COMP201 – Software Engineering I
Lecture 16: Architectural Design

Lecturer: T. Carroll

Email: Thomas.Carroll2@Liverpool.ac.uk

Office: G.14

See Vital for all notes

Coming Up…

What is Architectural Design?

Establishing the Overall Structure of a Software System

Architectural Design:

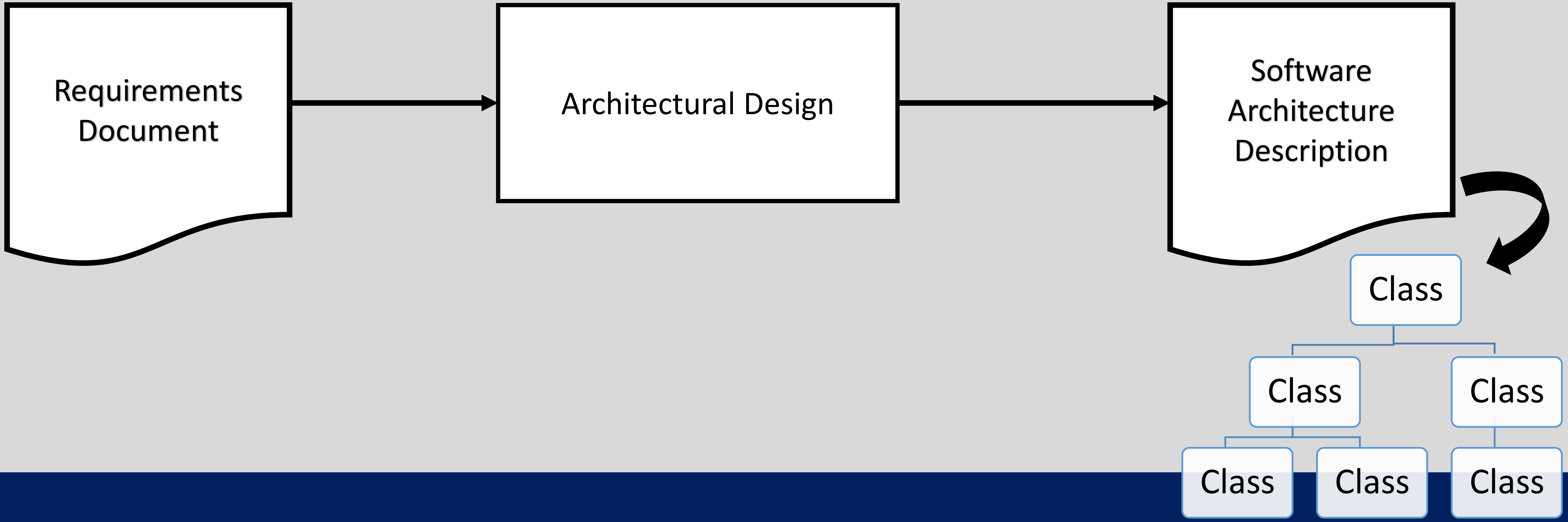
- System structuring
- Control models
- Modular decomposition

Distributed System Architectures:

- Multiprocessor architectures
- Client-server architectures
- Distributed object architectures

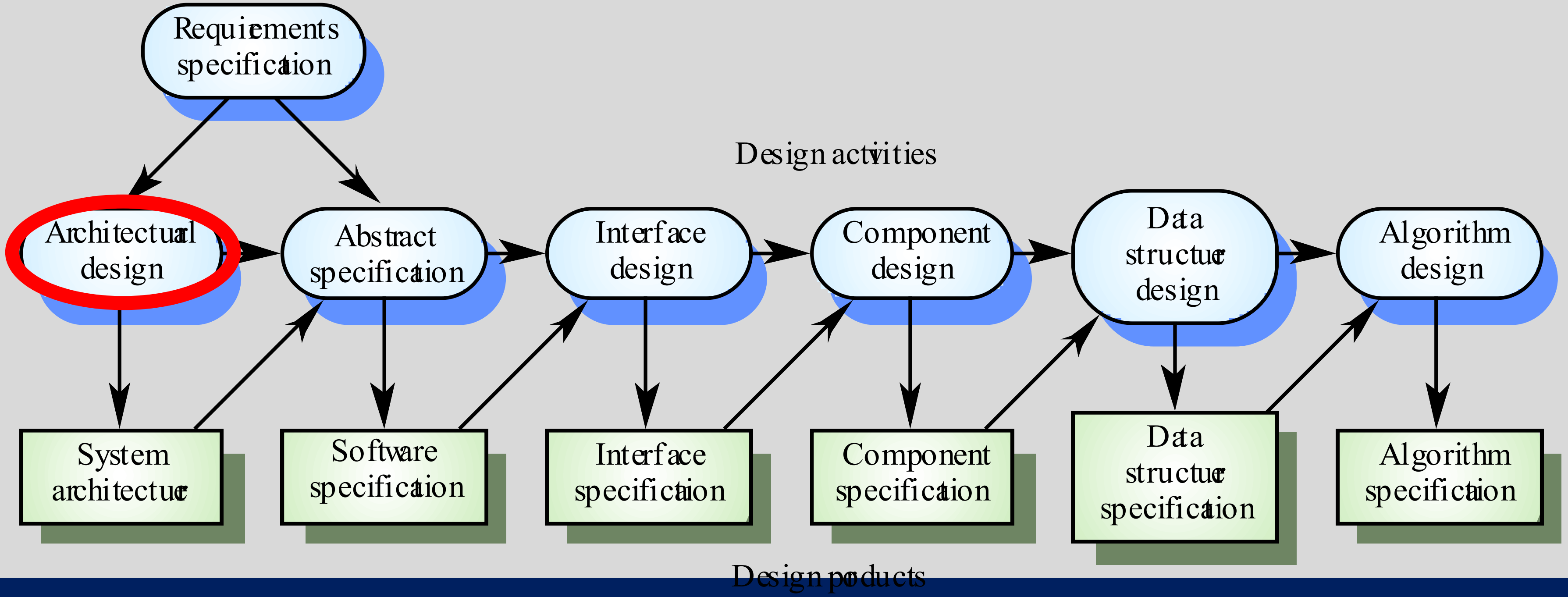
Architectural Design? Software Architecture?

- *Architectural Design*: The design process for:
  - identifying the sub-systems making up a system
  - Identifying the framework for sub-system control and communication.
- *Software Architecture (description of)*: The output of this design process.



Architectural Design

- Should be an early stage of the system design process
- Represents the link between specification and design processes
- Often carried out in parallel with some specification activities
- It involves identifying **major system components** and their **communications**



Architectural Design Process

Architectural Design Process

- **System structuring**
  - The system is decomposed into several principal sub-systems and communications between these sub-systems are identified
- **Control modelling**
  - A model of the control relationships between the different parts of the system is established
- **Modular decomposition**
  - The identified sub-systems are decomposed into modules

Sub-systems and Modules

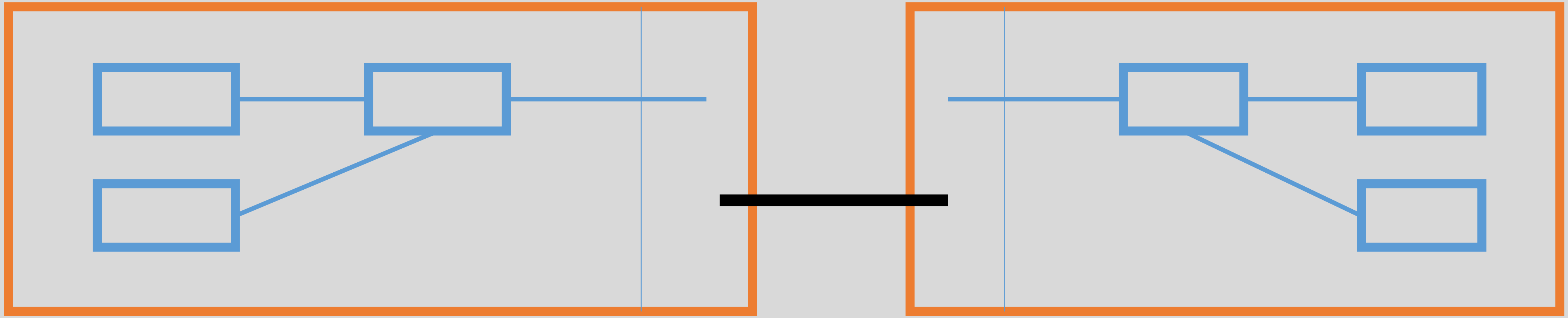
A **sub-system** is a system in its own right.
Operation is independent of the services of other sub-systems.

A **module** provides services to other components.
Not considered a separate system

Real world Sub-system examples

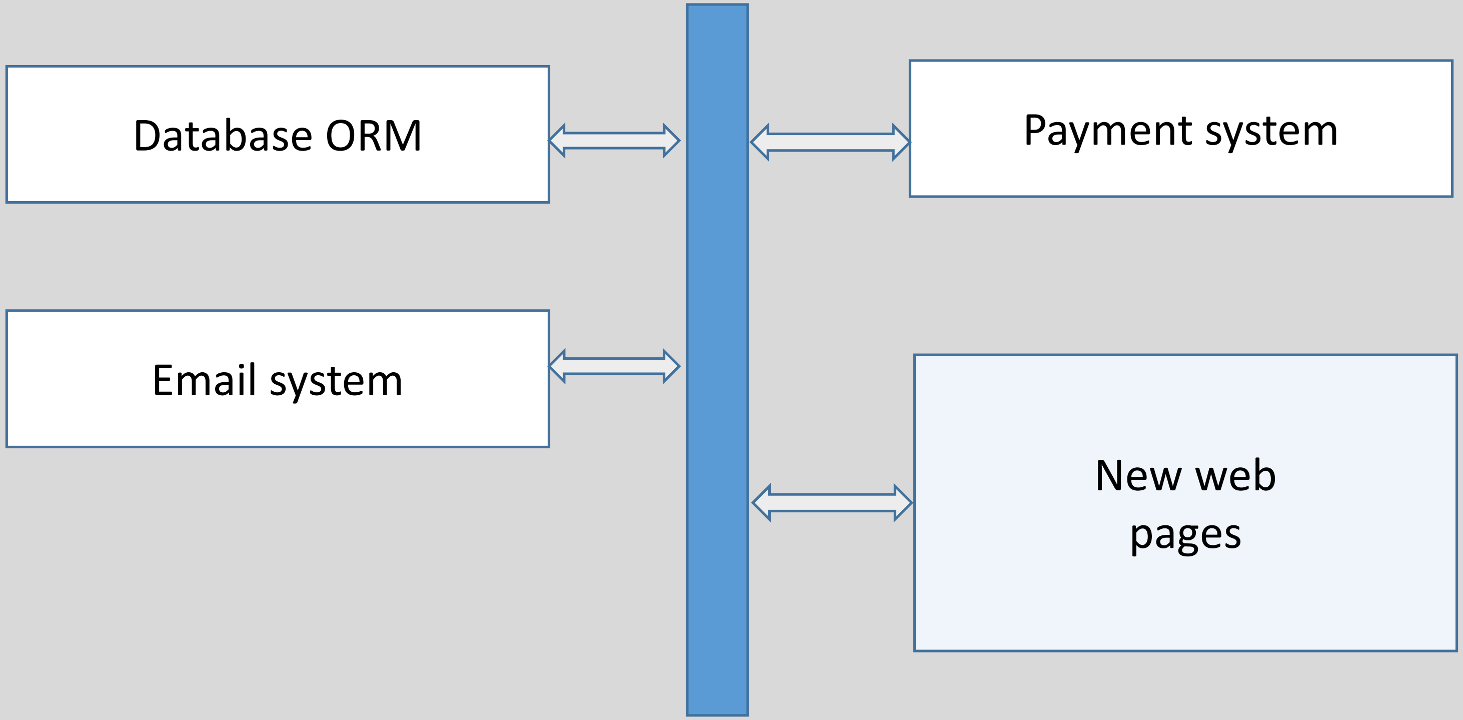
- Typically organized as Java packages/C++ libraries/C# assemblies
- Database access layer
  - MySQL access, JDBC layer
- Security services
  - Encryption classes, signature classes (modules)
- External Payment sub-system
- Email service sub-system
- Logging sub-system
- Financial transaction sub-system
- Marketing sub-system

Sub-systems and Modules



Benefits of sub-system modelling (eg: Ecommerce Site)

- You can now use sub-systems to build new system:



Architectural Models

- Static structural model
- Dynamic process model
- Interface model
- Relationships model

Architectural Models

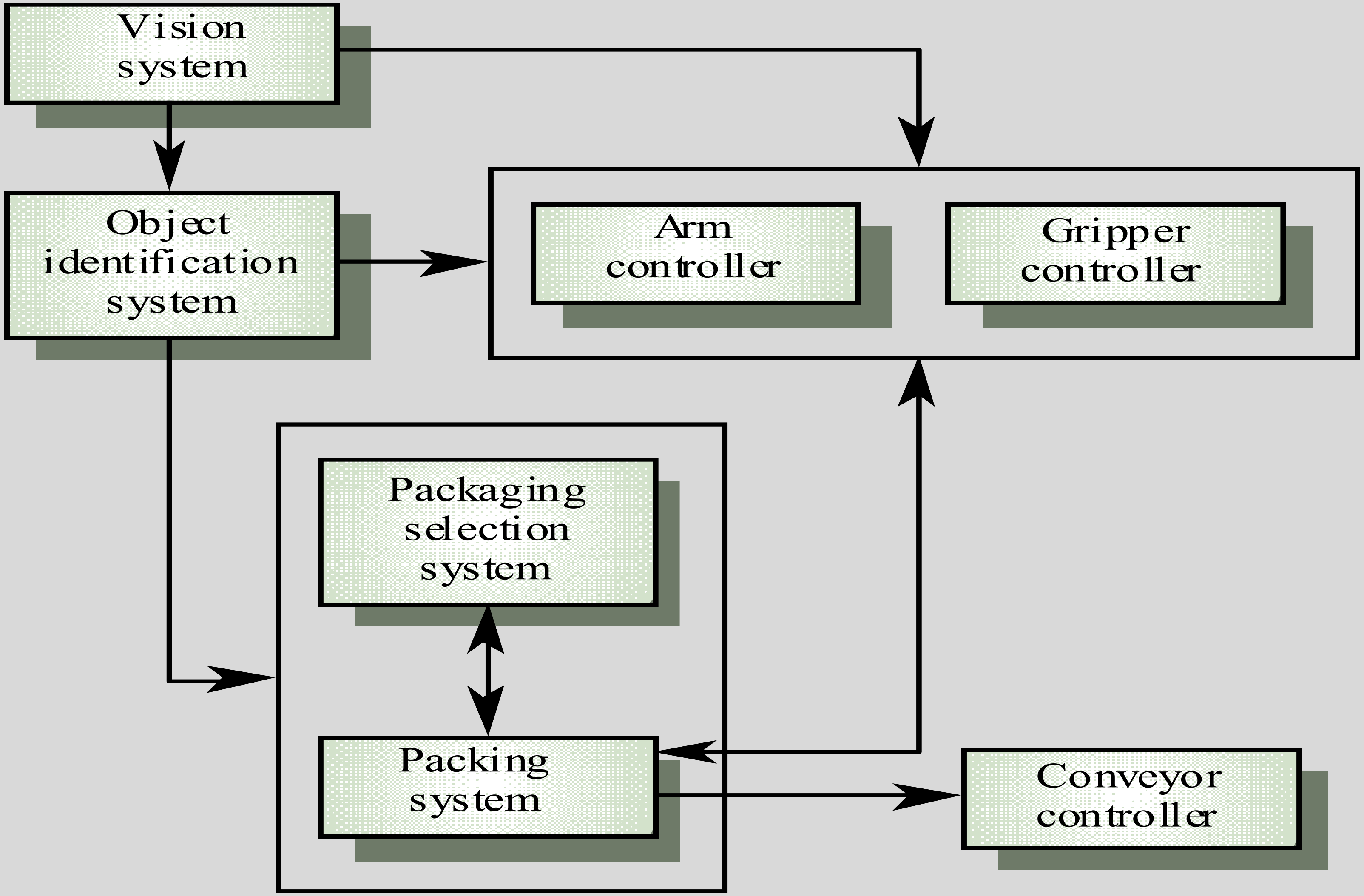
- **Different architectural models** may be produced during the design process
- Each model presents different perspectives on the architecture:
  - **Static structural models** show the major system components
  - **Dynamic process models** show the process structure of the system
  - **Interface models** define sub-system interfaces
  - **Relationships models** such as a data-flow model

System Structuring

Concerned with decomposing the system into interacting sub-systems

- The architectural design is normally expressed as a block diagram presenting an overview of the system structure
  - (More specific models showing how sub-systems share data, are distributed and interface with each other may also be developed)

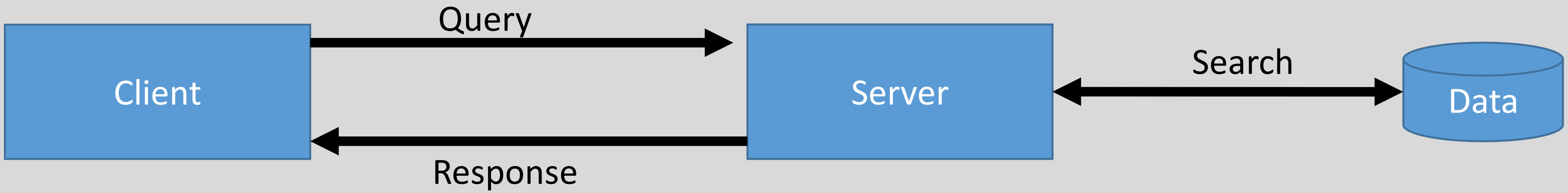
Example: Packing Robot Control System



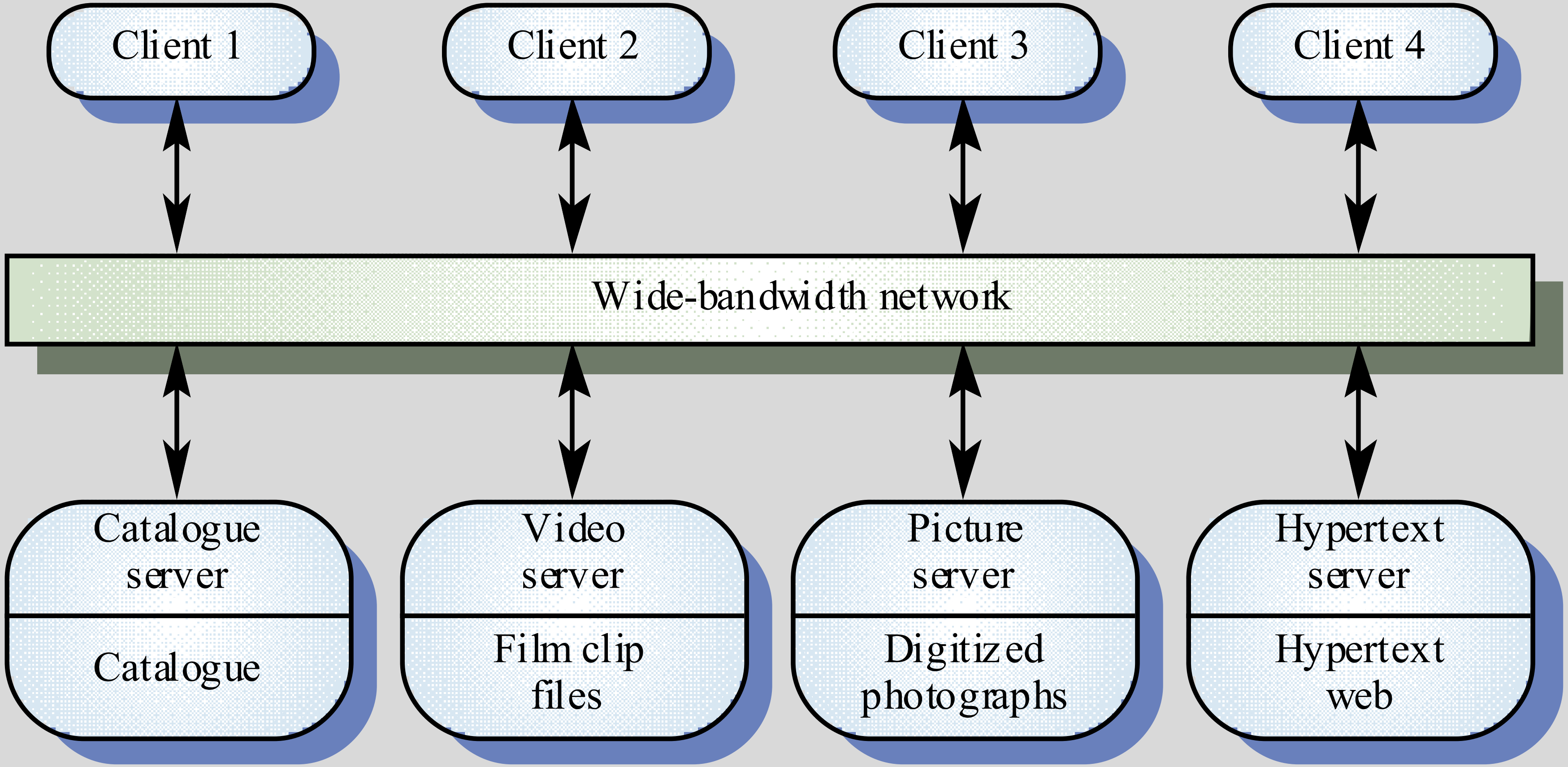
Client/Server Architecture

Client-Server Architecture

- Distributed system model
- Shows how data and processing is distributed across a range of components:
  - **Servers** provide specific services such as printing, data management, etc.
  - **Clients** call on these services
  - **Network** allows clients to access servers



Example: Film and Picture Library



Client-Server Characteristics

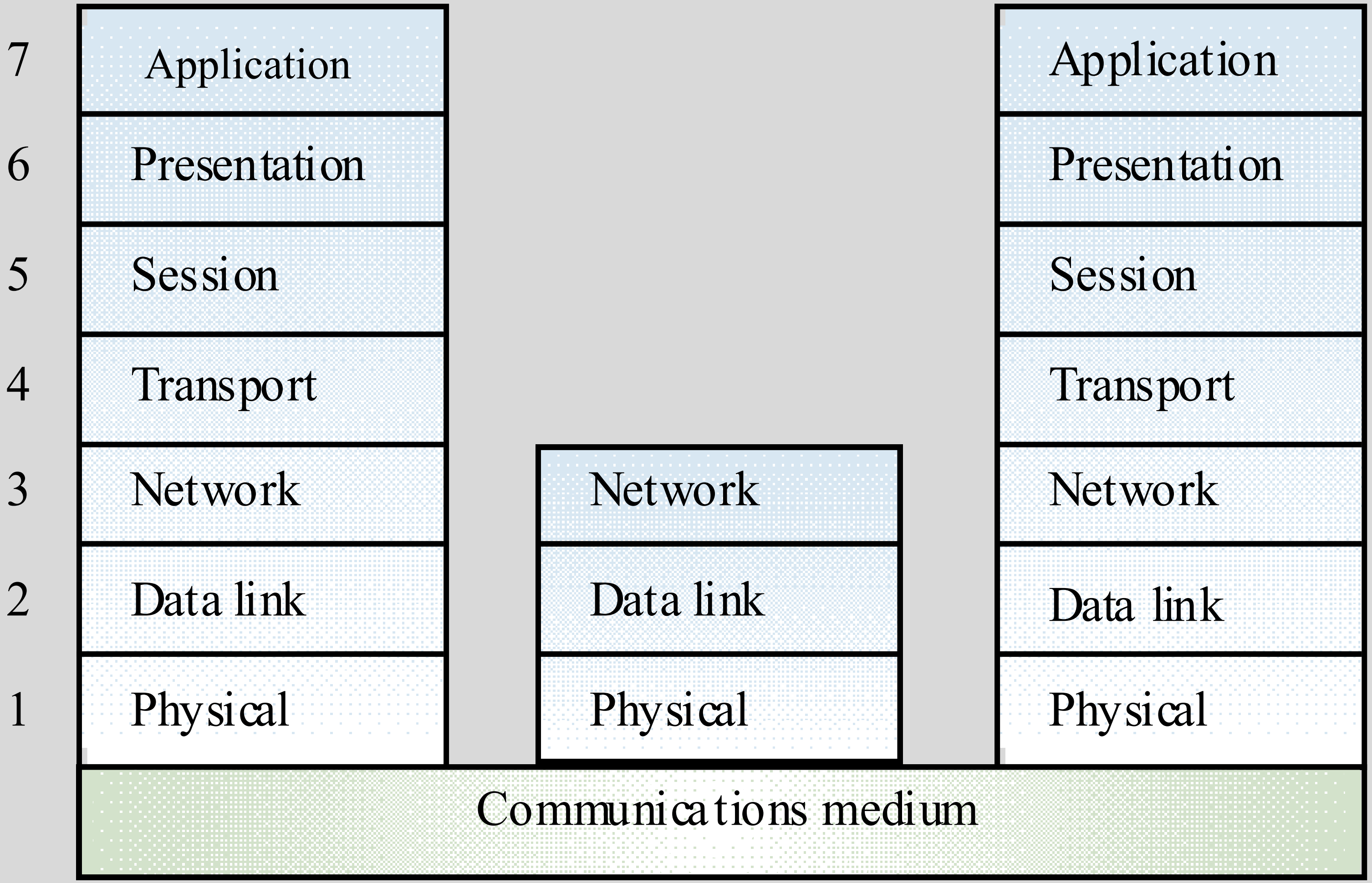
- Advantages
  - ✓Distribution of data is straightforward
  - ✓Makes effective use of networked systems.
  - ✓Could get away with using cheaper hardware
  - ✓Easy to scale
- Disadvantages
  - × No shared data model
  - × Redundant management in each server
  - × No central register of names and services

Abstract Machine Model

Abstract Machine Model

- Used to model the interfacing of sub-systems
- Organises the system into layers (or abstract machines)
  - Each layer provides a set of services
- Supports the incremental development of sub-systems in different layers.
- When a layer interface changes, only the adjacent layer is affected
- Can be difficult to structure a system in this way

Abstract Machine Model Example:
ISO/OSI Network Model



7	Application		Application
6	Presentation		Presentation
5	Session		Session
4	Transport		Transport
3	Network	Network	Network
2	Data link	Data link	Data link
1	Physical	Physical	Physical

Communications medium

Control Models

Control Models

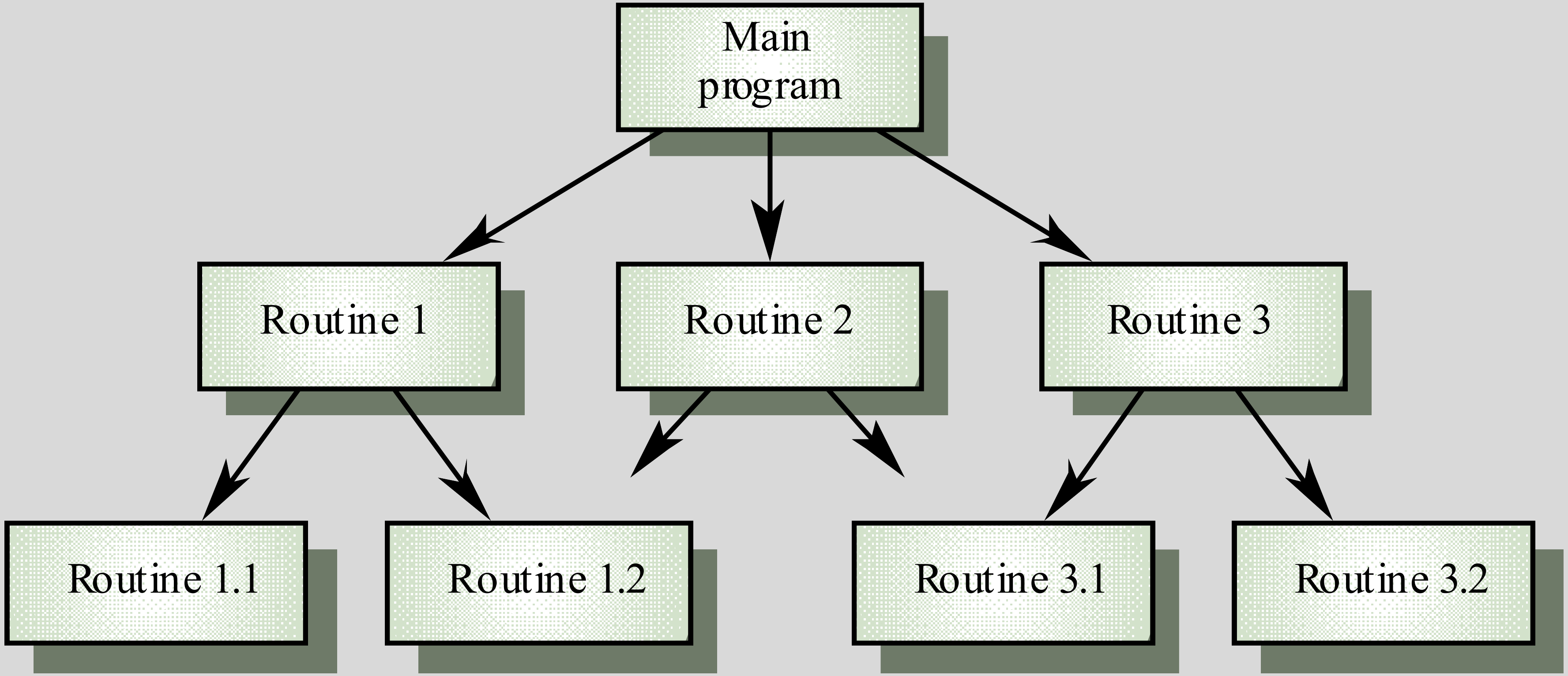
Control Models are concerned with the control flow between sub systems:

- **Centralised control**
  - One sub-system has overall responsibility for control
- **Event-based control**
  - Each sub-system can respond to externally generated events

Centralised Control

- A control sub-system takes responsibility for managing the execution of other sub-systems.
- There are two main types of centralised control models (sequential or parallel):
  - **Call-Return model (Top-Down subroutine model)**
    - Control starts at the top of a subroutine hierarchy and moves downwards. Applicable to **sequential systems**
    - Such a model is embedded into familiar programming languages such as C, Java …

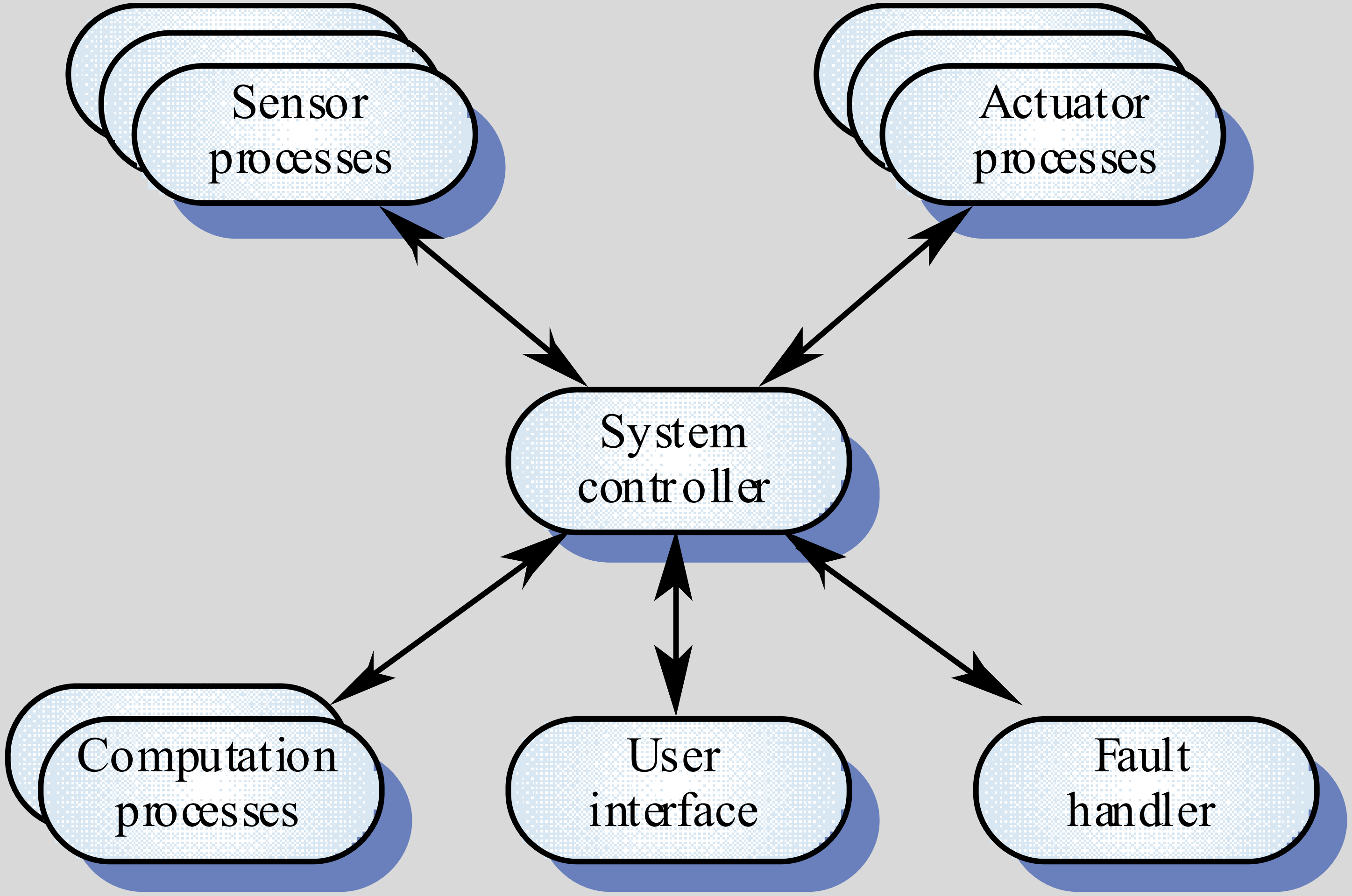
Call-Return Model



Centralised Control

- If the controlled subsystems run **in parallel**, then we may use the manager model of centralised control:
  - **Manager model -** Applicable to concurrent systems.
  - One system component controls the stopping, starting and coordination of other system processes.
  - Can also be implemented in sequential systems as a case statement.

Real-Time System Control



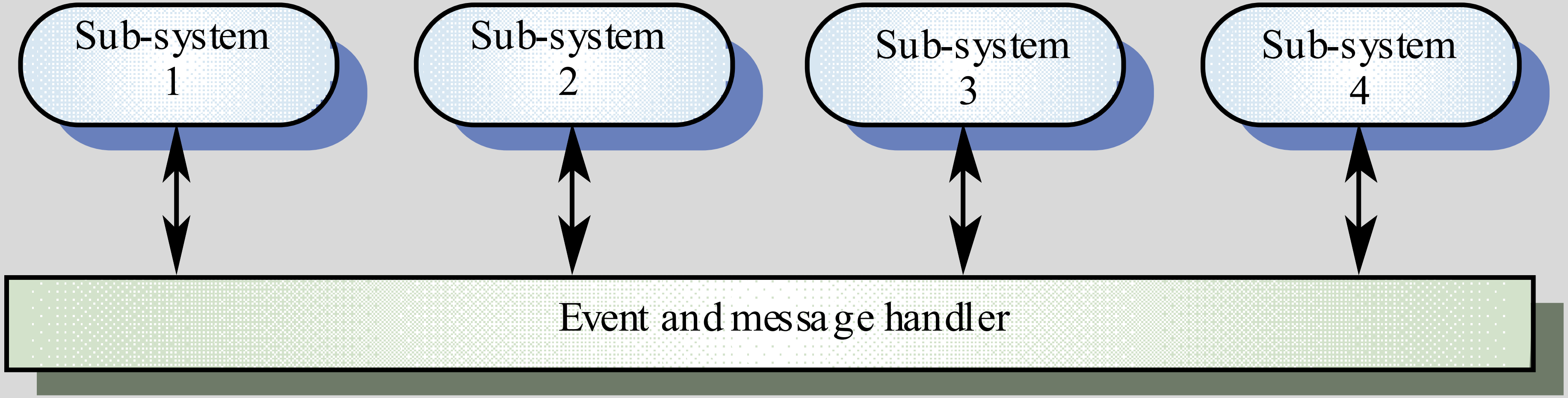
Event-Driven Systems

- **Driven by externally generated events** where the timing of the event is out of the control of the sub-systems which process the event
- There are two principal event-driven models:
  - **Broadcast models**. An event is broadcast to all sub-systems. Any sub-system which can handle the event may do so
  - **Interrupt-driven models**. Used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing

Broadcast Model

- **Effective in integrating sub-systems** on different computers in a network
- Sub-systems register an interest in specific events. When these occur, control is transferred to the sub-system which can handle the event
- **Control policy is not embedded in the event** and message handler.
- Sub-systems decide on events of interest to them
- **However**, sub-systems don’t know if or when an event will be handled

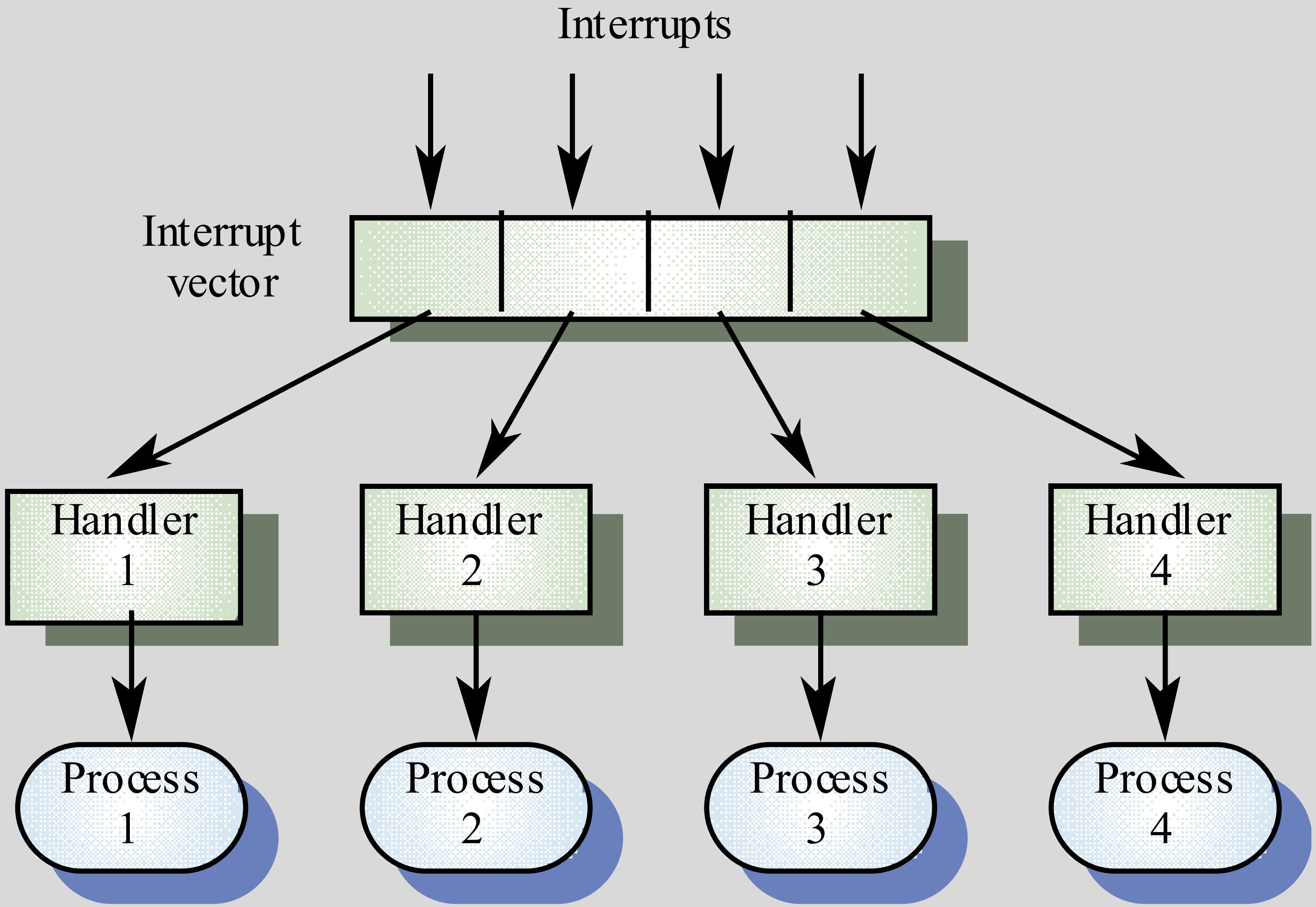
Selective Broadcasting



Interrupt-Driven Systems

- **Used in real-time systems** where fast response to an event is essential
- There are known interrupt types with a handler defined for each type
- Each type is associated with a memory location and a hardware switch causes transfer to its handler
- Allows fast response but complex to program and difficult to validate

Interrupt-Driven Control



Modular Decomposition

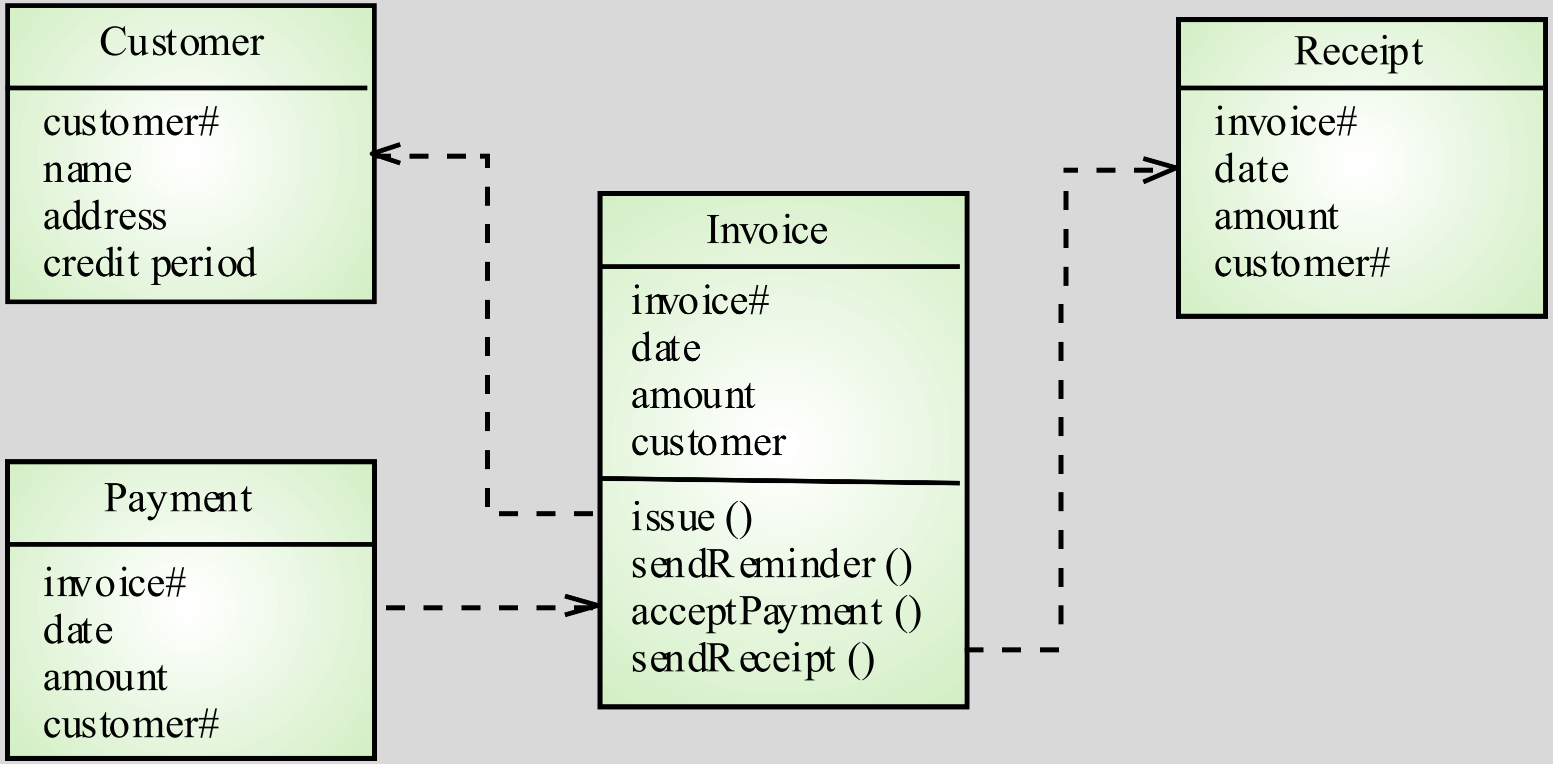
Modular Decomposition

- Another structural level where sub-systems are decomposed into modules
- Two modular decomposition models covered
  - An object model where the system is decomposed into interacting objects
  - A data-flow model where the system is decomposed into functional modules which transform inputs to outputs. Also known as the pipeline model
- If possible, decisions about concurrency should be delayed until modules are implemented

Object Models

- Structure the system into a set of loosely coupled 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

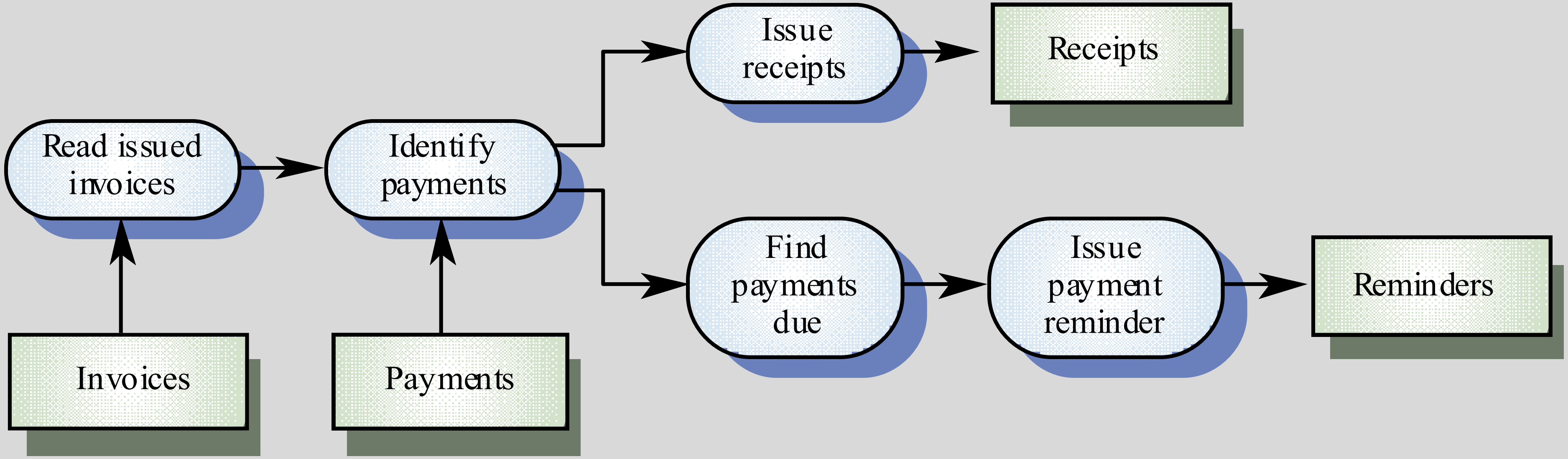
Invoice Processing System



Data-Flow Models

- Functional transformations process their inputs to produce outputs
- May be referred to as a pipe and filter model (as in UNIX shell)
- Variants of this approach are very common. When transformations are sequential, this is a batch sequential model which is extensively used in data processing systems
- Not really suitable for interactive systems

Invoice Processing System


Read issued invoices
Invoices
Identify payments
Payments
Issue receipts
Receipts
Find payments due
Issue payment reminder
Reminders