COMP201 Software Engineering I
Lecture 17 – Architectural Design

Lecturer: T.Carroll

Email: Thomas.Carroll2@Liverpool.ac.uk

Office: G.14

See Vital for all notes

Recap

What is Architectural Design?

Establishing the Overall Structure of a Software System

Architectural Design:

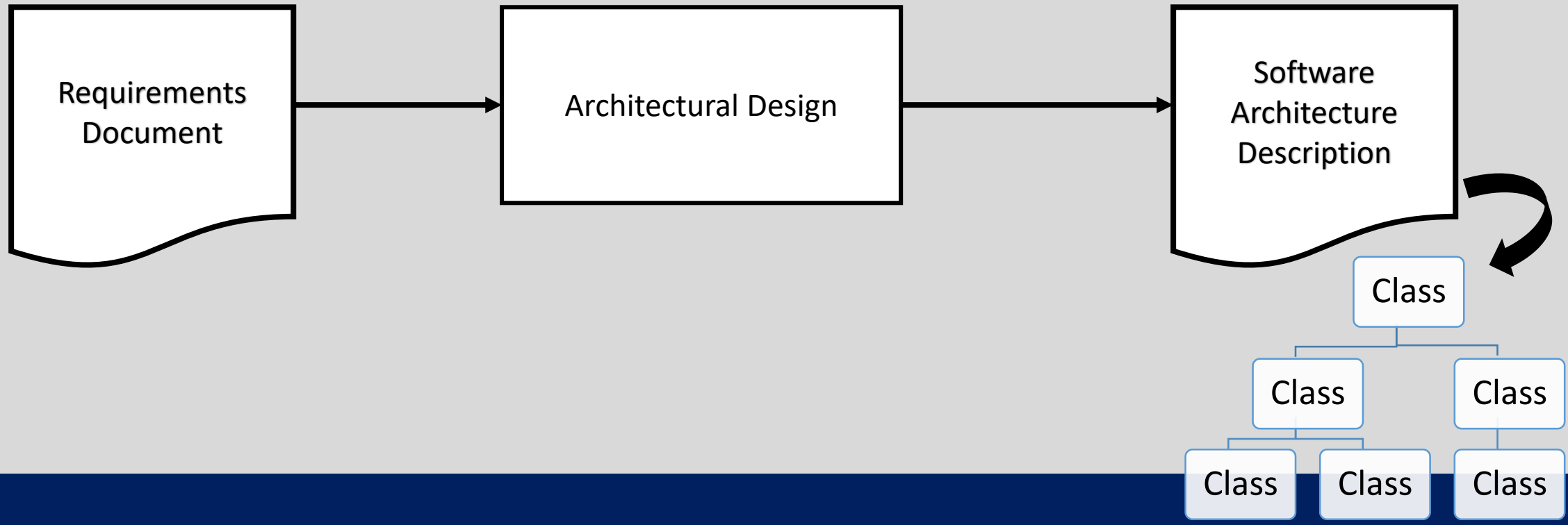
- System structuring
- Control models
- Modular decomposition

Distributed System Architectures:

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

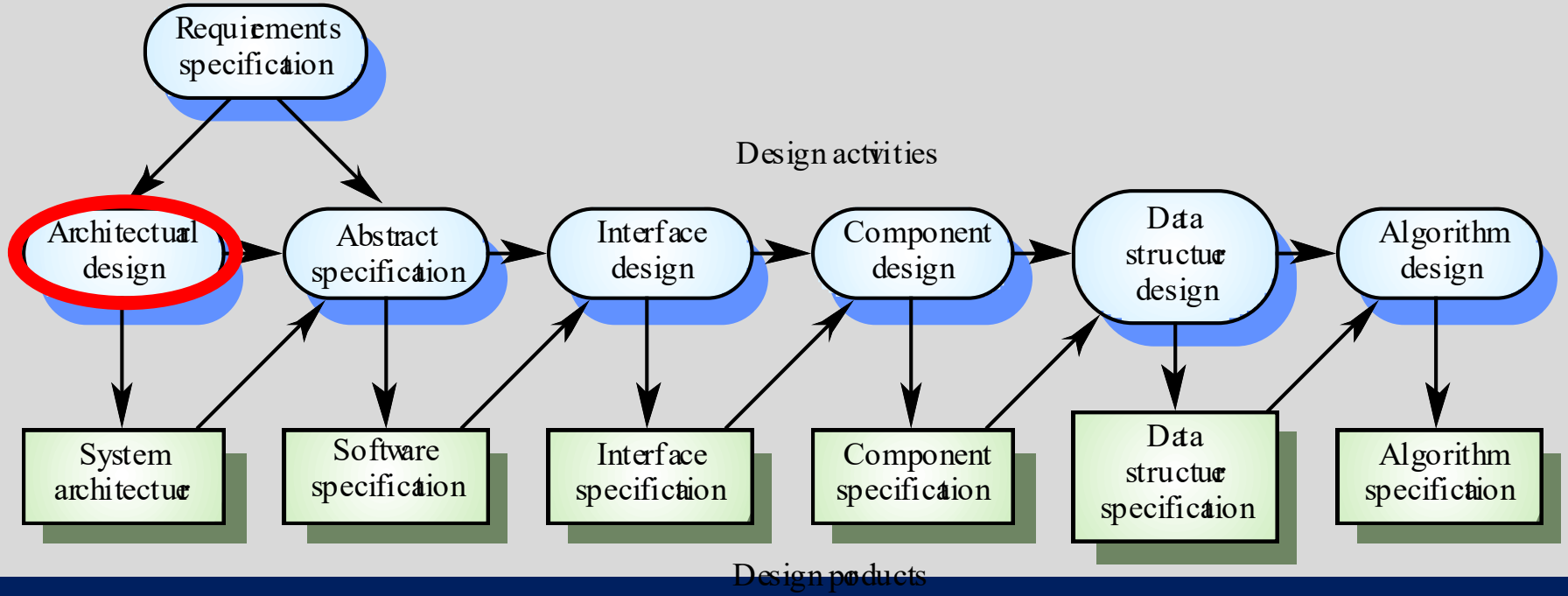
Architectural Design? Software Architecture?

- *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**



Recap – Lecture 16

Architectural Design: Establishing the Overall Structure of a Software System

- It is a **process** – outputs the software architecture description
- Early-on in the software design process
- Sub-System Modelling
- System Organisation
  - Models on how subsystems are organised and communicate…
  - Client-Server architecture
  - Abstract Machine Model (layered)
  - (Repository model) – Lecture
- Control models
  - How is the control flow regulated within the system?
  - Top-down (centralised, sequential)
  - Manager model (centralised, distributed)
  - Broadcast model (event driven)

Today

Overview – Lecture 17

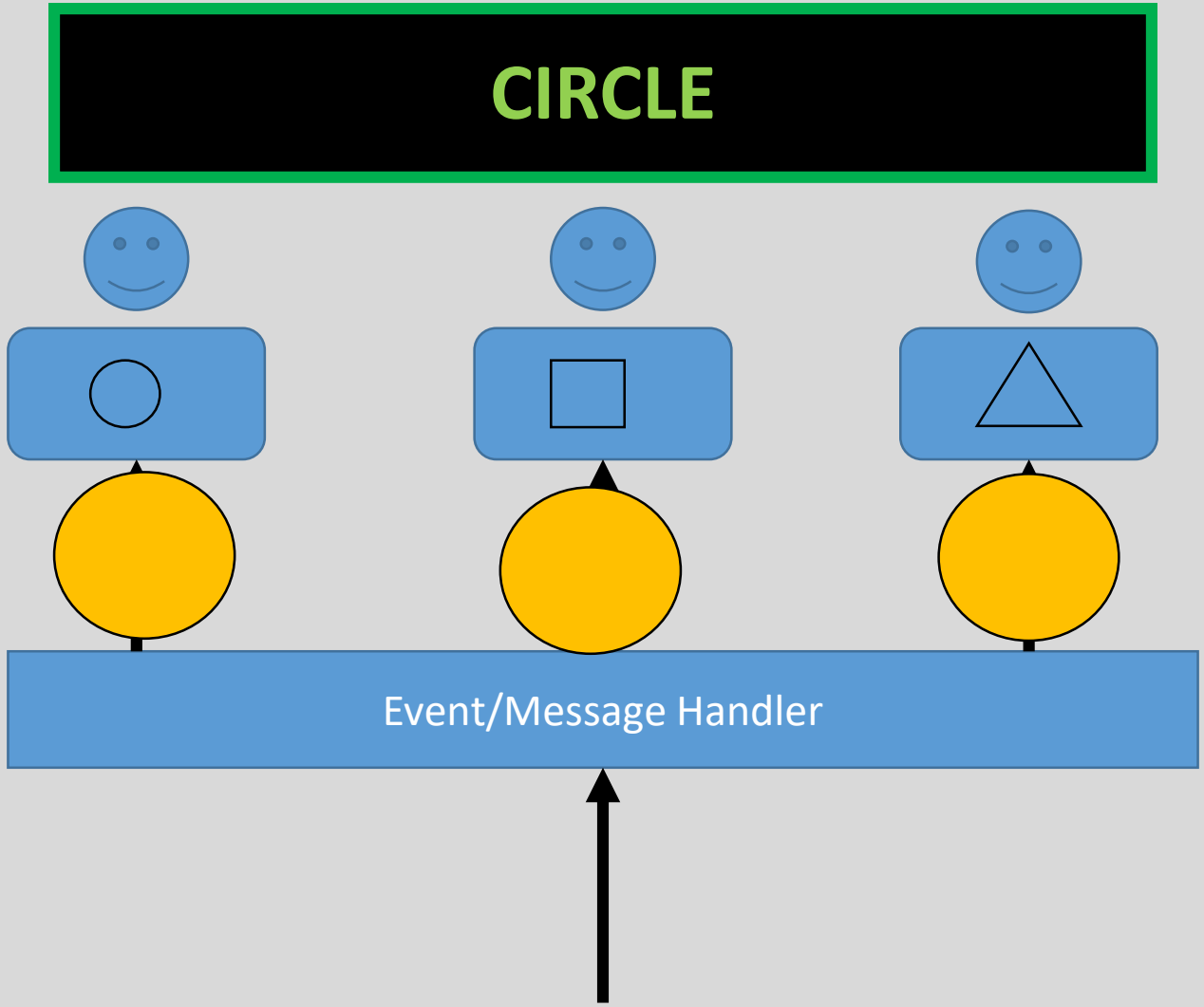
- Control Models continued…
  - Broadcast model recap
  - Interrupt driven system
- Modular Decomposition
  - Object model
  - Data flow model
- Distributed System Architectures

Broadcast Model Recap

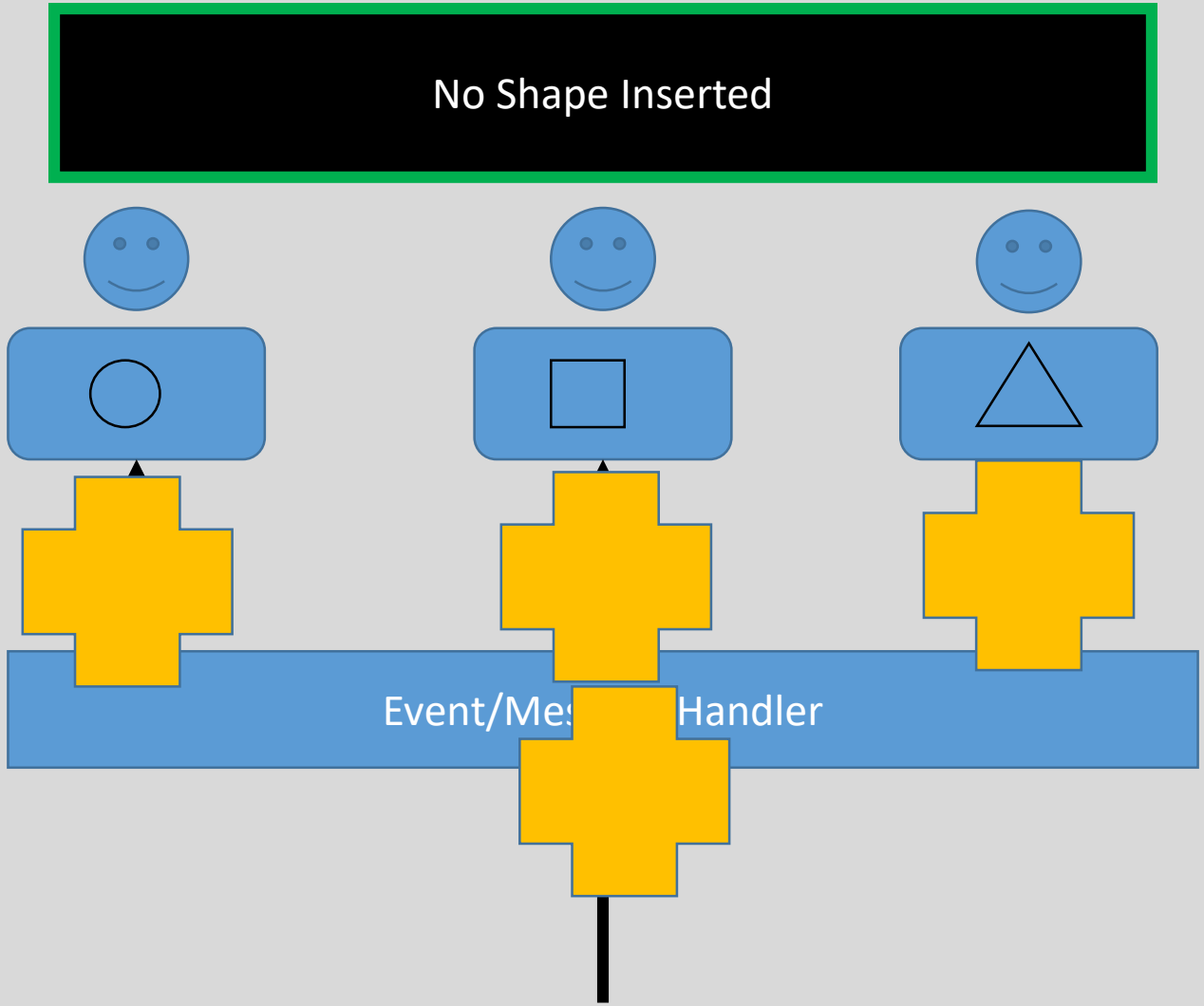
Broadcast Model Recap

- 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

Example: Shape Game



Example: Shape Game

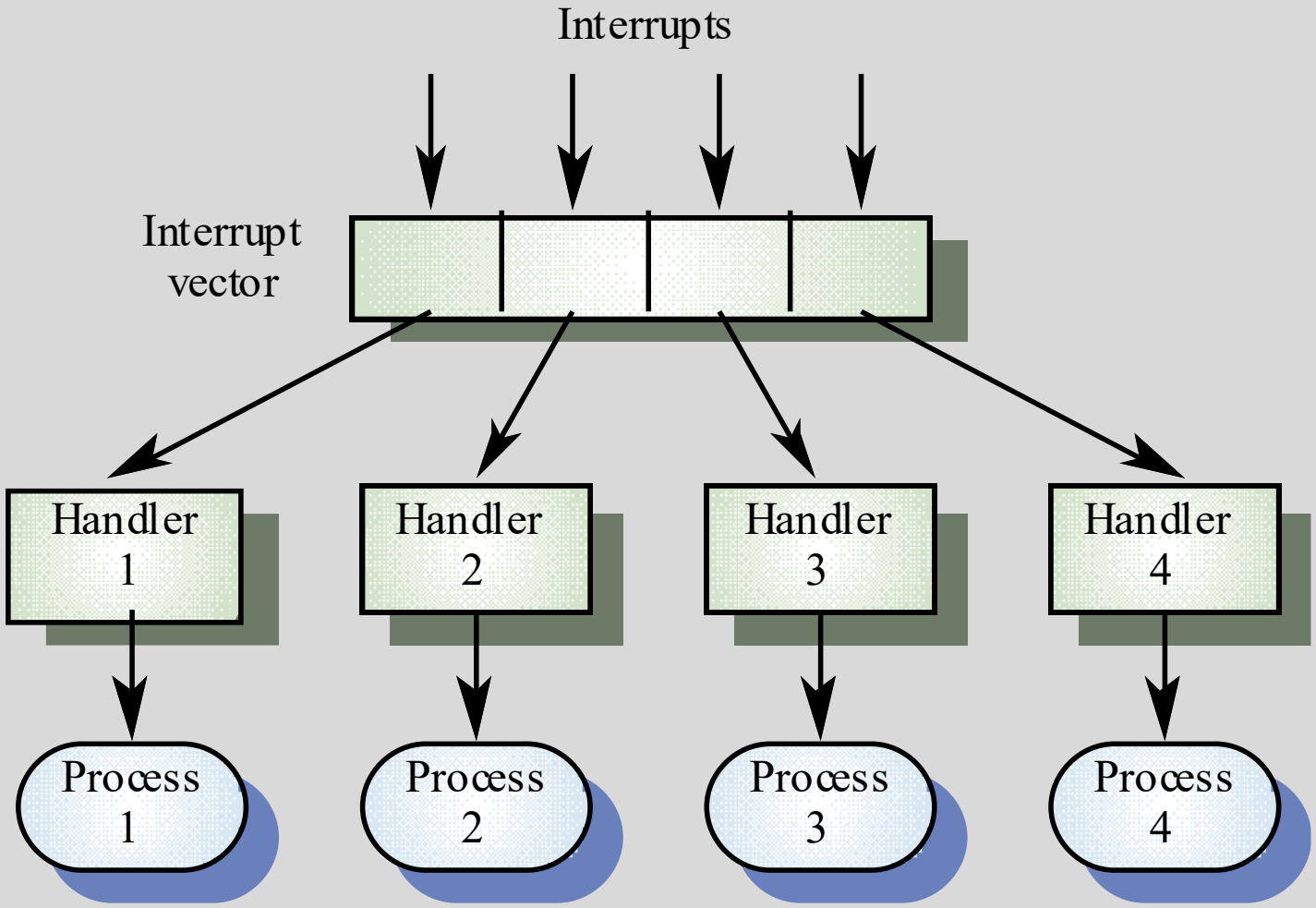


Interrupt Driven Systems

Interrupt-Driven Systems

- **Used in real-time systems** where fast response to an event is essential
- Interrupt types are pre-defined
- Each type has a handler
- Each type is associated with a memory location
- Hardware switch causes transfer to its handler
- Allows fast response but complex to program and difficult to validate

Interrupt-Driven Control



Modular Decomposition

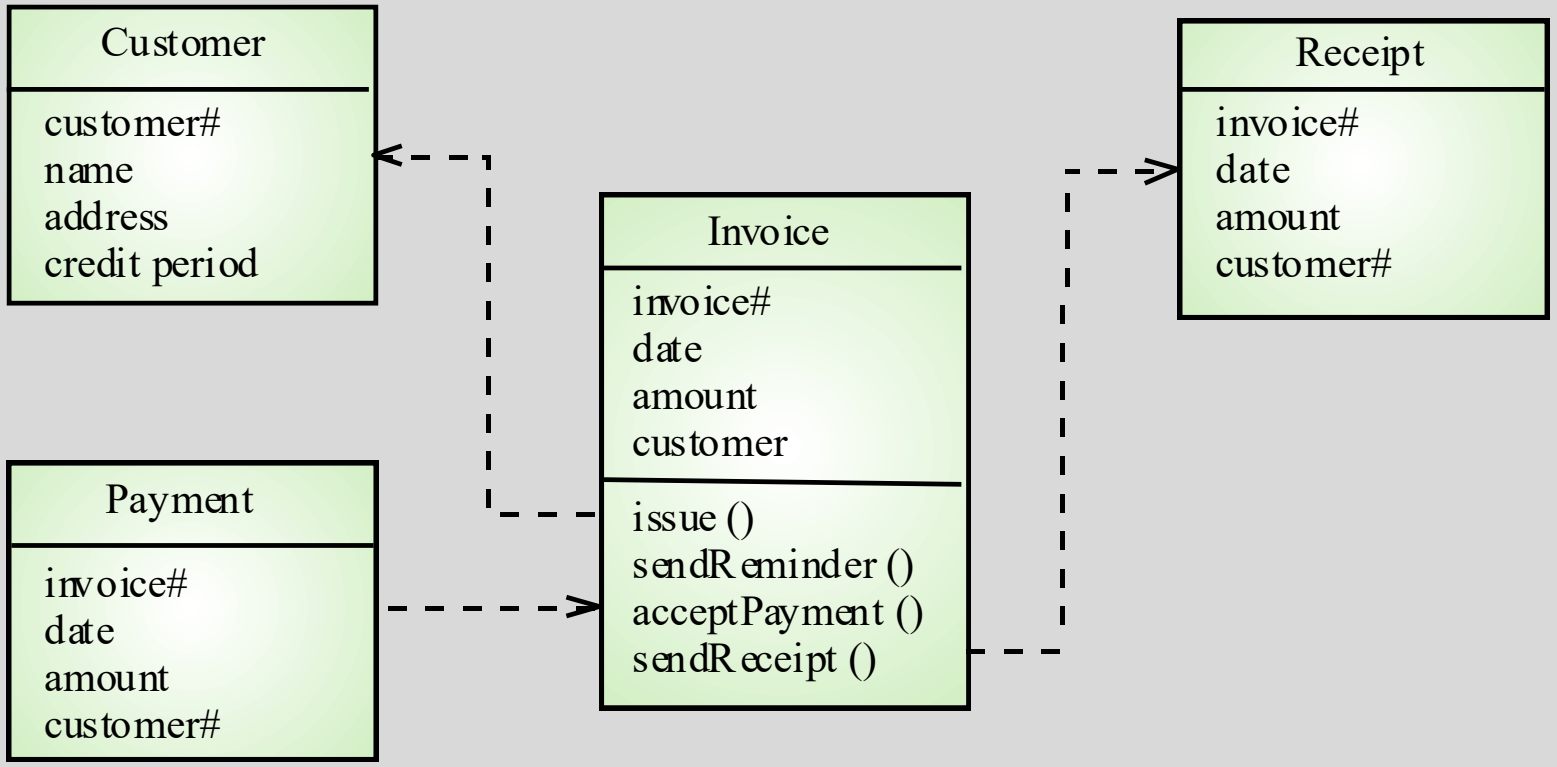
Modular Decomposition

- 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

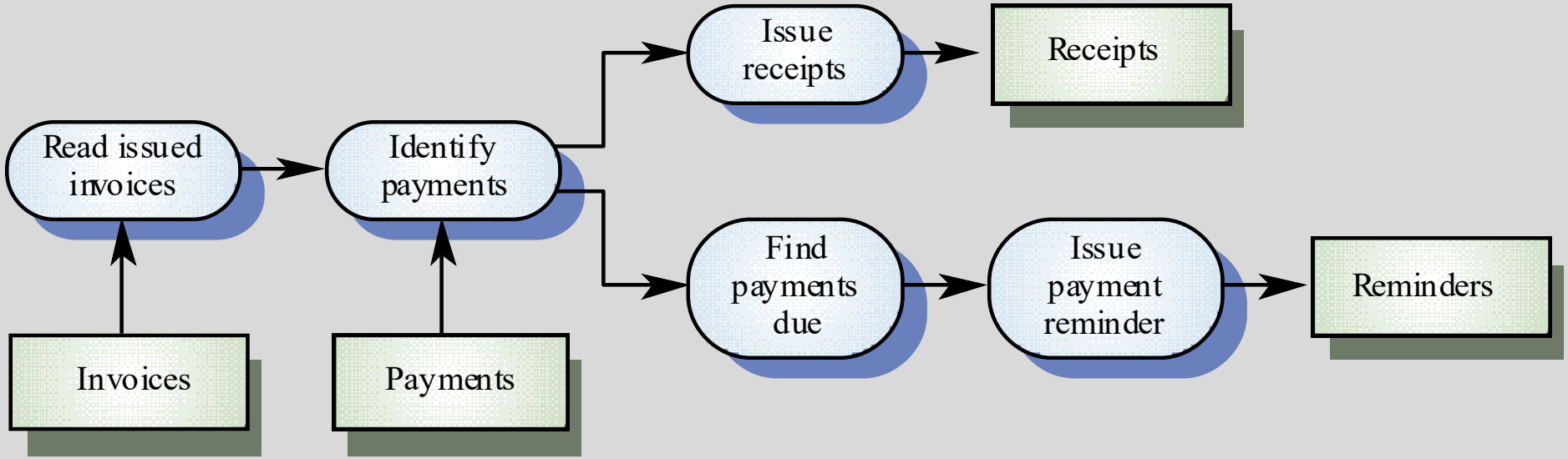
Invoice Processing System



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



Distributed Systems Architectures

Architectural design for software that executes on more than one processor

Distributed Systems

- Virtually all large computer-based systems are now distributed systems
- **Information processing** is distributed over several computers rather than confined to a single machine
- **Distributed software engineering** is now very important

System Types

- **Personal systems** that are not distributed and that are designed to run on a personal computer or workstation.
- **Embedded systems** that run on a single processor or on an integrated group of processors.
- **Distributed systems** where the system software runs on a loosely integrated group of cooperating processors linked by a network.

Distributed System Characteristics

- **Advantages:**
  - Resource sharing
  - Openness
  - Concurrency
  - Scalability
  - Fault tolerance
  - Transparency
- **Disadvantages:**
  - Complexity
  - Security
  - Manageability
  - Unpredictability

Distributed Systems Architectures

- **Client-server architectures**
  - Distributed services which are called on by clients.
  - Servers that provide services are treated differently from clients that use services
- **Distributed object architectures**
  - No distinction between clients and servers.
  - Any object on the system may provide and use services from other objects

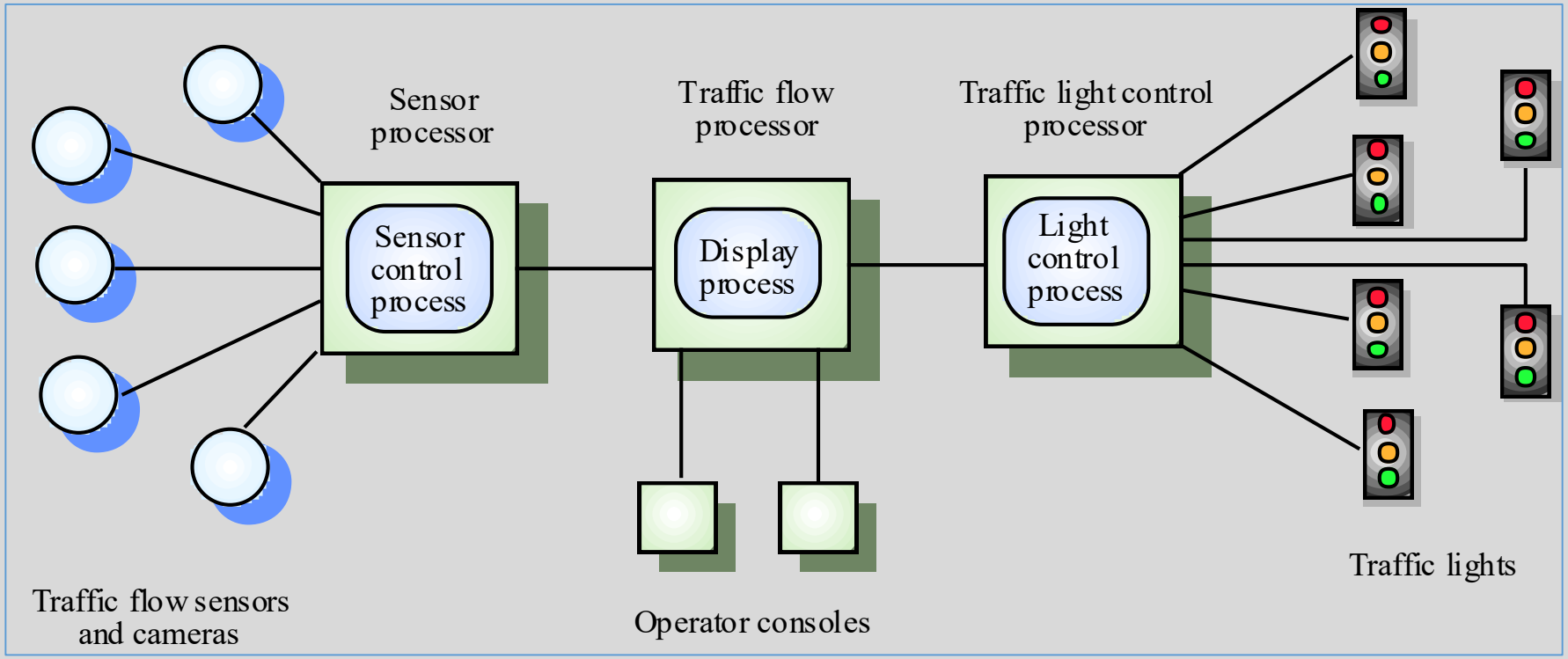
Middleware

- Software that manages and supports the different components of a distributed system.
- In essence, it sits in the ***middle*** of the system
- Usually **off-the-shelf** rather than specially written software
- Examples
  - Transaction processing monitors
  - Data converters
  - Communication controllers

Multiprocessor Architectures

- **Simplest distributed system model**
- System composed of multiple processes which may (but need not) execute on different processors
- Architectural model of many large real-time systems
- Distribution of process to processor:
  - may be pre-ordered
  - may be under the control of a dispatcher

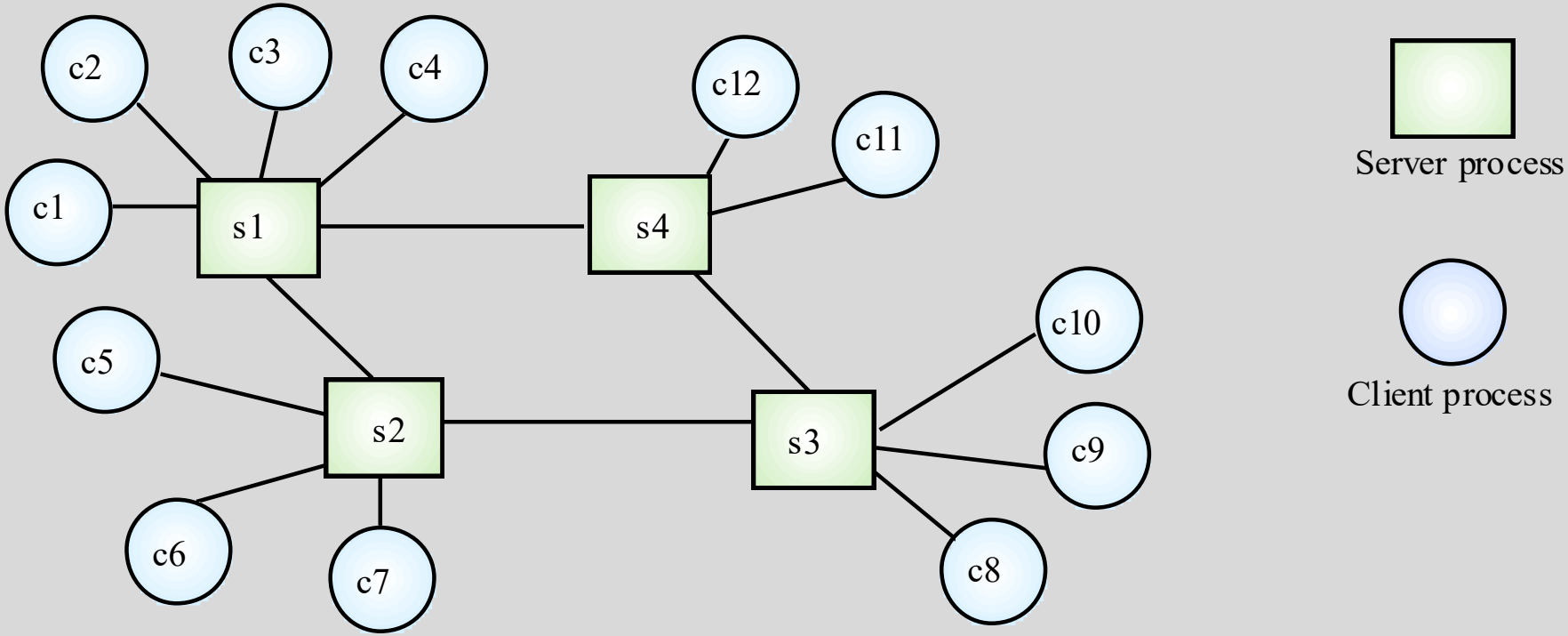
A Multiprocessor Traffic Control System



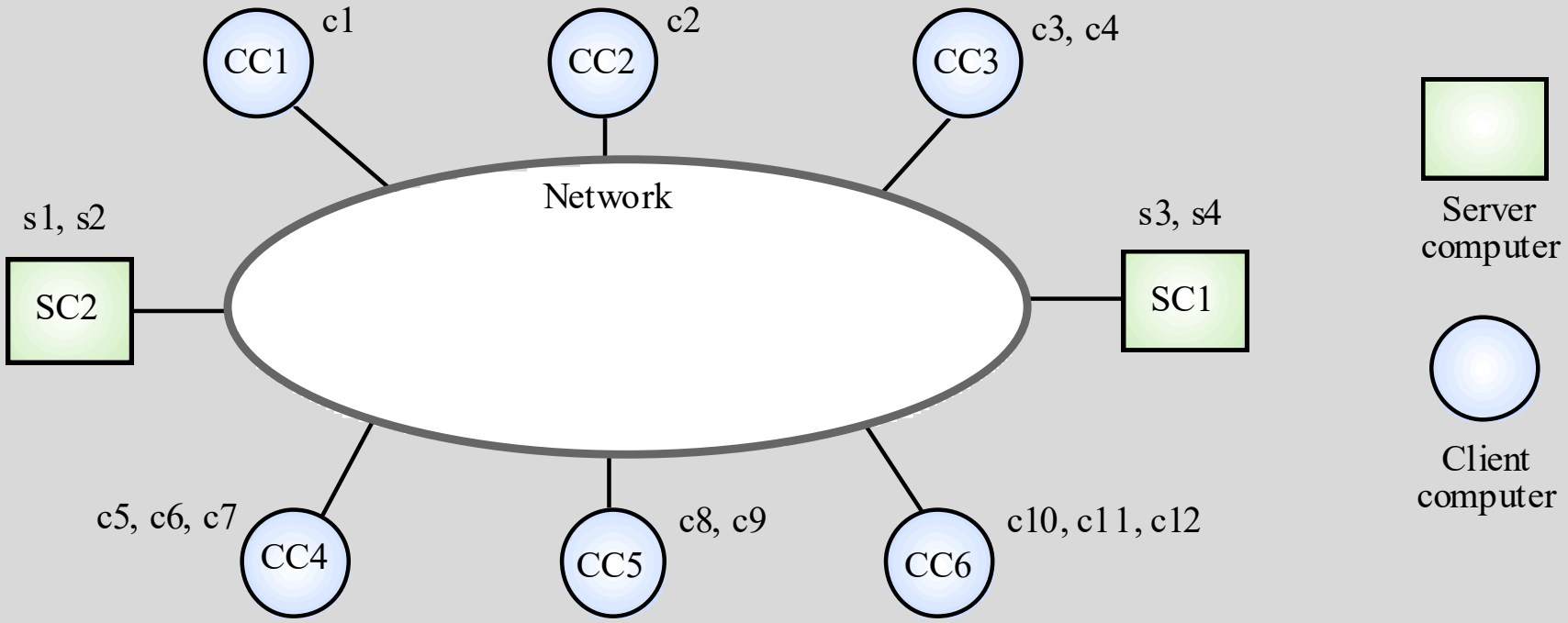
Client-Server Architectures

- The application is modelled as a set of services
  - Provided by servers
  - Used by a set of clients
- **Clients know of servers but servers need not know of clients**
- Clients and servers are logical processes
- The mapping of processors to processes is not necessarily 1 : 1

A Client-Server System



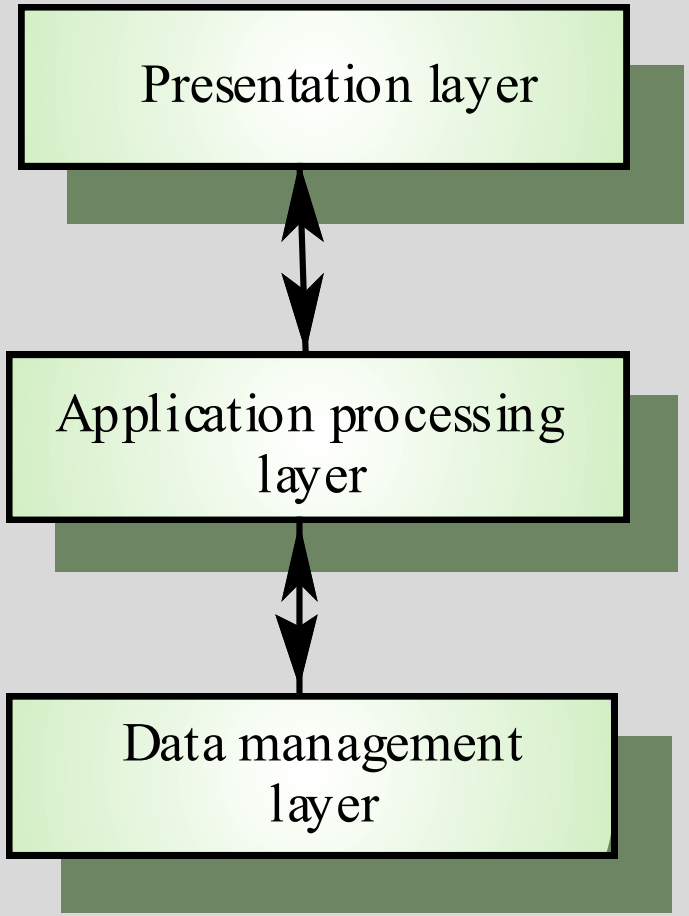
Computers in a C/S Network



Layered Application Architecture

- Presentation layer
  - Concerned with presenting the results of a computation to users
  - Collects user inputs
- Application processing layer
  - Provides application specific functionality
    - e.g., in a banking system: banking functions such as open account, close account, etc.
- Data management layer
  - Manages the system databases

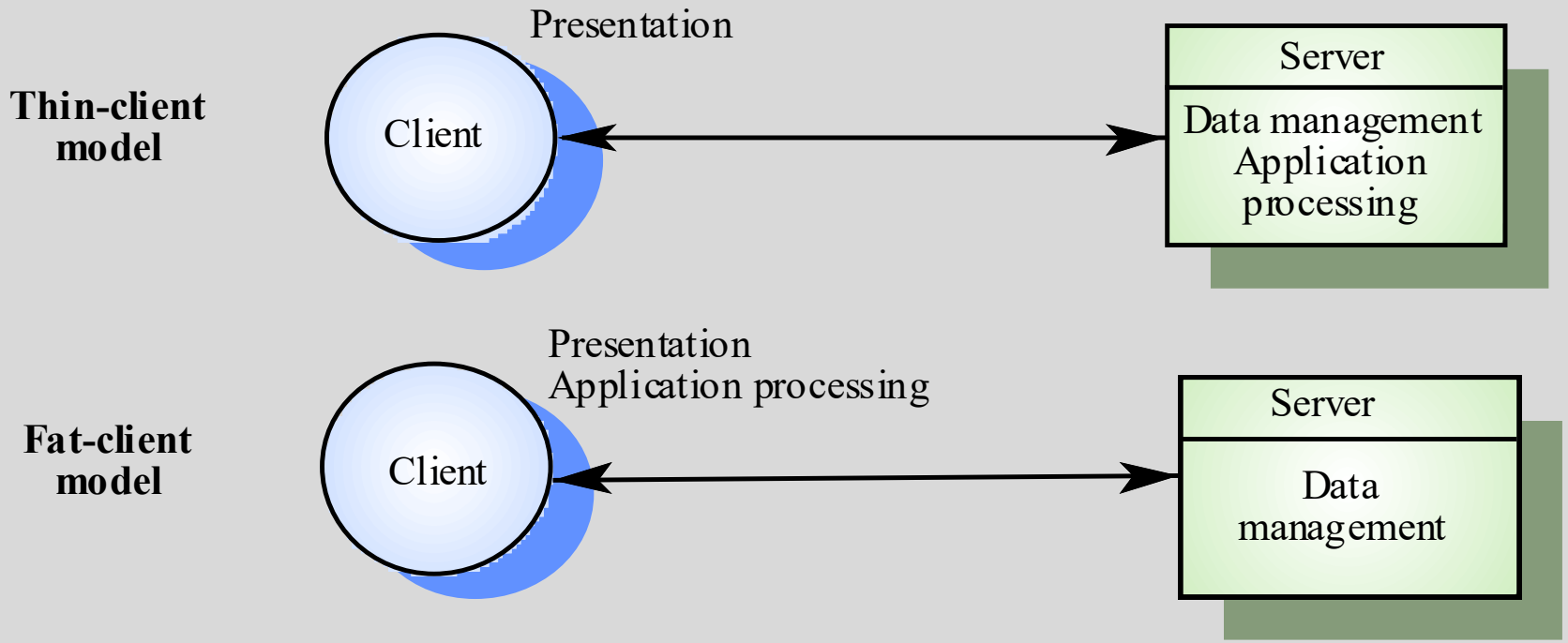
Application Layers



Thin and Fat Clients

- ***Thin-client model***
  - All application processing and data management is carried out on the server.
  - **The client is simply responsible for running the presentation software**.
- ***Fat-client model***
  - Server is only responsible for data management.
  - **The software on the client implements the application logic and the interactions with the system user**.

Thin and Fat Clients



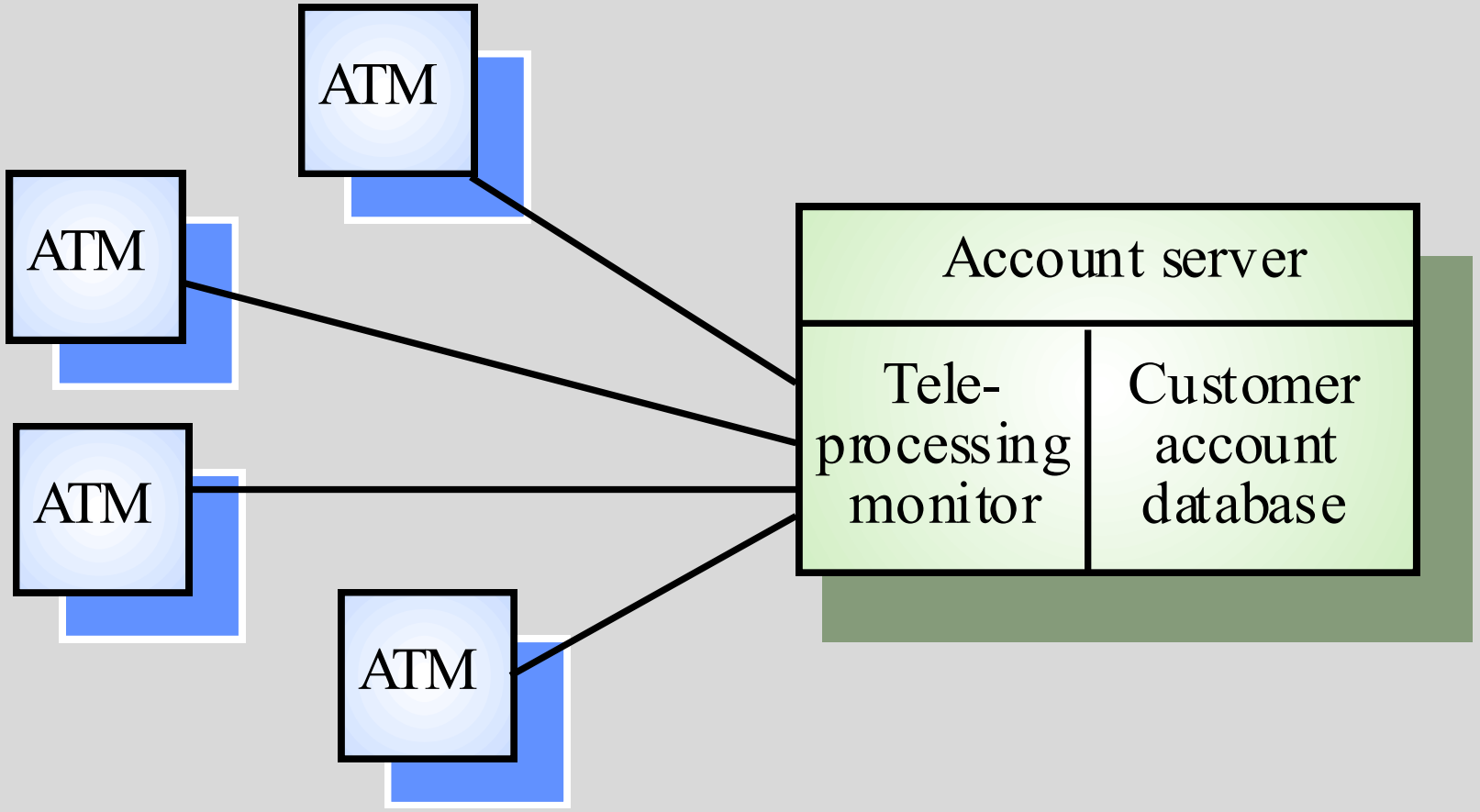
Thin Client Model

- **Used when legacy systems are migrated to client server architectures.**
  - The legacy system acts as a server in its own right with a graphical interface implemented on a client
- A major disadvantage is that it places a heavy processing load on both the server and the network

Fat Client Model

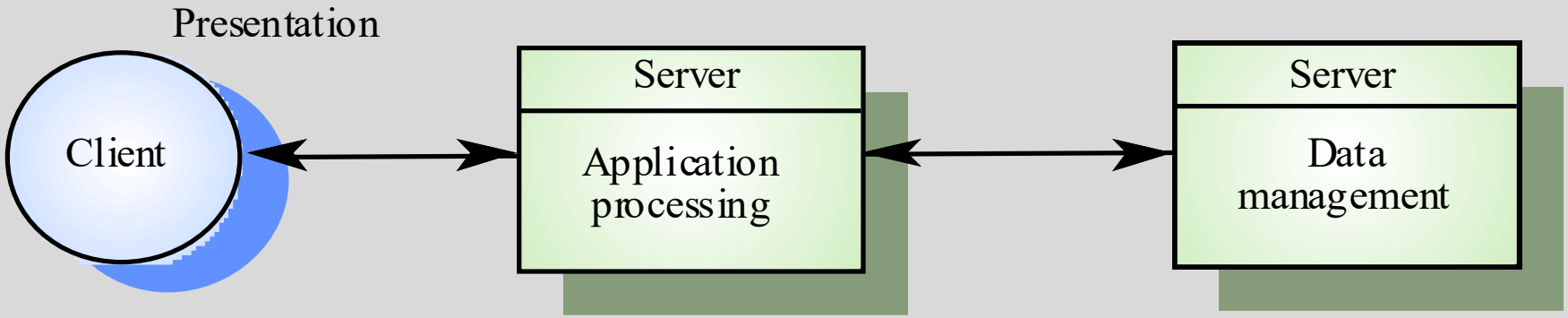
- More processing is delegated to the client
- Application processing is locally executed
- Most suitable for new client-server systems where the capabilities of the client system are known in advance
- More complex than a thin client model especially for management.
- New versions of the application have to be installed on all clients

A Client-Server ATM System

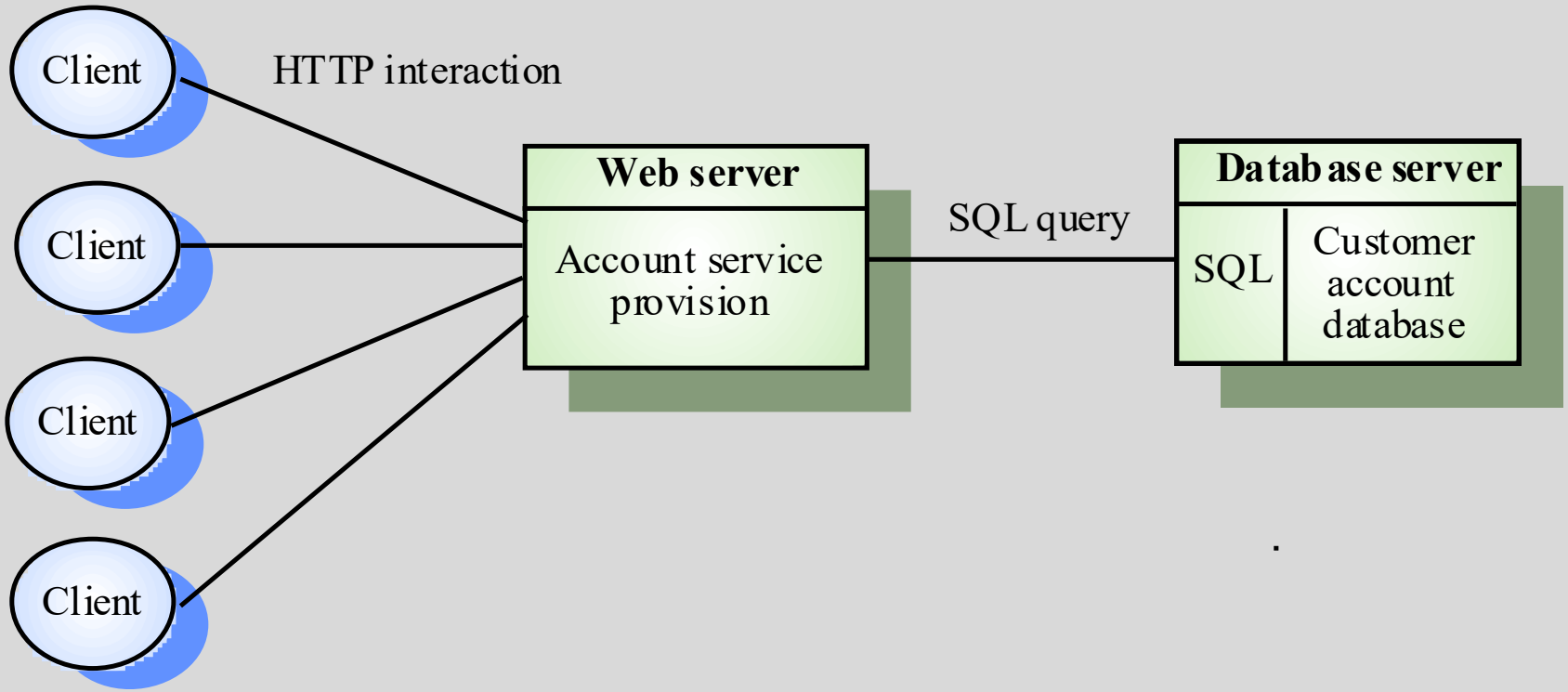


Three-Tier Architectures

- In a 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 to the data management or application processing layers.



An Internet Banking System



Lecture Key Points

- Broadcast Systems and Interrupt based systems handle events and stimuli in different ways
- Modular decomposition can follow an **OO** approach or a **functional (data flow)** approach
- Client-server systems are distributed systems where the system is modelled as a set of services provided by servers to client processes.
- In a client-server system, the user interface always runs on a client and data management is always provided by a shared server.
- Application functionality may be implemented on the client computer or the server.