

Architectural Design

- The design process for identifying the sub-systems making up a system and the framework for sub-system control and communication is **architectural design**.
- The output of this design process is a description of the **software architecture**.

Advantages of explicit architecture

Stakeholder communication

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

System analysis

- Means that analysis of whether the system can meet its non-functional requirements is possible

Large-scale reuse

- The system architecture is often the same for systems with similar requirements and so can the architecture may be reusable across a range of systems

Architecture and system characteristics

- ▶ **Performance** - If performance is a critical requirement, the architecture should be designed to localize critical operations within a small number of subsystems, with as little communication as possible between these sub-systems. **Localise operations to minimise sub-system communication**
- ▶ **Security** - If security is a critical requirement, **Use a layered architecture** with critical assets protected in the inner layers
- ▶ **Safety** - If safety is a critical requirement, the architecture should be designed so that **safety-related operations are all located in either a single sub-system or in a small number of sub-systems**
- ▶ **Availability** - If availability is a critical requirement, the architecture should be designed to include **redundant components** and so that it is possible to replace and update components without stopping the system.
- ▶ **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..

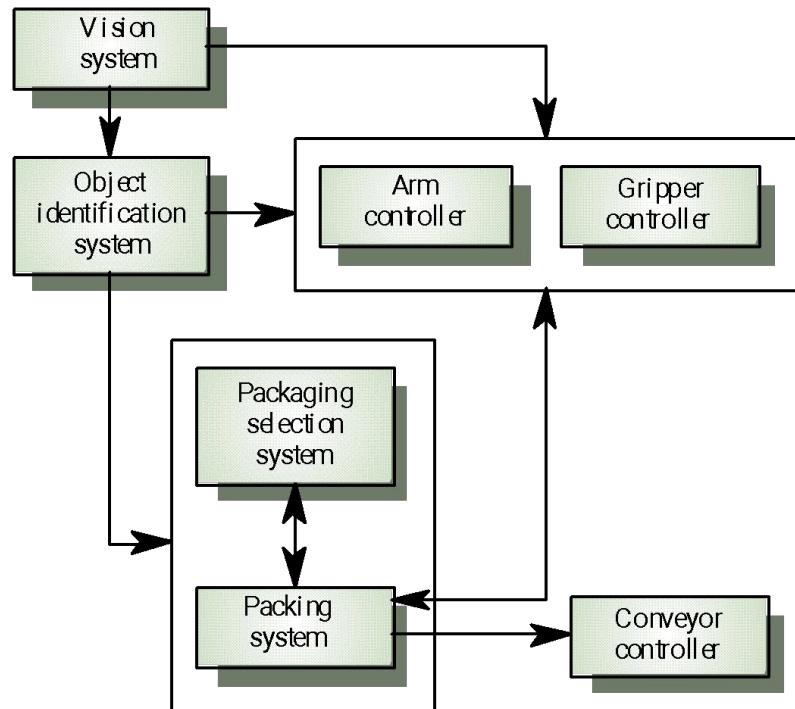


Figure 11.1 is an abstract model of the architecture for a packing robot system that shows the sub-systems that have to be developed.



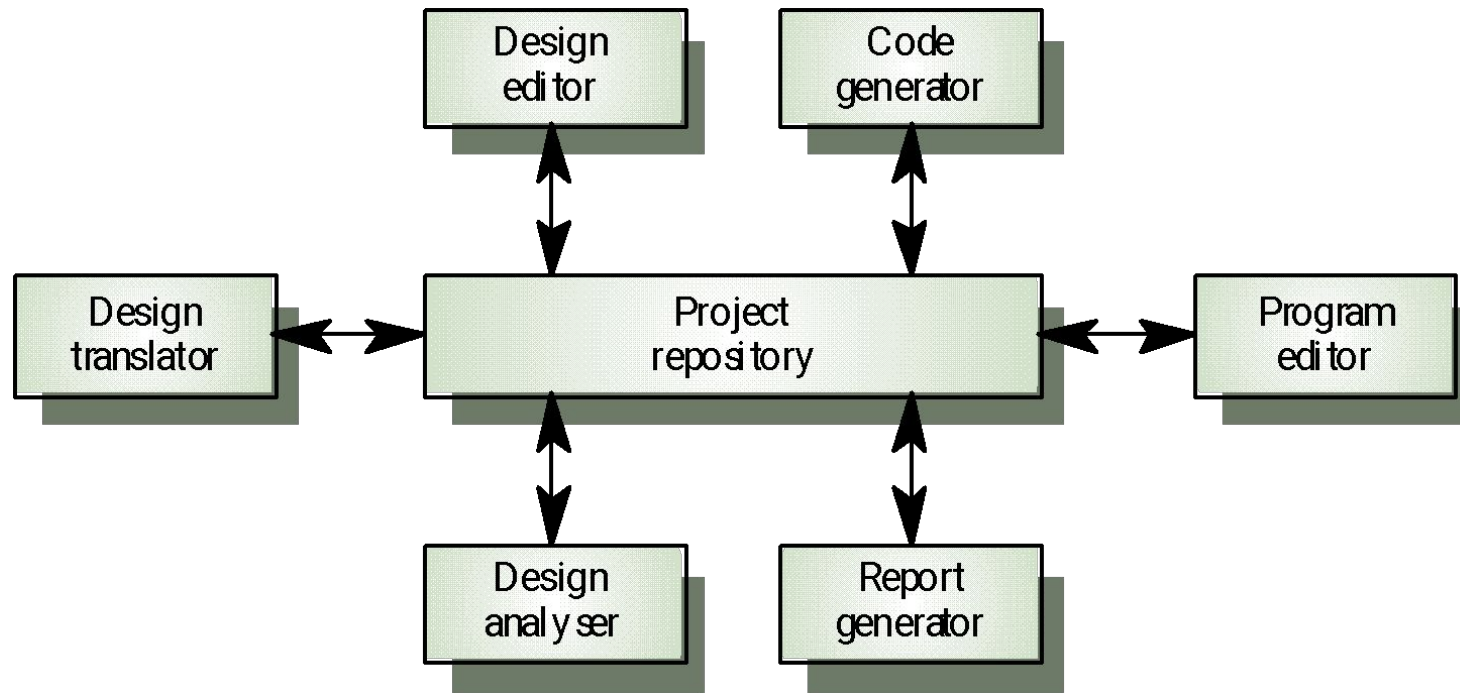
A Packing Robot System

- ▶ This robotic system can pack different kinds of object.
- ▶ It uses a vision sub-system to pick out objects on a conveyor, identify the type of object and select the right kind of packaging.
- ▶ It places packaged objects on another conveyor.



System organisation

- ▶ The software that is built for computer-based systems can exhibit one of these many architectural styles.
- ▶ Three organizational styles that are very widely used are
 1. Data repository style
 2. The client-server model
 3. The layered style



The Repository Model: Figure 11.2 is an example of a CASE toolset architecture based on a shared repository



The repository model

- ▶ Sub-systems making up a system must exchange information so that they can work together effectively.

There are two fundamental ways in which this can be done.

1. Central database that can be accessed by all subsystems and system model based on a shared database is sometimes called a repository model.
2. Each sub-system maintains its own database. Data is interchanged with other sub-systems by passing messages to them.

When large amounts of data are to be shared, the repository model of sharing is the most commonly used.

Repository model

- ▶ Advantages

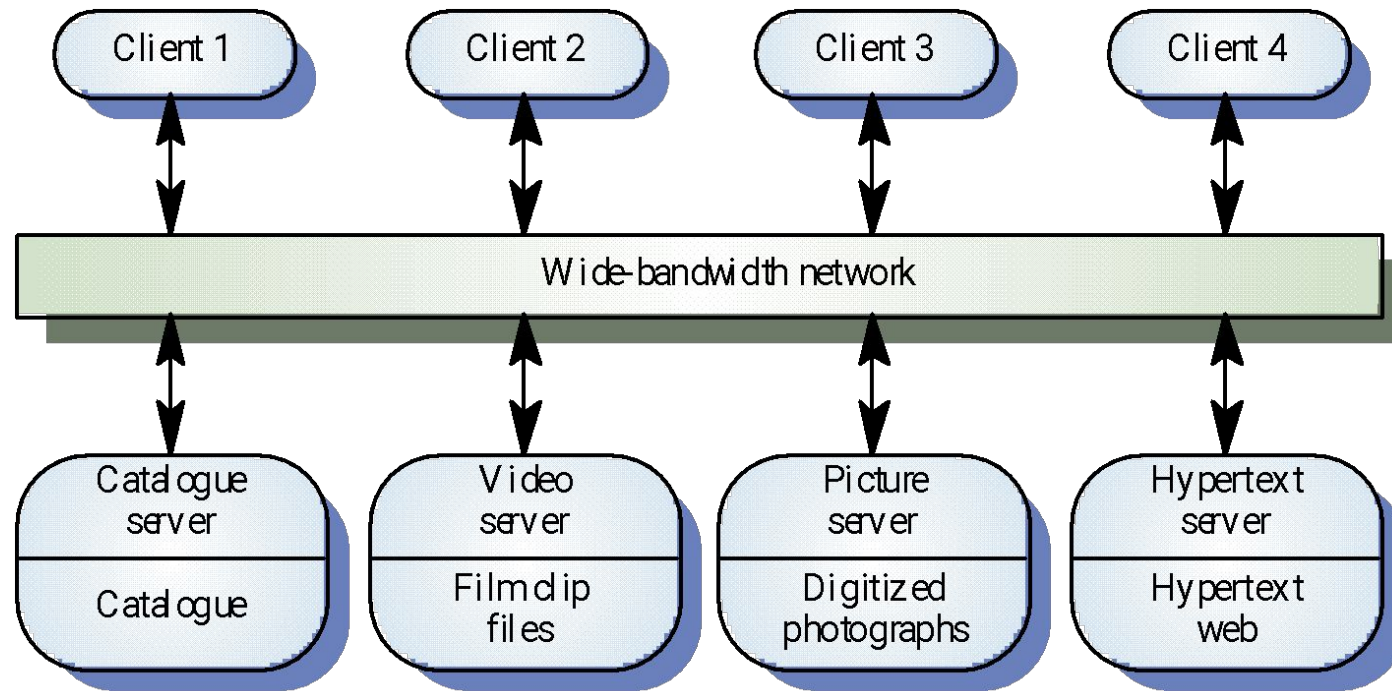
- ▶ Efficient way to share large amounts of data (Centralised)
- ▶ Sub-systems need not be concerned with how data is used by other sub-systems.
- ▶ Sharing model is visible through the repository schema

- ▶ Disadvantages

- ▶ Sub-systems must agree on a repository data model.
- ▶ Data evolution is difficult and expensive
- ▶ Difficult to distribute efficiently

Client-server architecture

- ▶ Distributed system model which shows how data and processing is distributed across a range of components
- ▶ Set of stand-alone servers which provide specific services such as printing, data management, etc.
- ▶ Set of clients which call on these services
- ▶ Network which allows clients to access servers



Film and picture library

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

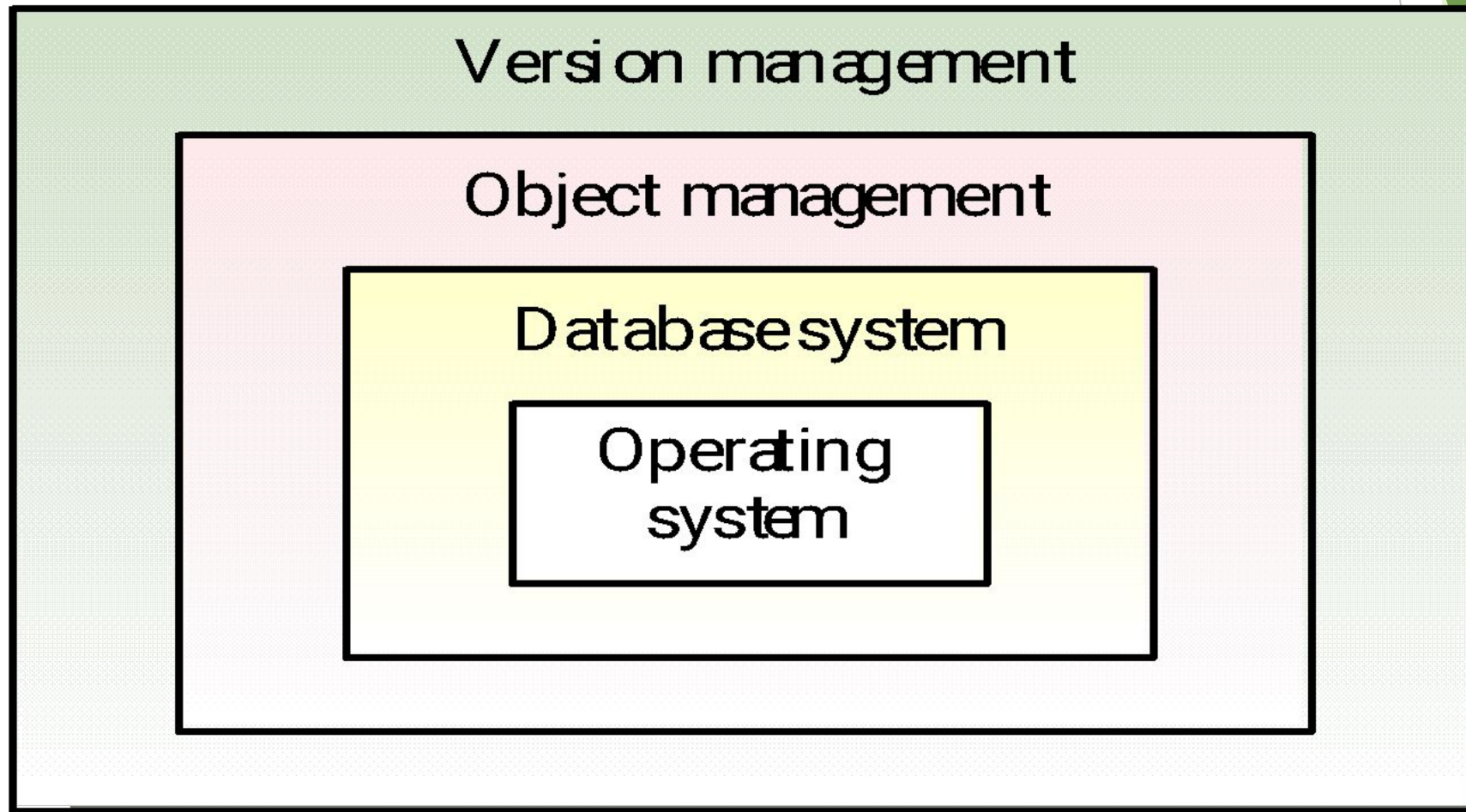
► Disadvantages

- No shared data model so sub-systems use different data 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

Abstract/Layered machine model

- ▶ Used to model the interfacing of sub-systems
- ▶ Organises the system into a set of layers (or abstract machines) each of which provide 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
- ▶ However, often difficult to structure systems in this way

Version management system



1. The primary objective of is to develop a modular program structure and represent the control relationships between modules.

- A) architectural design
- B) object oriented design
- C) function oriented design
- D) interface design

Within model, each sub-system maintains its own database and data is interchanged with other sub-systems by passing message to them.

- A) client server
- B) abstract machine
- C) shared repository
- D) control

The is a distributed system model which show how data and processing is distributed across a range of processors.

- A) client server model
- B) abstract machine model
- C) shared repository model
- D) Layered model

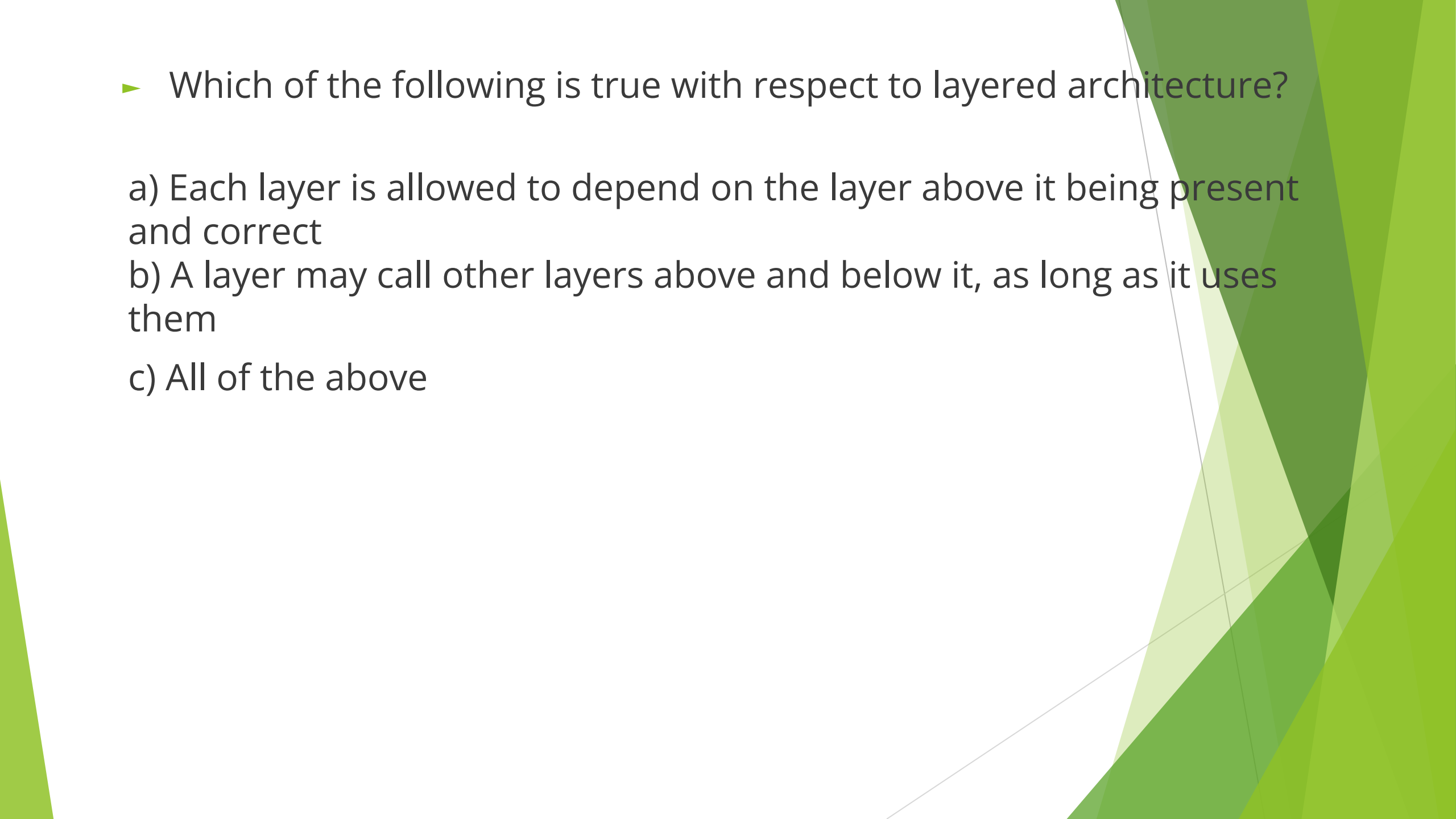
The supports the incremental development of system.

- A) client server approach
- B) abstract machine approach
- C) shared repository approach
- D) layered approach

State whether the following statements about shared repository model are True or False.

- i) It is an efficient way to share large amount of data.
- ii) It is easier to distribute the repository over a number of machines.
- iii) There is no need to transmit data explicitly from one sub-system to another.

- A) i-True, ii-True, iii-False
- B) i-True, ii-False, iii-False
- C) i-True, ii-False, iii-True
- D) i-False, ii-True, iii-True

- 
- ▶ Which of the following is true with respect to layered architecture?
 - a) Each layer is allowed to depend on the layer above it being present and correct
 - b) A layer may call other layers above and below it, as long as it uses them
 - c) All of the above

Modular decomposition styles

- ▶ After an overall system organization has been chosen, you need to make a decision on the approach to be used in decomposing sub-systems into modules
- ▶ Sub-systems are composed of modules and have defined interfaces, which are used for communication with other sub-systems.

Modular decomposition styles

There are two main strategies that you can use when decomposing a sub-system into modules:

1. Object-oriented decomposition where you decompose a system into a set of communicating objects.
2. Function-oriented pipelining where you decompose a system into functional modules that accept input data and transform it into output data.

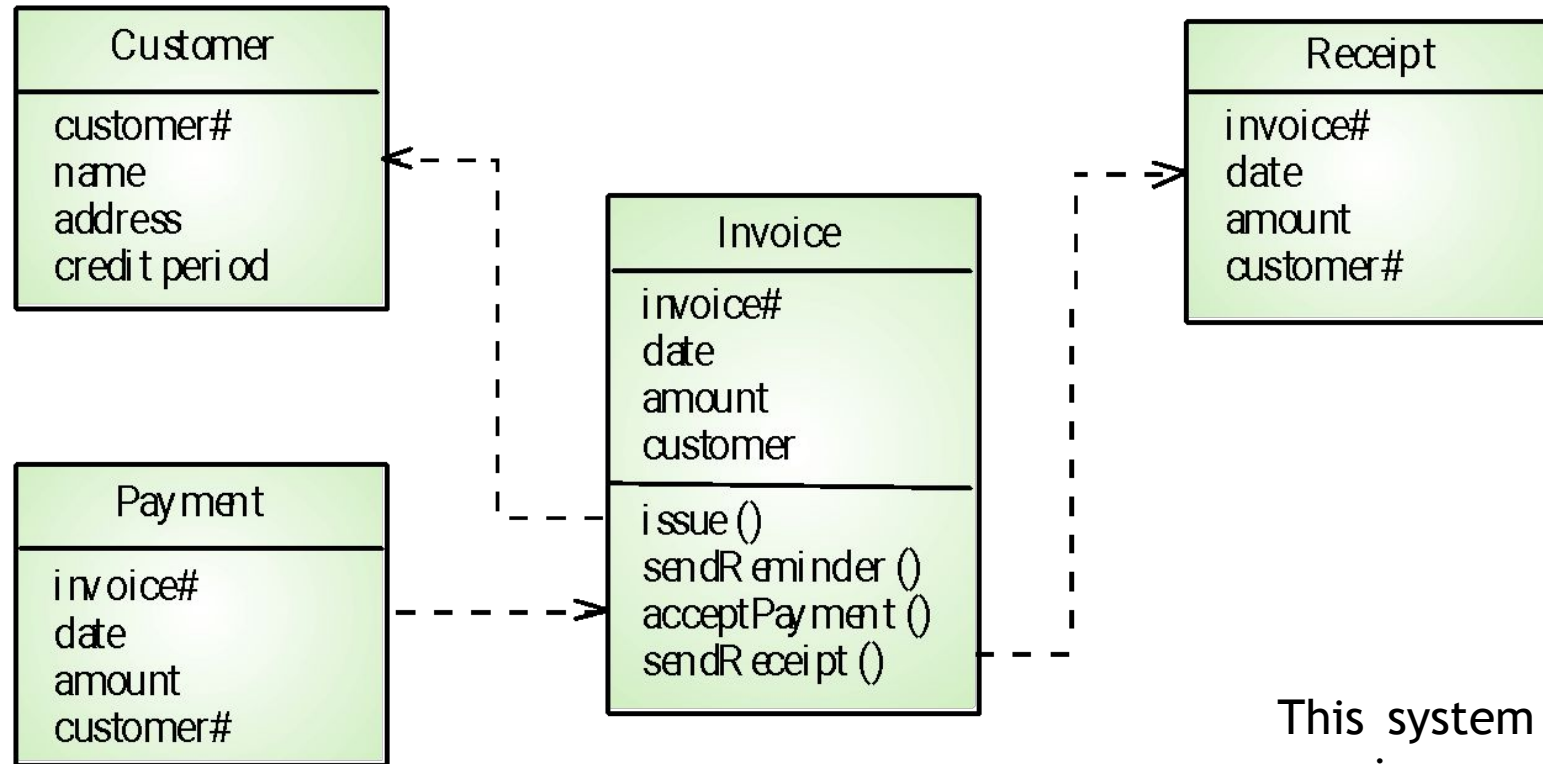
In the object oriented approach, modules are objects with private state and defined operations on that state.

In the pipelining model, modules are functional transformations. In both cases, modules may be implemented as sequential components or as processes.

Object-oriented decomposition

- ▶ An object-oriented, architectural model structures the system into a set of loosely coupled objects with well-defined interfaces.
- ▶ When implemented, objects are created from these classes and some control model used to coordinate object operations.

Invoice processing system

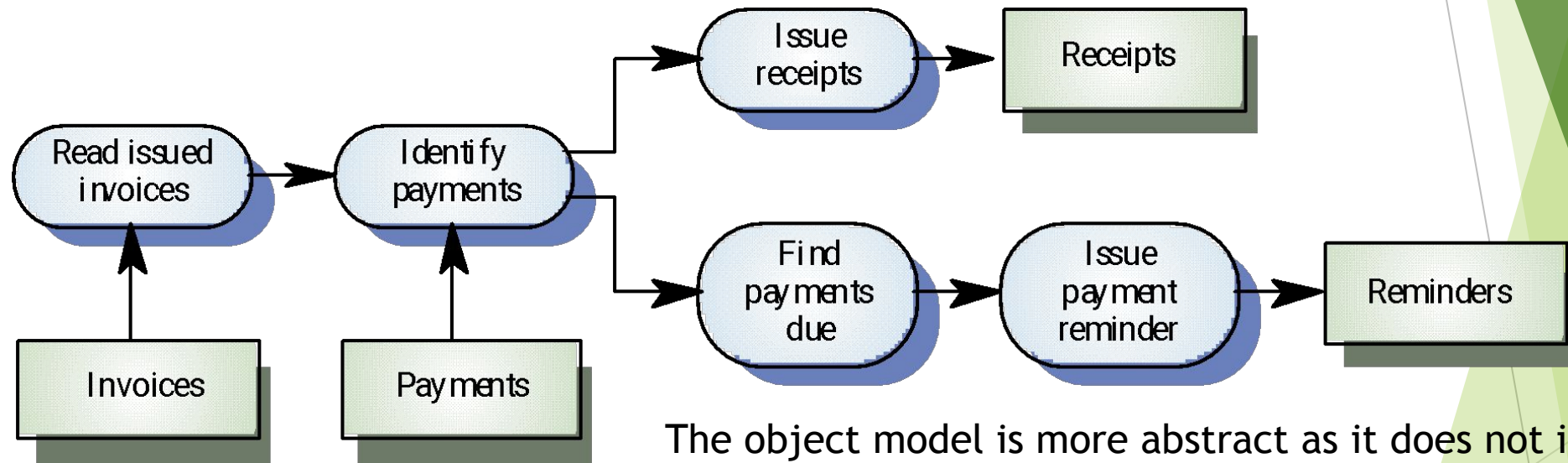


This system can issue invoices to customers, receive payments, and issue receipts for these payments and reminders for unpaid invoices. Dashed arrows indicate that an object uses the attributes or services provided by another object.

Function-oriented pipelining

- ▶ In a function-oriented pipeline or data-flow model, functional transformations process their inputs and produce outputs

Invoice processing system



The object model is more abstract as it does not include information about the sequence of operations.

Think-Pair-Share

► A Case Study on Architectural Styles

Substantiate with reasons for your Answer, Suggest and discuss an appropriate model for the following Systems:

1. An automated ticket issuing system used by passengers at a railway station
2. A Video Conferencing System that allows video, audio and computer data to be visible to several participants at the same time.
3. A Robot floor-cleaner that is intended to clean relatively clear spaces such as corridors. The cleaner must be able to sense walls and other obstructions.

Architectural models

- ▶ Architectural models that may be developed may include:
 1. A static structural model that shows the sub-systems or components that are to be developed as separate units.
 2. A dynamic process model that shows how the system is organized into processes at run-time. This may be different from the static model.
 3. An interface model that defines the services offered by each sub-system through its public interface.
 4. Relationship models that shows relationships, such as data flow, between the sub-systems.
 5. A distribution model that shows how sub-systems may be distributed across computers.

Control styles

- ▶ Are concerned with the control flow between sub-systems.

There are two generic control styles that are used in software systems:

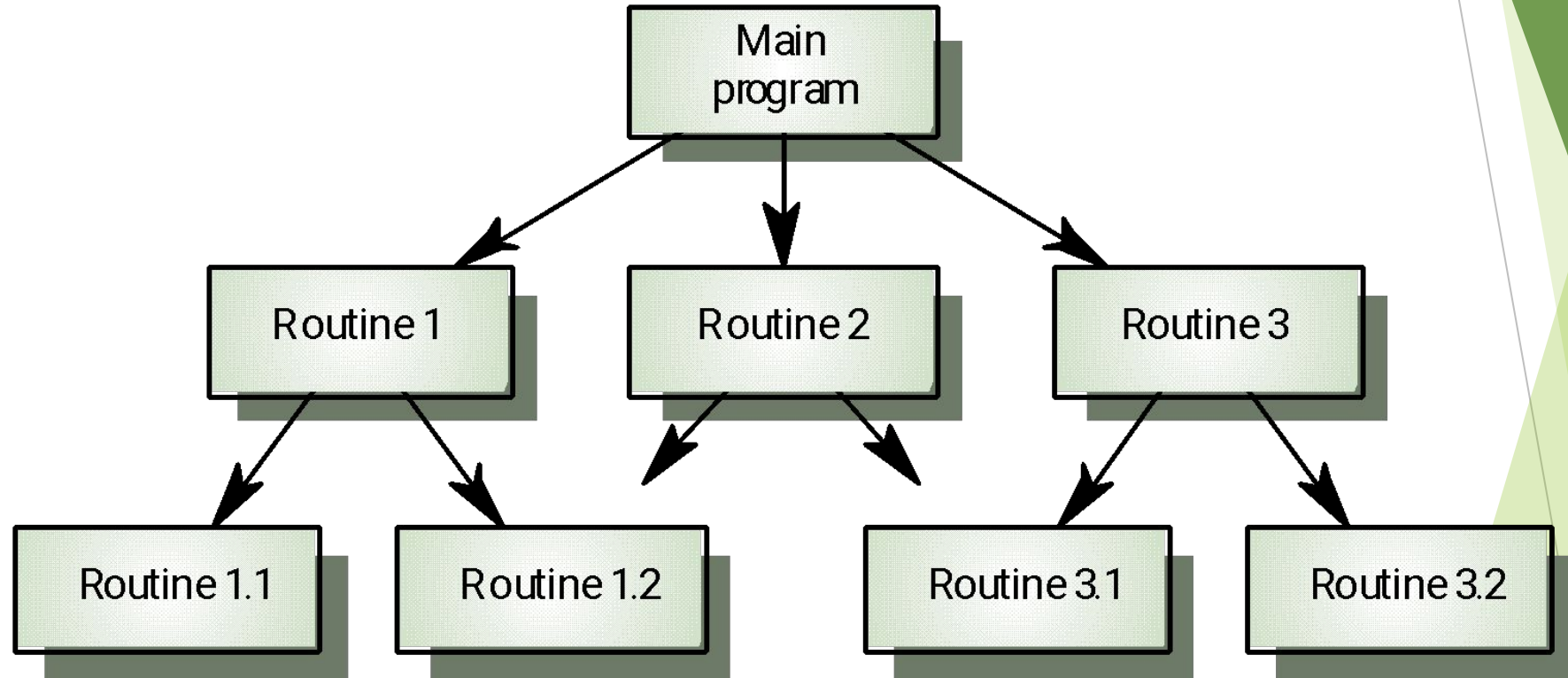
- ▶ Centralised control
 - ▶ One sub-system has overall responsibility for control and starts and stops other sub-systems
- ▶ Event-based control
 - ▶ Each sub-system can respond to externally generated events from other sub-systems or the system's environment

Centralised control

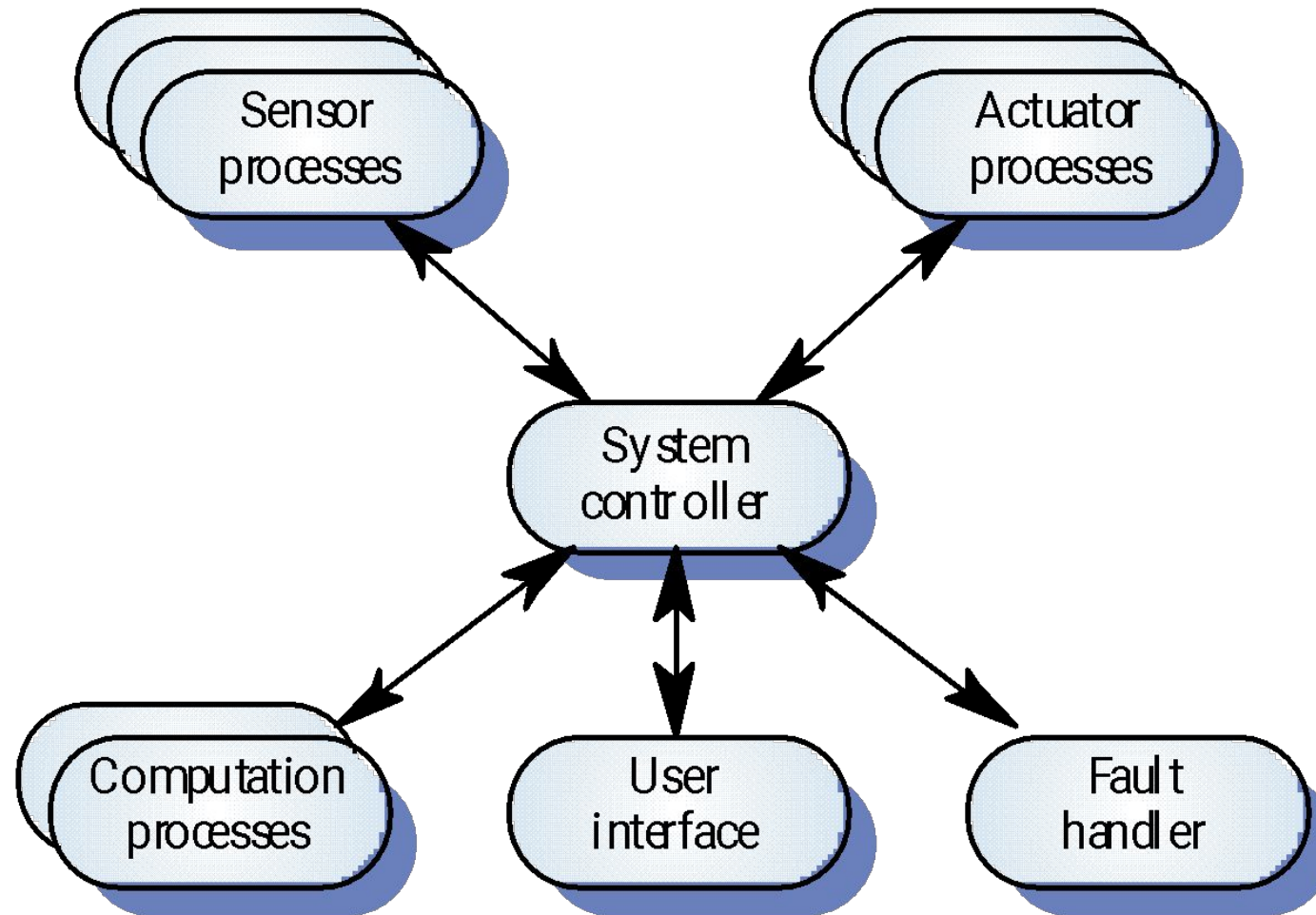
Centralized control models fall into two classes, depending on whether the controlled sub-systems execute sequentially or in parallel.

- ▶ Call-return model
 - ▶ Top-down subroutine model where control starts at the top of a subroutine hierarchy and moves downwards. Applicable to sequential systems
- ▶ Manager model
 - ▶ Applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes. Can be implemented in sequential systems as a case statement

Call-return model



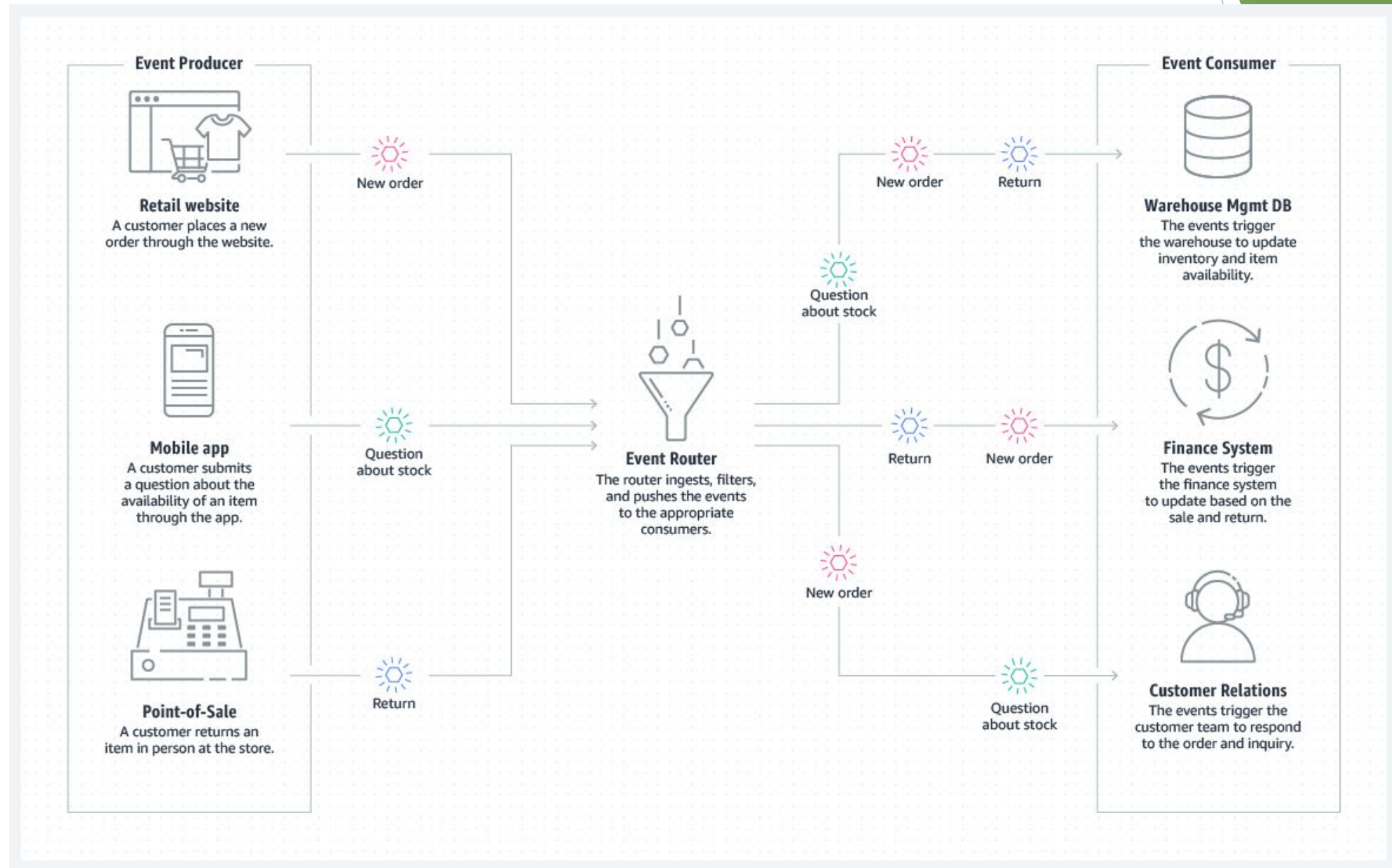
Real-time system control - Manager Model



Event-driven systems

- ▶ Driven by externally generated events where the timing of the event is out with the control of the sub-systems which process the event
- ▶ 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
- ▶ Other event driven models include spreadsheets and production systems

Event Driven control Systems



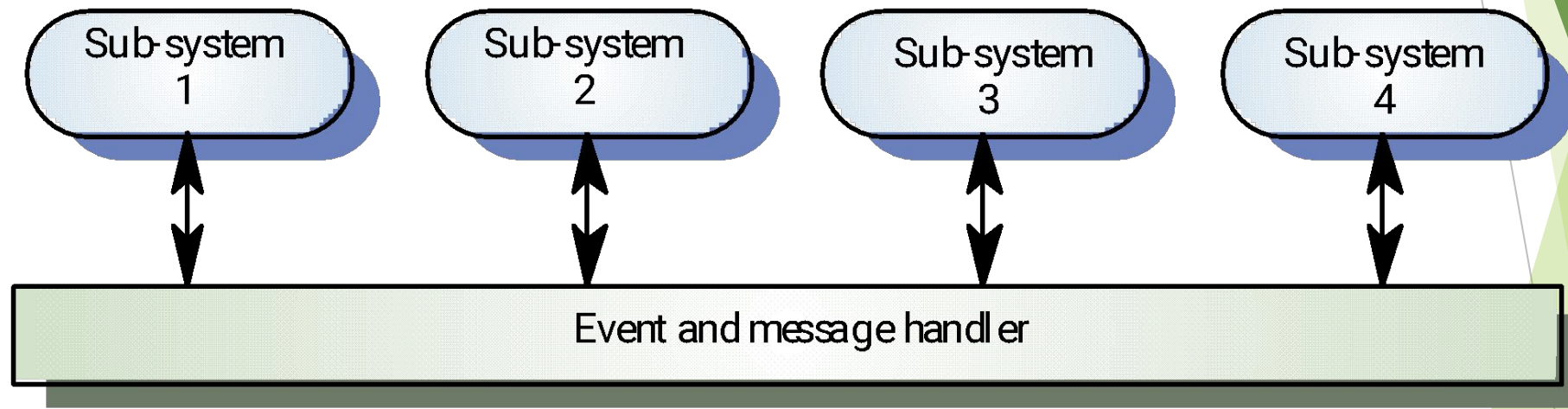
Event-driven systems

- ▶ Driven by externally generated events where the timing of the event is outwith the control of the sub-systems which process the event
- ▶ **Amazon EventBridge** is recommended when you want to build an application that reacts to events from SaaS applications, AWS services, or custom applications.
- ▶ **Amazon SNS** is recommended when you want to build an application that reacts to high throughput and low latency events published by other applications, microservices, or AWS services, or for applications that need very high fanout (thousands or millions of endpoints).

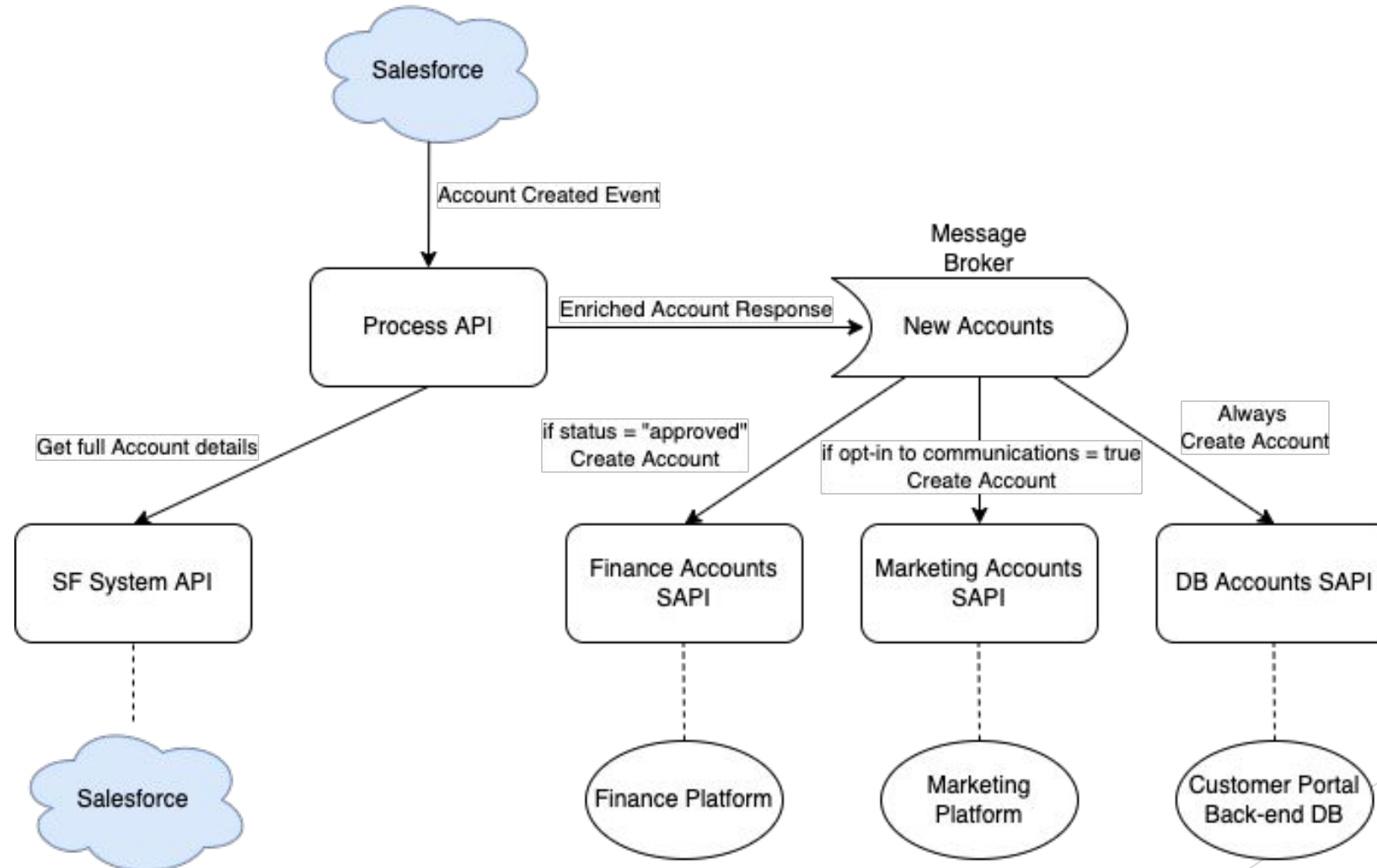
Broadcast model

- ▶ 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



Broadcasting Control in Event Driven Architecture



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

