository across several computers may be difficult.

System organization



SYSTEM ORGANIZATION

2. client/server style:

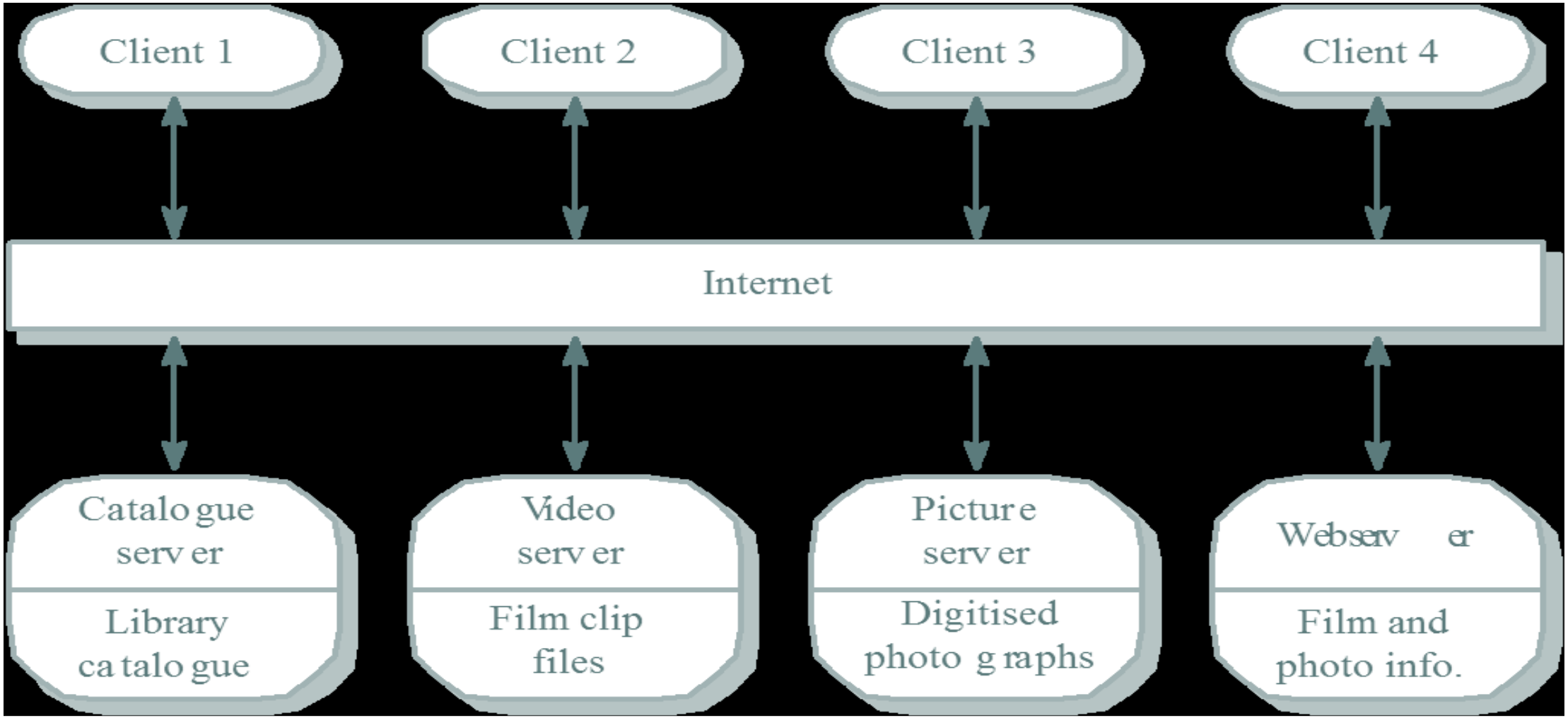
- The system that follows the client-server pattern is organized as a set of services and associated servers and clients that access and use the services.

The major component of this model are:

- A set of servers that offer the services to other components. Example print servers that offer printing services, file server that offer file management services.
- A set of clients that call on the services offered by servers.
- A network that allows the clients to access these services

When used: used when data in a shared database has to be accessed from a range of locations.

System organization



System organization

Advantages

- Distribution of data is straightforward;
- Makes effective use of networked systems.
- May require cheaper hardware;
- Easy to add new servers or upgrade existing servers.

Disadvantages

- No shared data model so sub-systems use different data organisation. Data interchange may be inefficient;
- Redundant management in each server;
- No central register of names and services -it may be hard to find out what servers and services are available.

SYSTEM ORGANIZATION

3**. Abstract machine/layered style:**

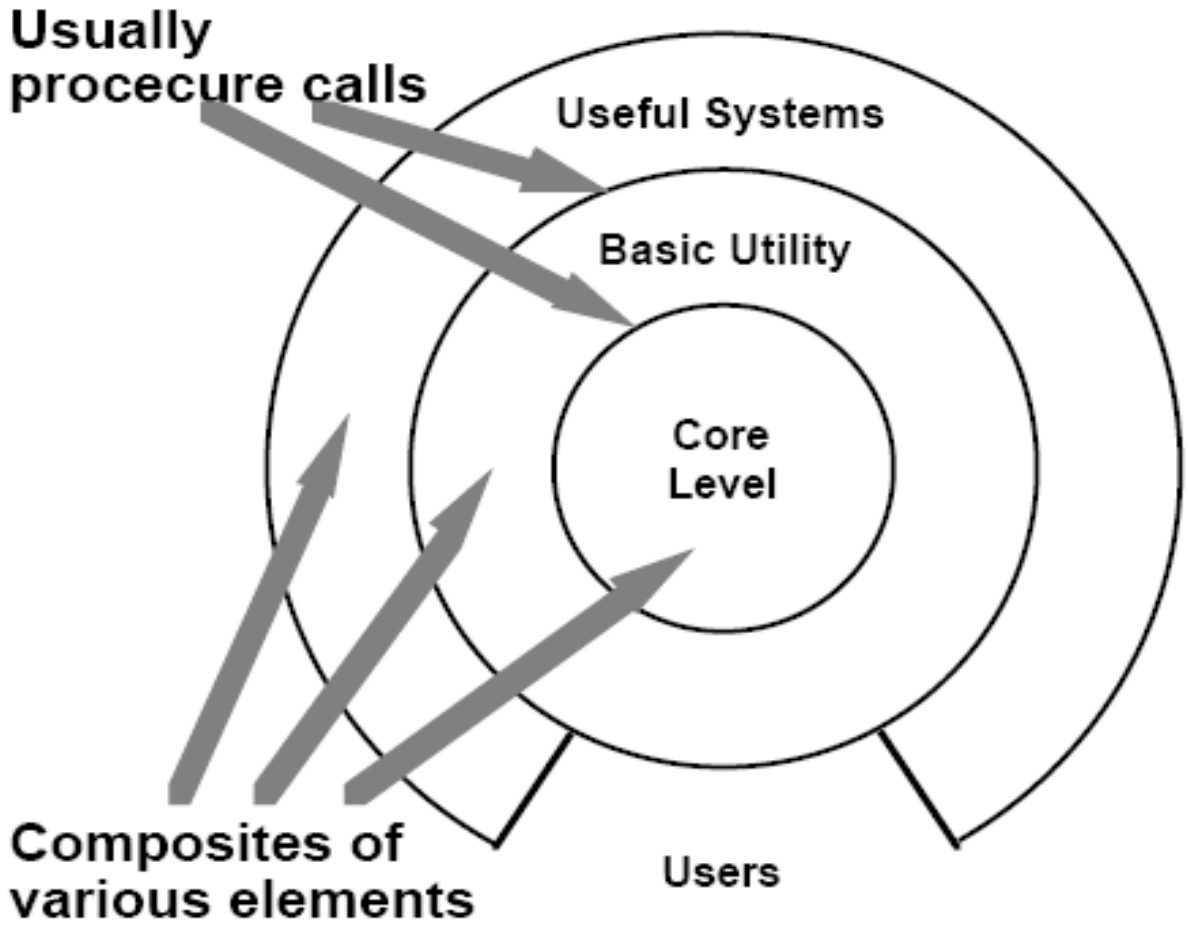
- Used to model the interfacing of sub-systems.
- Organises the system into a set of layers (or abstract machines)
- Each layer provides a set of services to the layer above and serves as a client to the layer below.
- Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.

When used: used when building new facilities on top of existing systems

When the development is spread across several teams with each team responsibility for a layer of functionality

When there is requirement for multi-level security.

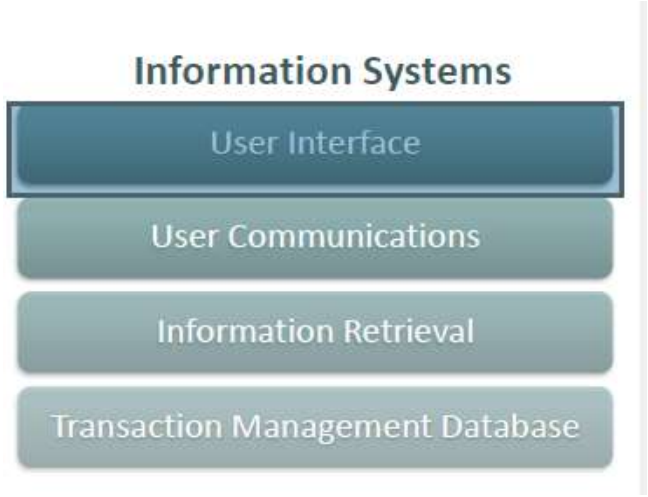
System Organization



System Organization

- Communication protocols (OSI reference model for network protocols).
- Operating systems
- Database Management systems (storage, transaction, query, optimizer, etc)

- **Information Systems**



System Organization

Advantages

- Supports incremental development
- Changeable (if a layer changes, only adjacent layers are affected)

Disadvantages

- Structuring systems into layers is difficult
- Inner layers may provides facilities required by all layers (e.g. file management)
- Performance is degraded.

MODULAR DECOMPOSITION STYLES

- The architectural design process where the sub-system are decomposed into modules is known as modular decomposition architecture.
- Two modular decomposition models can be covered as:

MODULAR DECOMPOSITION STYLES

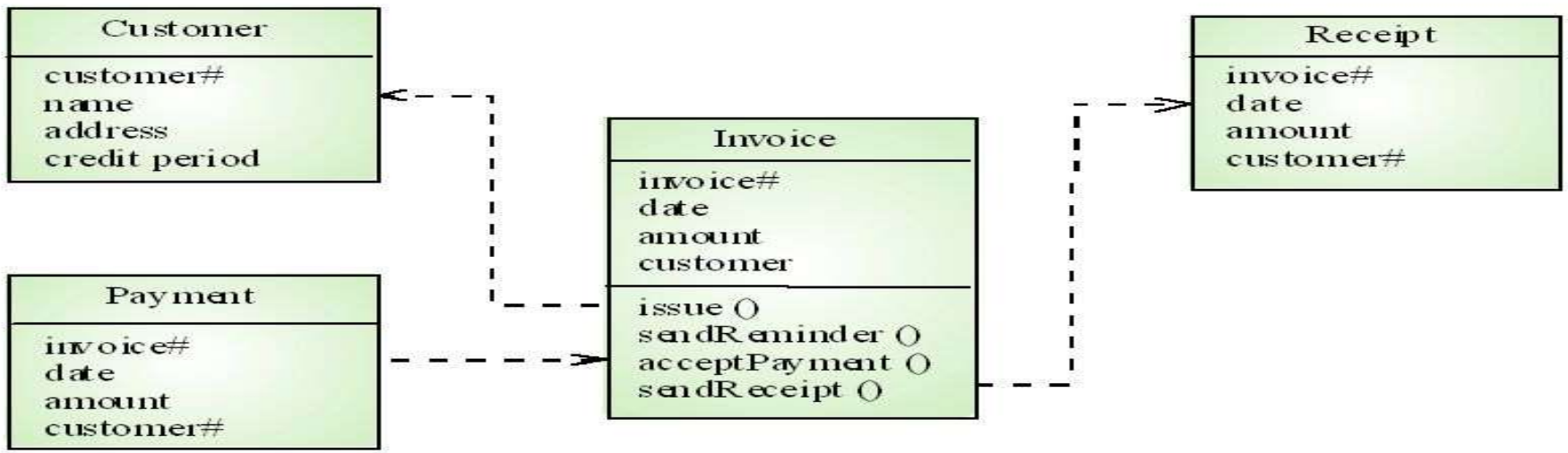
1. Object model:

- The model where the system is decomposed into interacting objects with well-defined interfaces.
- Object-oriented decomposition is concerned with identifying object classes, their attributes and operations.
- When implemented, objects are created from these classes and some control model used to coordinate object operations.

MODULAR DECOMPOSITION STYLES

The example of object model can be shown as:

Example of simple object model:
Invoice processing system



DATA-FLOW MODEL

- A data-flow model is the model where the system is decomposed into functional modules which transform inputs to outputs.
- Also known as the pipeline model
- Functional transformations process their inputs to produce output.
- May be referred to as a pipe and filter model(as inUNIX shell).
- When transformation are sequential, this is a batch sequential model which is extensively used in data processing system.
- Not really suitable for interactive systems.

DATA-FLOW MODEL

The example of data-flow model can be shown as:

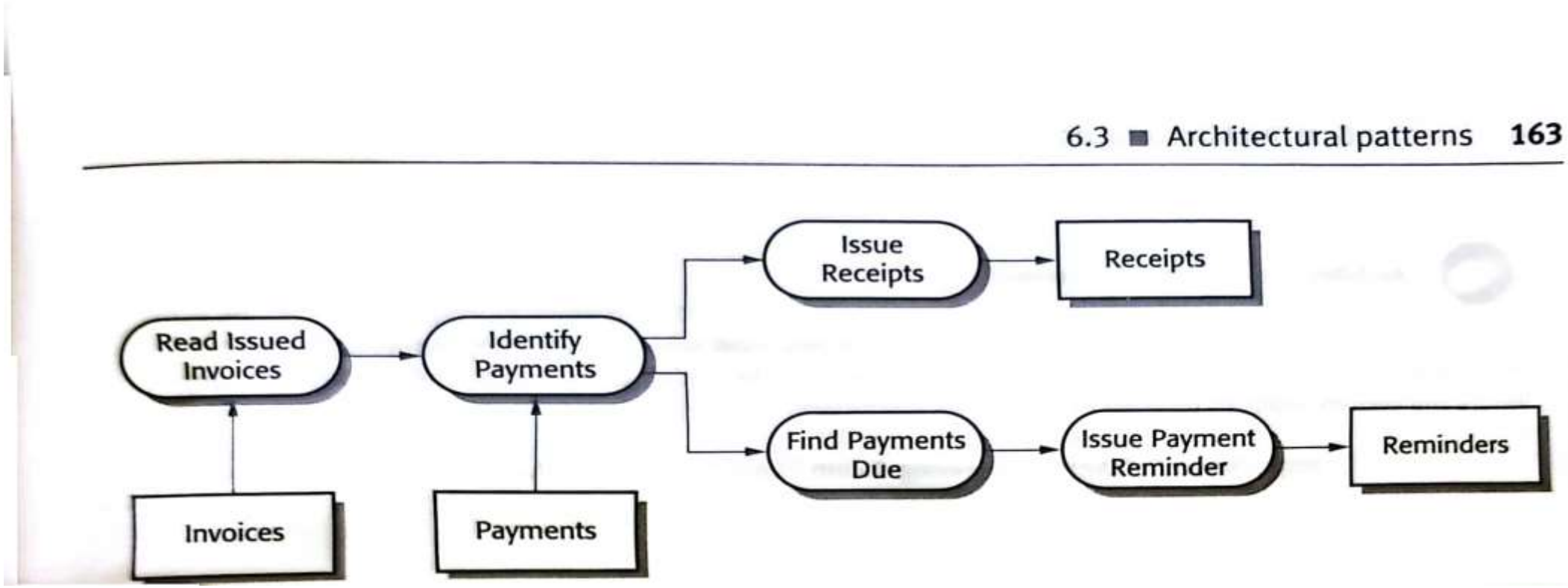


Fig: Invoice Processing system as data-flow model

CONTROL STYLES

The modeling technique which is concerned with the control flow between sub-system is called as control modeling.

Control styles are classified as:

1. Cetralized control
2. Event-based control

CONTROL STYLES/MODELLING

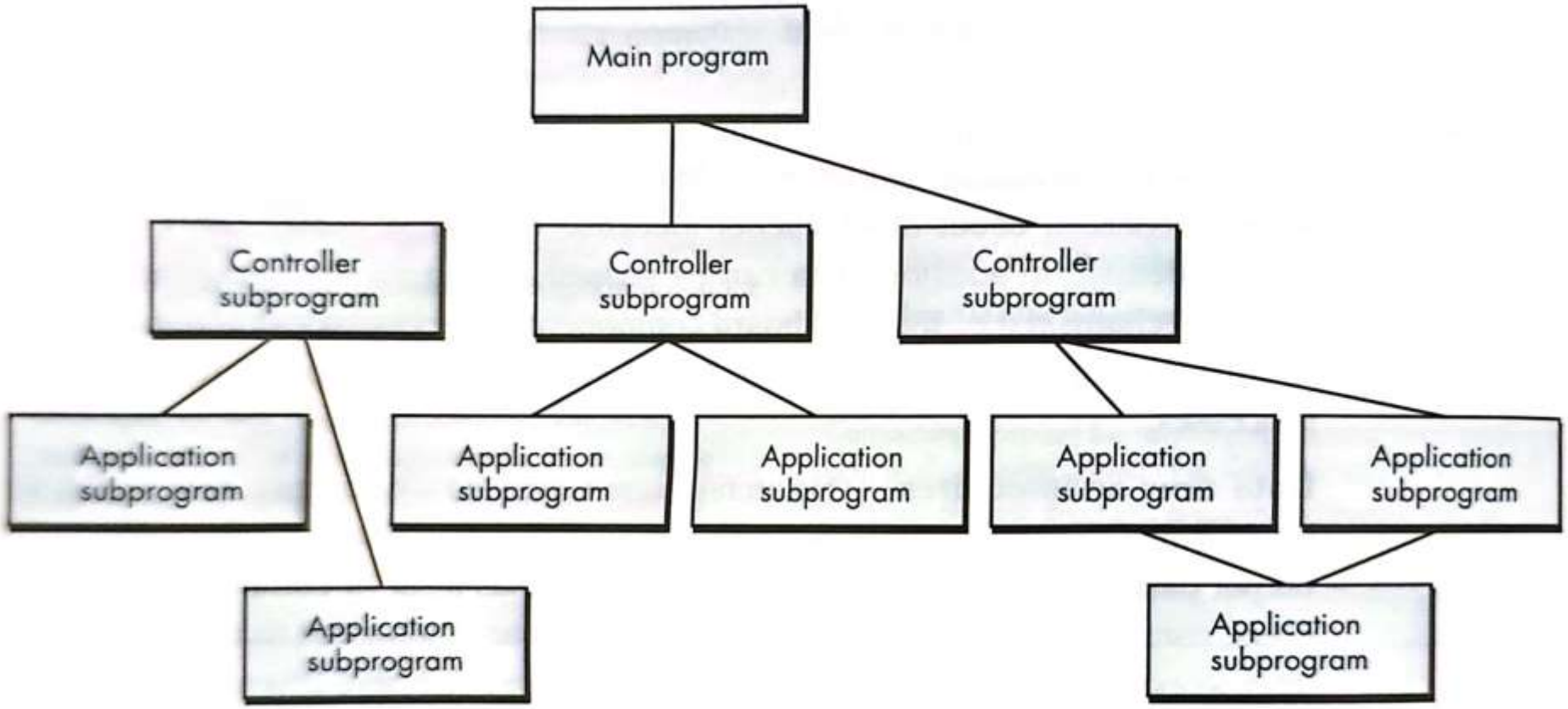
1. Centralized control:

- One sub-system has overall responsibility for control and starts and stops other sub-system
- A control sub-system takes responsibility for managing the execution of other sub-systems.

a. call-return model:

- Top-down sub-routine model where control starts at the top of a subroutine hierarchy and moves downwards.
- It is applicable to sequential systems.

CONTROL STYLES/MODELLING

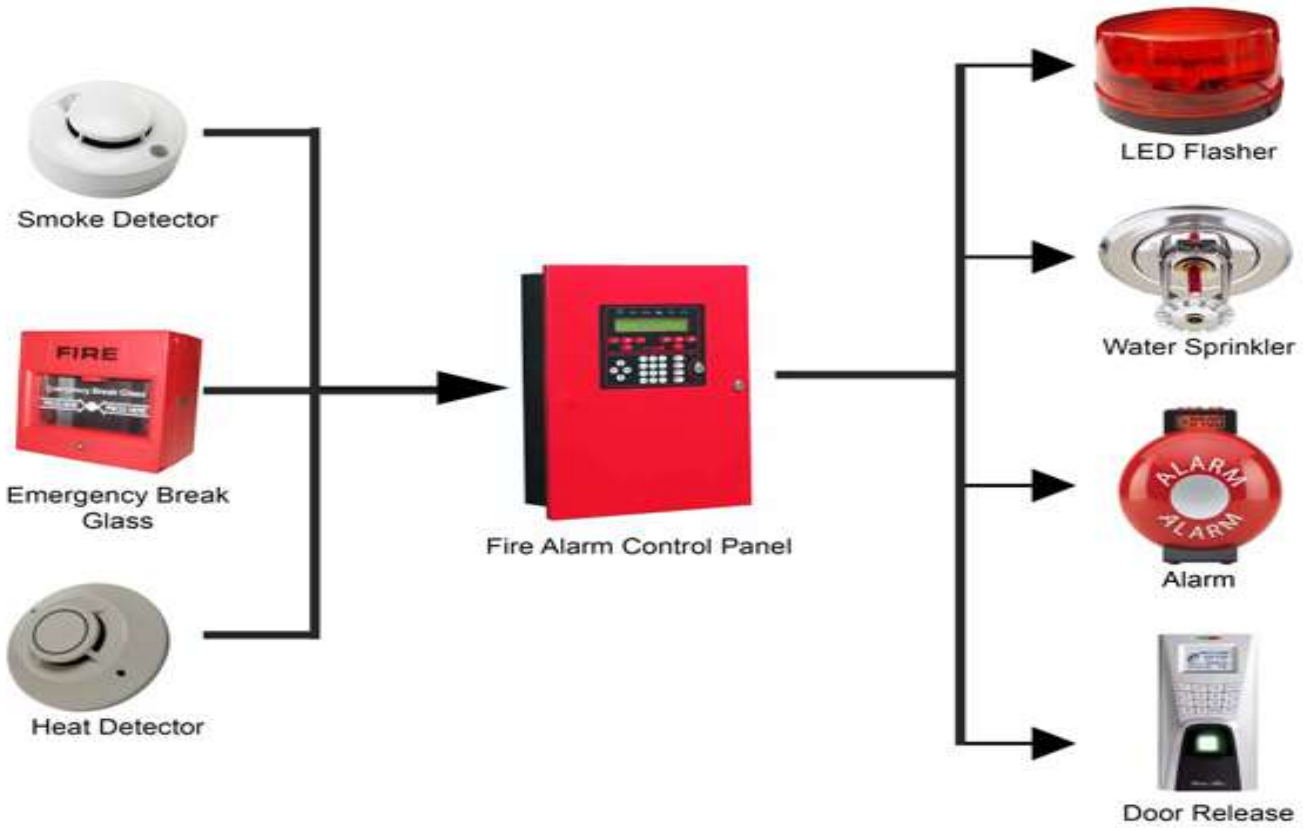


CONTROL STYLES

b. Manager model

- It is applicable to concurrent systems.
- One system component controls the stopping, starting and coordination of other system processes.

CONTROL STYLES



CONTROL STYLES

2. **Event-Based Control**

- Each sub-system can respond to externally generated events from other sub-systems or the systems environment.
- Driven by externally generated events where the timing of the events is out of the control of the sub-systems which process the event.

CONTROL STYLES

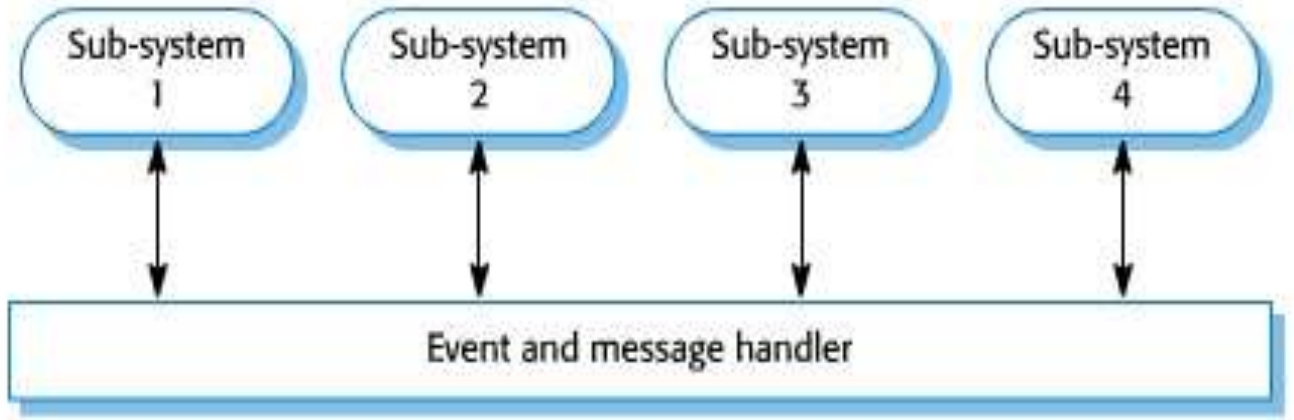
a. Broadcast models:

- An event is broadcast to all sub-systems.
- Any sub-system which can handle the event may do so.

CONTROL STYLES

a. Broadcast models:

- An event is broadcast to all sub-systems.
- Any sub-system which can handle the event may do so.

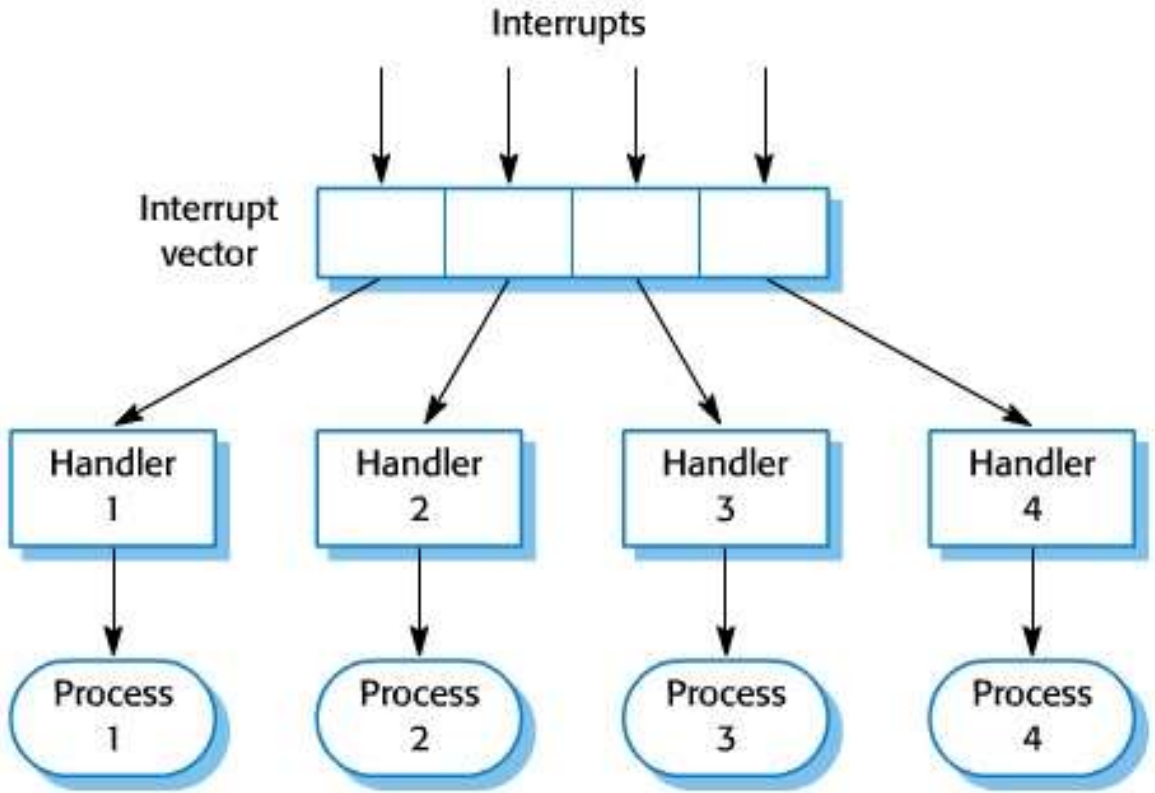


CONTROL STYLES

b. interrupt-driven models:

- It is used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing.
- The event driven models include spreadsheets and production systems.

CONTROL STYLES



REFERENCE ARCHITECTURE

- The model used as a basis for system implementation or to compare different systems is known as reference architecture.
- Derived from a study of the application domain rather than from existing system.
- It acts as a standard against which systems can be evaluated.
- OSI model is a layered model for communication systems.

REFERENCE ARCHITECTURE

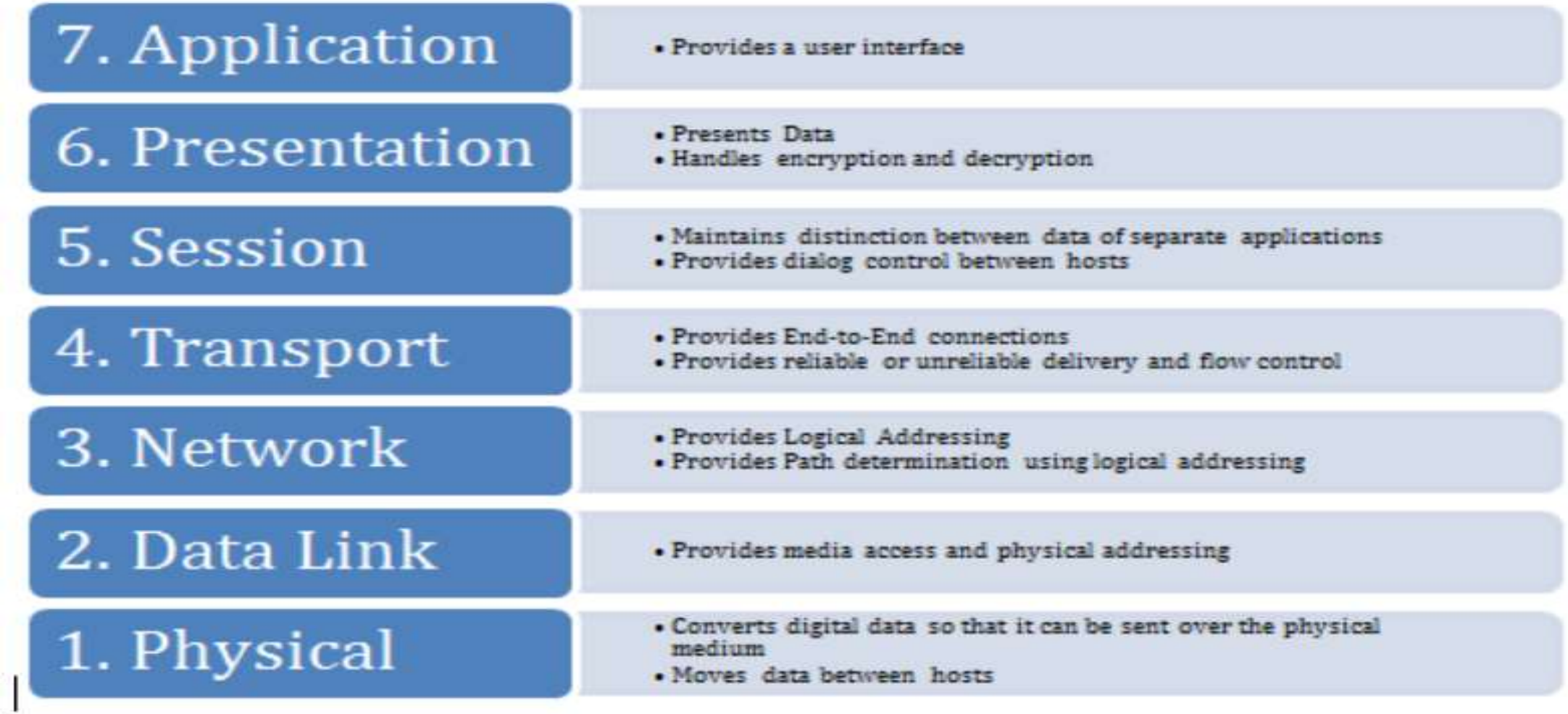


Fig: OSI reference model

MULTIPROCESSOR ARCHITECTURE

- The simplest distributed system model in which the system is composed of multiple processes which may execute on different processors is called as multiprocessor architecture.
- Architectural model of many large real-time system.
- Distribution of process to processor may be pre-ordered or may be under the control of a dispatcher.

MULTIPROCESSOR ARCHITECTURE

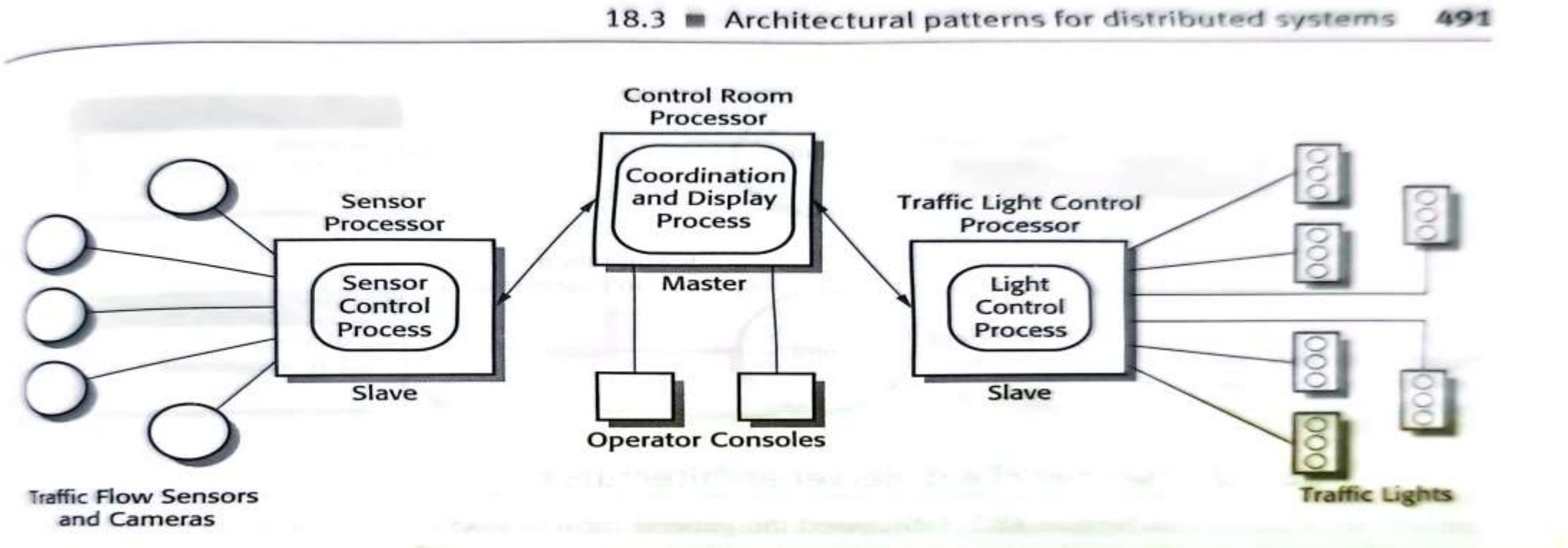
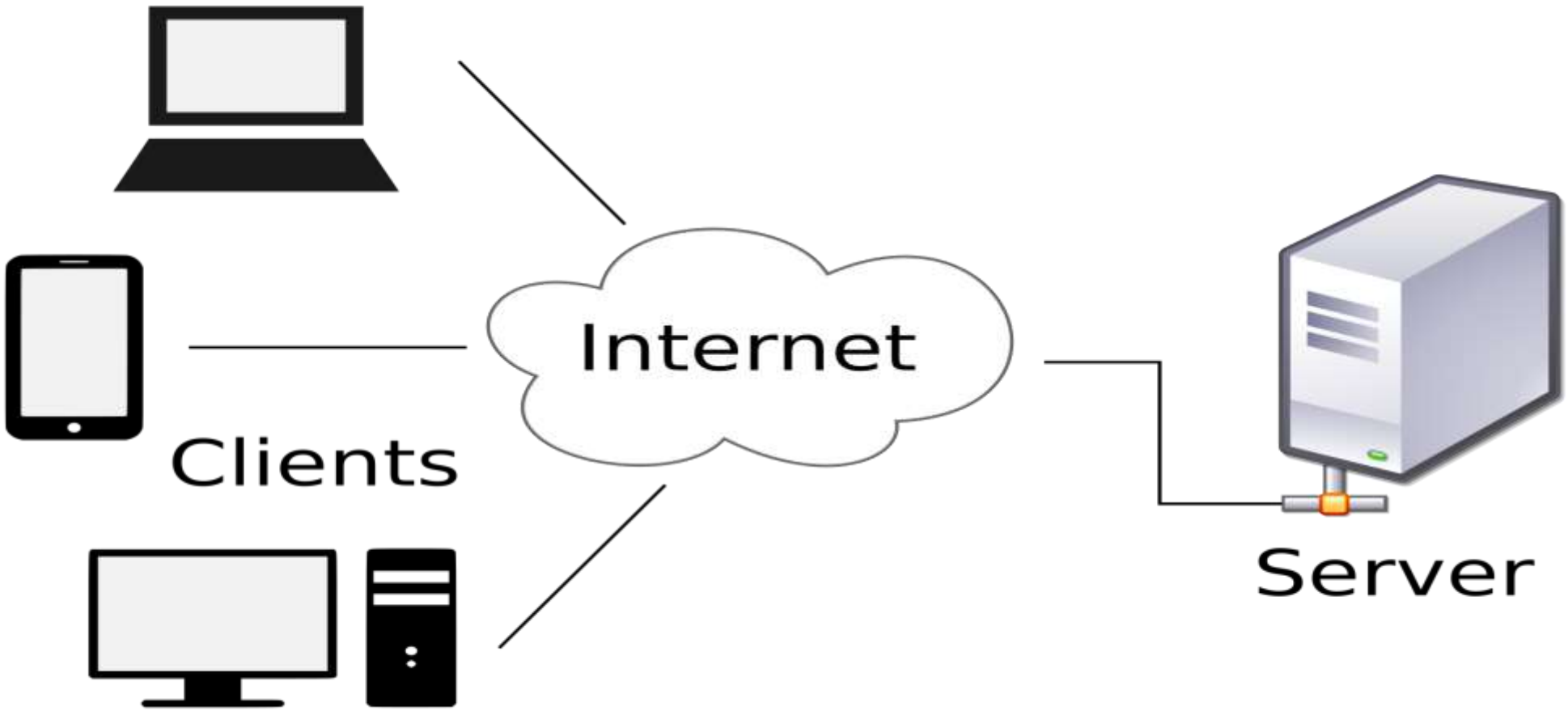


Fig: traffic flow sensors and cameras

CLIENT SERVER ARCHITECTURE

- Client-server architecture is a computing model in which the server hosts, delivers and managers most of the resources and services to be consumed by the clients.
- Consists of one or more client computers connected to a central server over a network or internet connection
- System shares computing resources.
- Referred to as a networking computing model because all the requests and services are delivered over a network
- Servers are powerful computers or processes dedicated to managing disk drivers, printers or network traffic.
- Clients are PCs or workstations on which users run applications.

CLIENT SERVER ARCHITECTURE



CLIENT SERVER ARCHITECTURE

- Layers in a client-server system

Presentation:

Concerned with presenting information to the user and managing all user interface.

Data hiding:

Manages the data that is passed to and from the client, implement checks on the data, generate web pages etc.

CLIENT SERVER ARCHITECTURE

- **Application processing layer:**

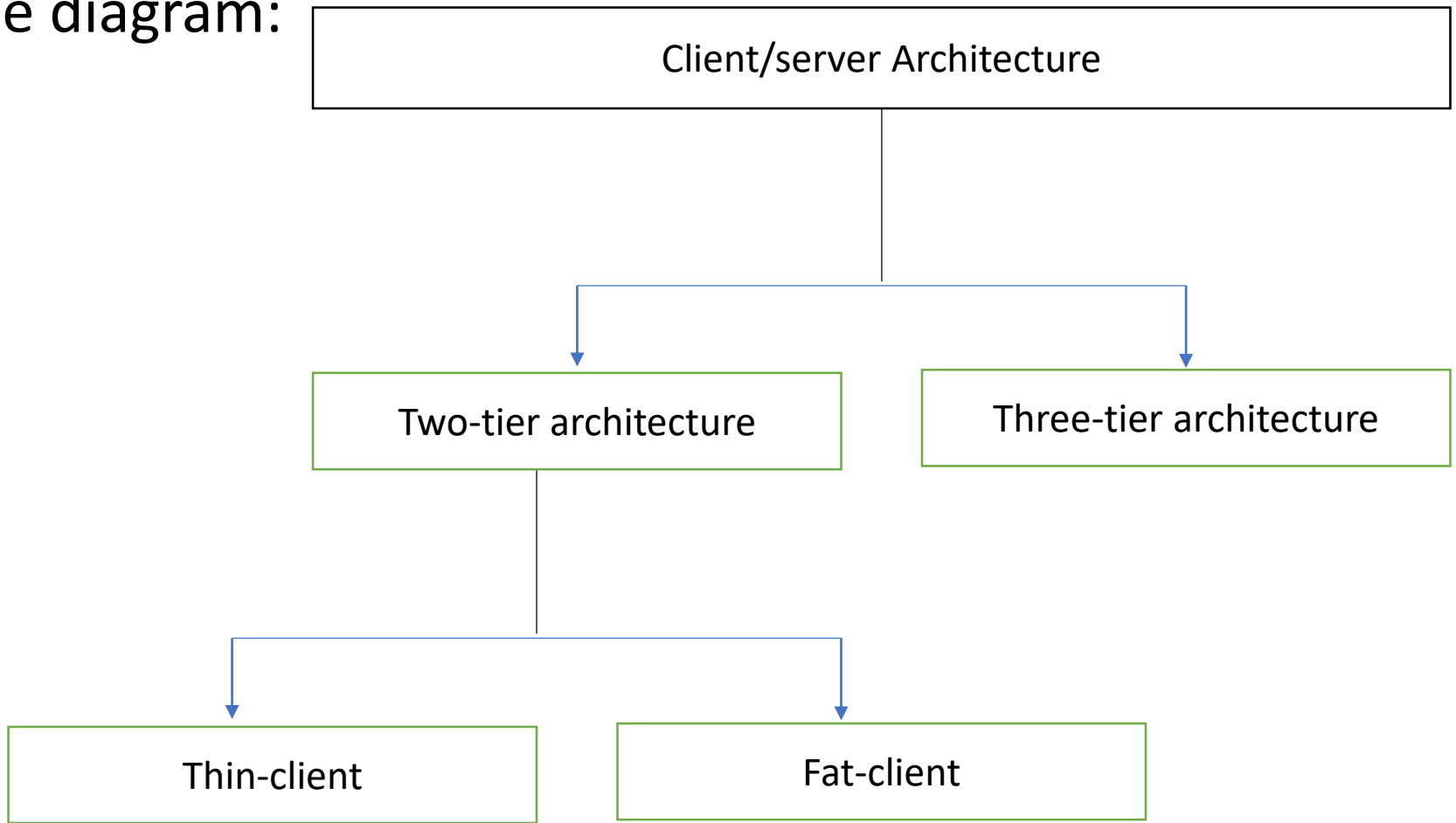
Concerned with implementing the logic of the application and so providing the required functionality to end users.

Database:

Stores data and provides transaction management services, etc.

CLIENT SERVER ARCHITECTURE

Moreover the client-server architecture can be classified as shown in the diagram:



CLIENT SERVER ARCHITECTURE

Two-tier client-server Architecture:

- A two-tier client-server architecture is the simplest form of client-server architecture.
- The system is implemented as a single logical server plus an indefinite number of clients that use that servers.

CLIENT SERVER ARCHITECTURE

Thin-client model:

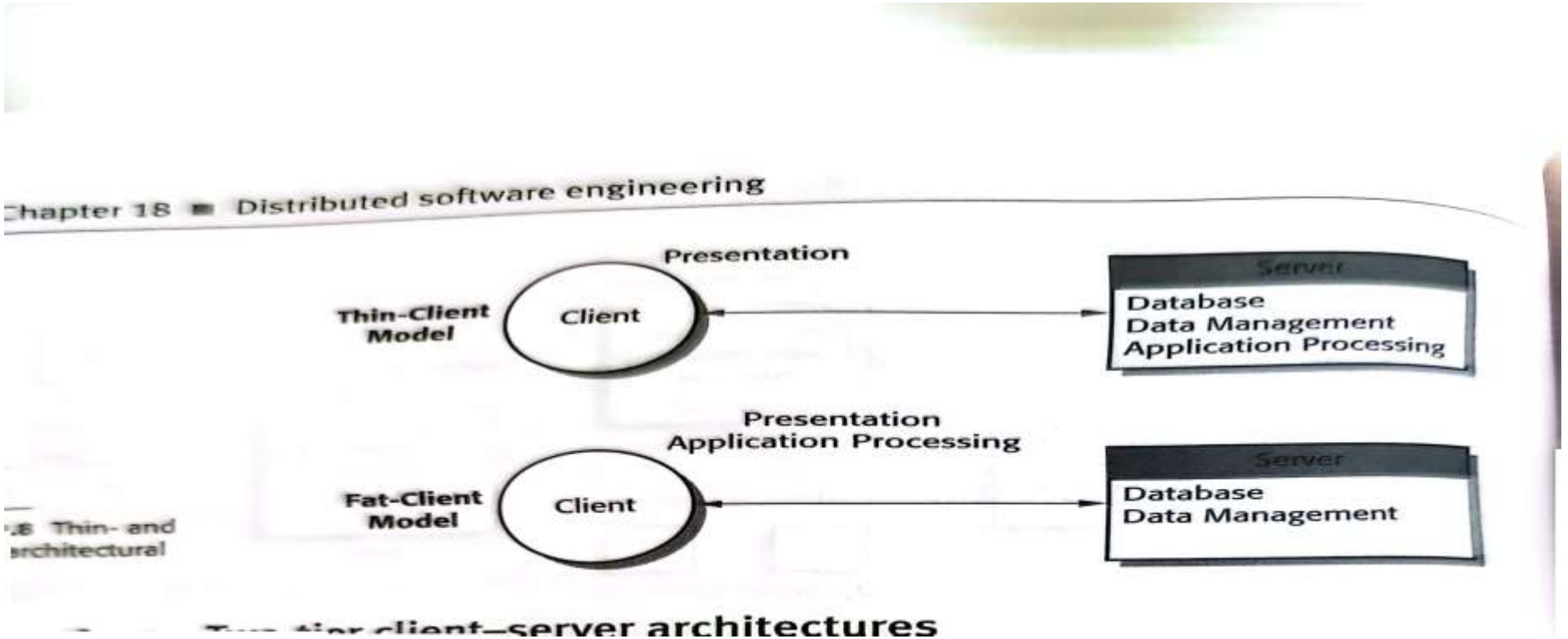
- In a thin-client model, all of the database, application processing and data management is carried out on the server.
- The client is simply responsible for running the presentation software.
- A major disadvantage is that It places a heavy processing load on both server and the network.

CLIENT SERVER ARCHITECTURE

Fat-client model:

- In this model, the server is only responsible for data management and database functions.
- The software on the client implements the application logic and the interactions with the system user.
- More processing is delegated to the client as the application processing is locally executed.
- Most suitable for new client-server systems where the capabilities for the client system are known in advance.
- More complex than the thin client model, especially for management.

CLIENT SERVER ARCHITECTURE



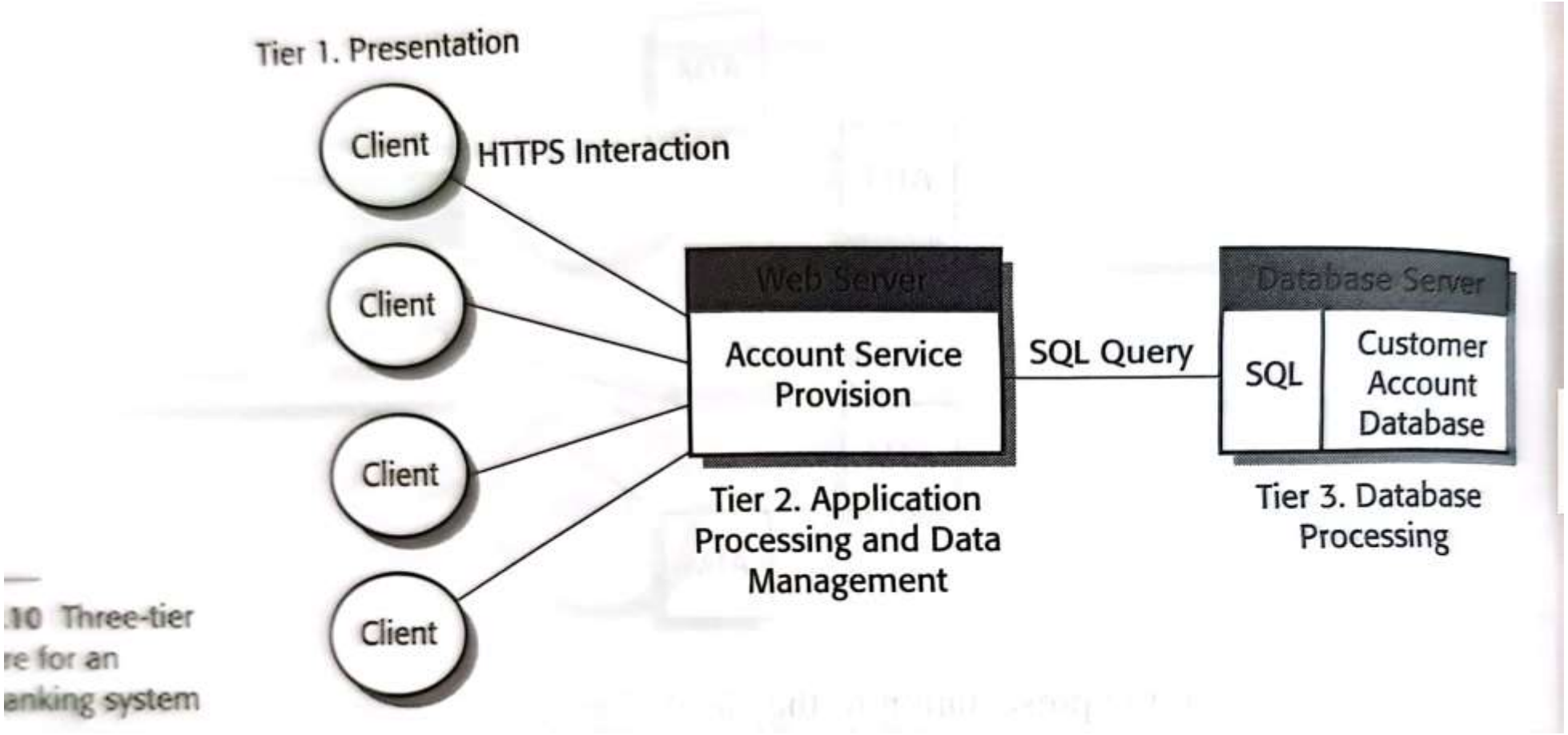
Thin and fat client architectural model

CLIENT SERVER ARCHITECTURE

Three-tier client server architecture:

- In three-tier architecture each of the application architecture layers may execute on a separate processor.
- Allows for better performance than a thin-client approach and is simpler to manage than a fat-client approach
- A more scalable architecture-as demands increase, extra servers can be added.

CLIENT SERVER ARCHITECTURE



Three-tier architecture for an internet banking system

CLIENT SERVER ARCHITECTURE

Client-Server Characteristics:

Advantages:

- Distribution of data is straightforward
- Makes effective use of networked systems
- May require cheaper hardware
- Easy to add new servers or upgrade existing servers.

Disadvantages:

- No shared data model so sub-systems. Use different data organization, data interchange may be inefficient.
- Redundant management in each server.
- No central register of names and services-it may be hard to find out what servers and services are available

DISTRIBUTED OBJECT ARCHITECTURE

- It is the system architecture where each distributable entity is an object that provides services to other objects and receives services from other objects
- There is no distinction in distributes object architecture between clients and servers.
- Used as a logical model that allows you to structure and organize the system
- Object communication is through a middleware system called an object request broker.
- However, distributed object architecture are more compex to design than c/s system.
- The ORB handles object communications.
- It knows of all objects in the system and their interfaces

DISTRIBUTED OBJECT ARCHITECTURE

Advantages:

- It is a very open system architecture that allows new resources to be added to it as required
- The system is flexible and scalable.
- It is possible to reconfigure the system dynamically with objects migrating across the network as required

DISTRIBUTED OBJECT ARCHITECTURE

Disadvantages

- Distributed component architectures are difficult for people to visualize and understand
- They are more complex to design than client-server systems

Inter-organization distributed computing

- Used for security and inter-operability reasons.
- Local standards, management and operational processes apply for such inter-organizational distribution computing
- Newer models of distributed computing have been designed to support inter-organizational computing where different nodes are located in different organizations.
- For example:
- Government Systems: Government agencies often need to collaborate and exchange data across different departments and agencies. Inter-organizational distributed systems can facilitate this information sharing, allowing government entities to work together on various initiatives, such as sharing citizen data, coordinating emergency responses, or managing public infrastructure.

CORBA-COMMON OBJECT REQUEST BROKER ARCHITECTURE

- Corba is an architecture and specification for creating, distributing, and managing distributed program objects in a network
- It is and international standard for an object Request broker-middleware to manage communications between distributes objects.
- Allows programs at different locations and developed by different vendors to communicate in a network through an interface broker
- Developed by a consortium of vendors through the object management group which currently includes over 500 member companies
- Facilities the communication of systems that are written in different languages and deployed on diverse platforms
- Enables collaboration between systems on different operating systems, programming languages and computing hardware.