Software Architectural Design

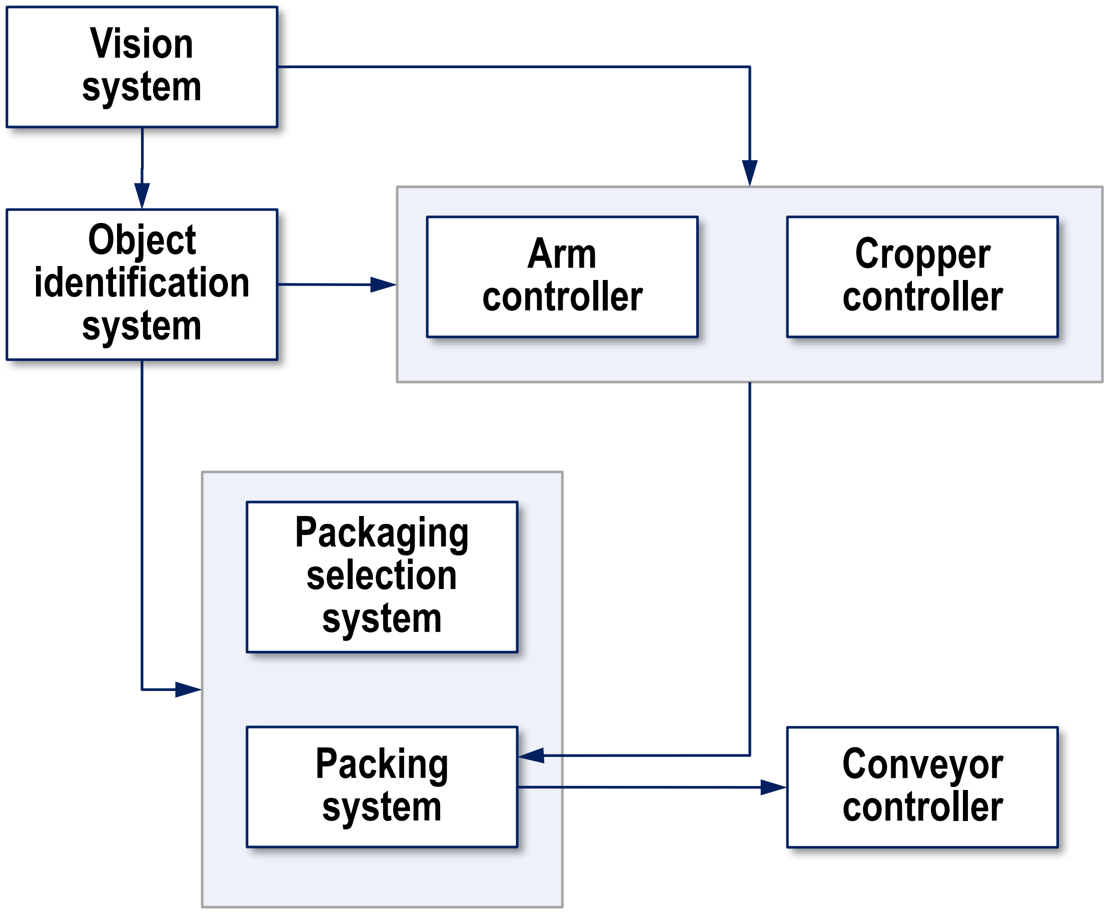
•Chap6

Software architecture

- 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.

예) The architecture of a packing robot control system



Advantages of explicit architecture

- **Stakeholder communication**
  - Architecture may be used as a focus of discussion by system stakeholders.
- **System analysis**
  - Means that analysis of whether the system can meet its non-functional requirements is possible.
- **Large-scale reuse**
  - The architecture may be reusable across a range of systems
  - Product-line architectures may be developed.

Architectural patterns

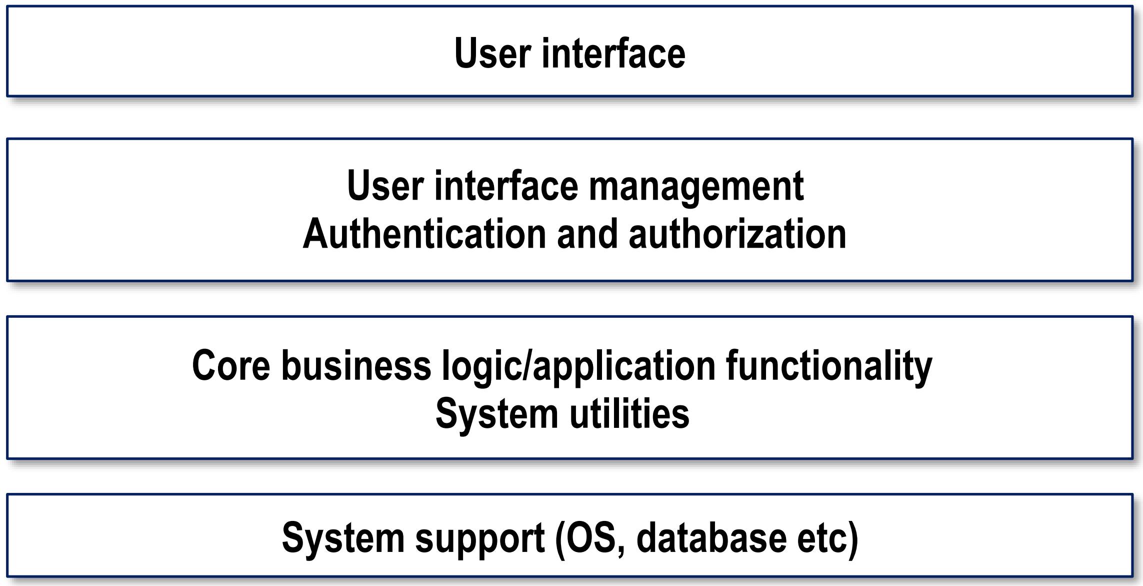
- **Layered architecture**
- **Repository architecture**
- **Client-server architecture**
- **Pipe and filter architecture**
- **Model-View-Controller (MVC) architecture**

Layered architecture

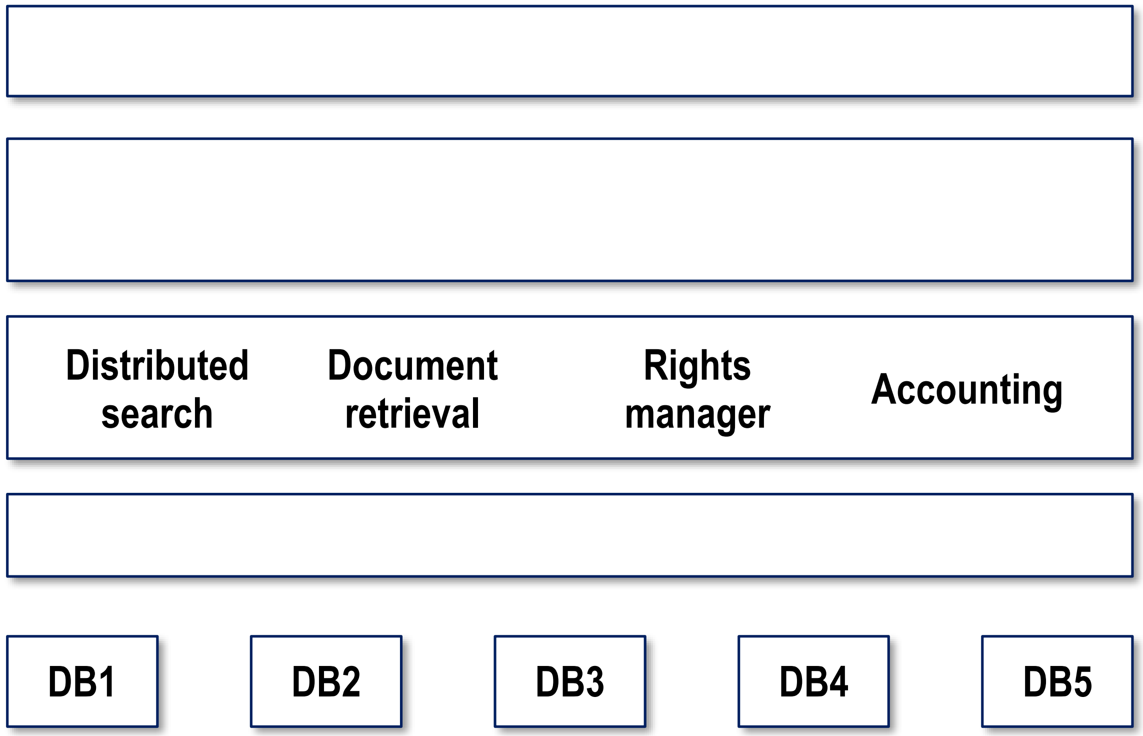
- **Used to model the interfacing of sub-systems.**
- **Organises the system into a set of layers each of which provide a set of services.**
- **Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.**

- **When used**
  - Used when building new facilities on top of existing systems

Generic layered architecture


User interface
User interface management
Authentication and authorization
Core business logic/application functionality
System utilities
System support (OS, database etc)

예) The architecture of the LIBSYS system



- **Advantages**
  - Allows replacement of entire layers so long as the interface is maintained. Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.
- **Disadvantages**
  - In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it.
  - Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

Repository architecture

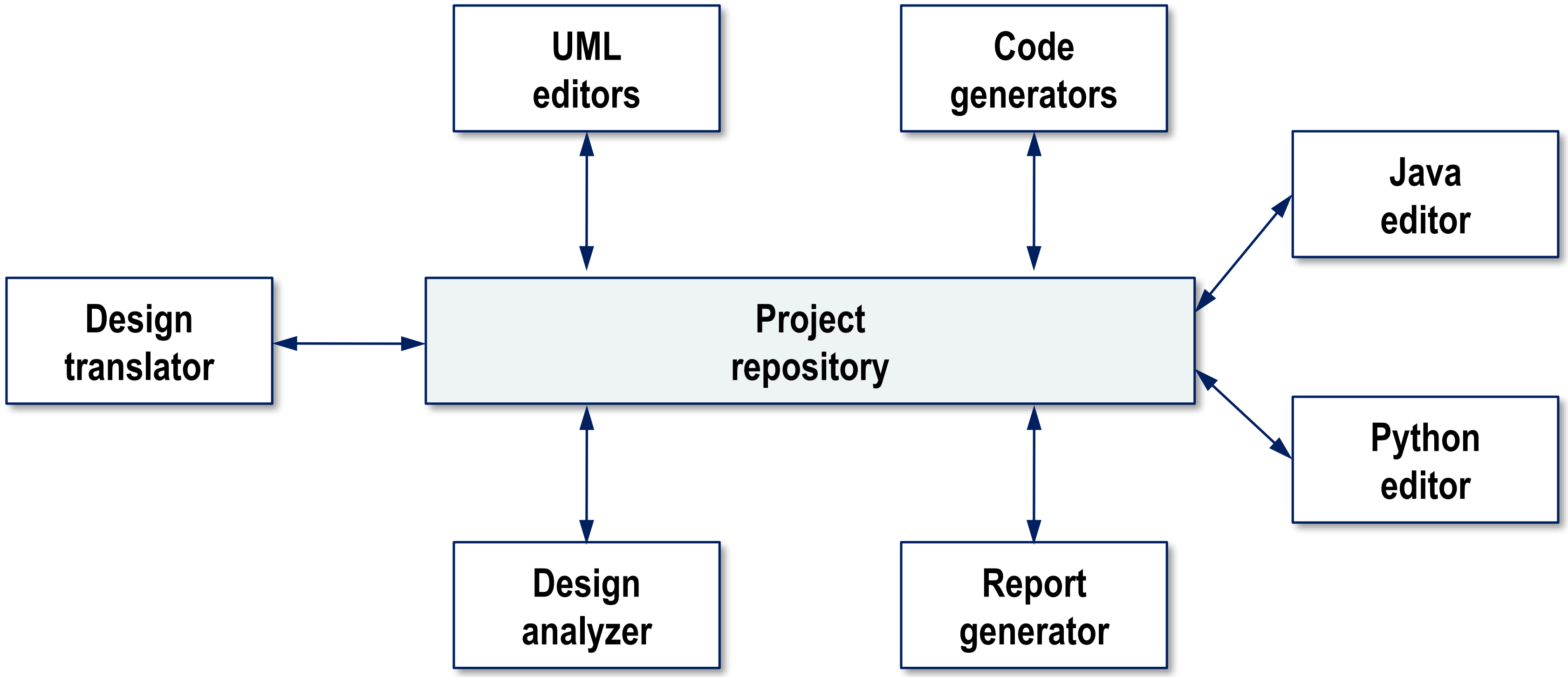
- **Sub-systems must exchange data. This may be done in two ways:**
  - Shared data is held in a central database or repository and may be accessed by all sub-systems;
  - Each sub-system maintains its own database and passes data explicitly to other sub-systems.

- **When large amounts of data are to be shared, the repository model of sharing is most commonly used and this is an efficient data sharing mechanism.**

- **When used**
  - Used when you have a system in which large volumes of information are generated that has to be stored for a long time.

예) A repository architecture for an IDE

Integrated Development Environment 통합개발환경

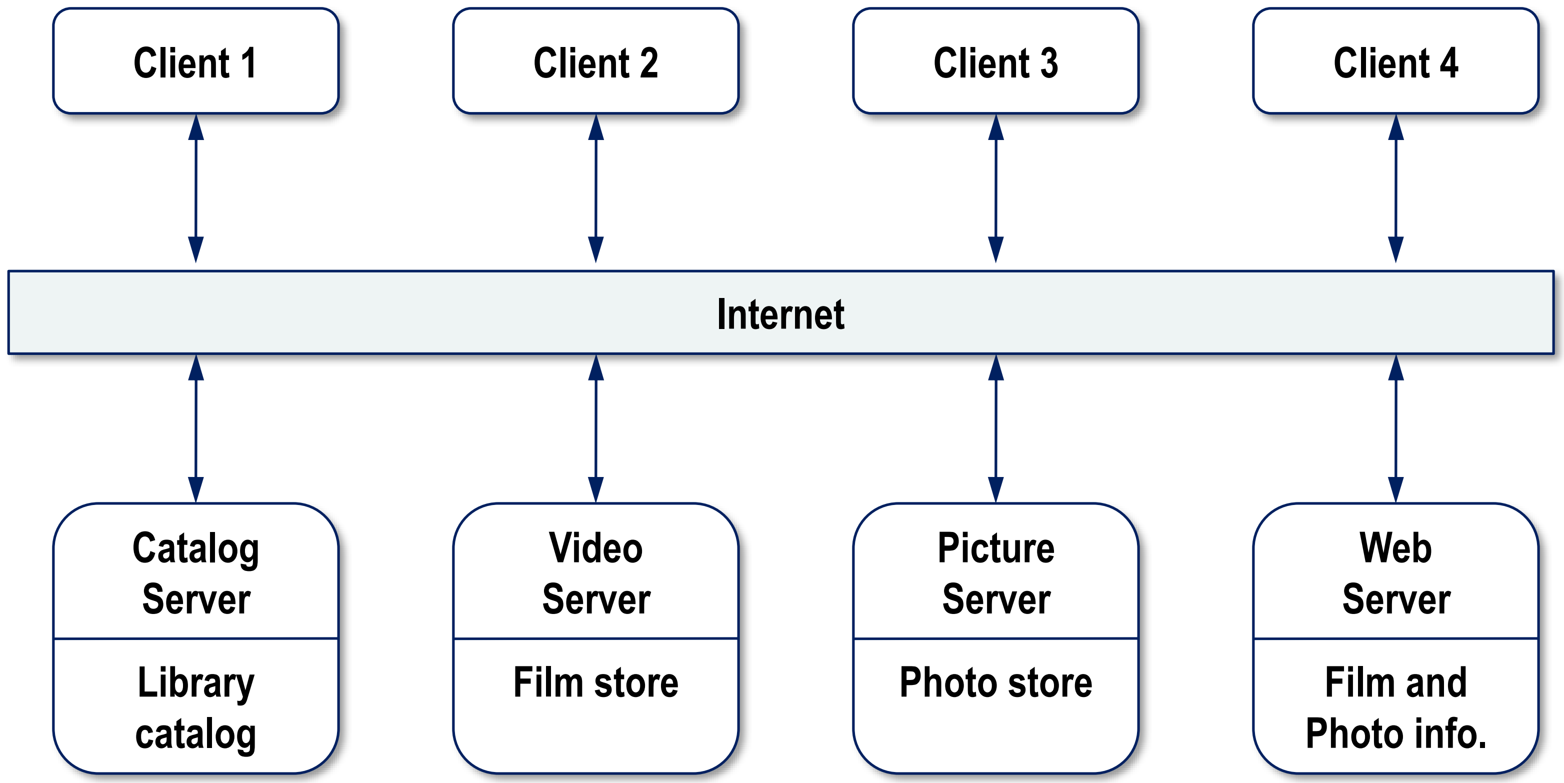


- **Advantages**
  - Components can be independent. They do not need to know of the existence of other components.
  - Changes made by one component can be propagated to all components.
  - All data can be managed consistently (e.g., backups done at the same time) as it is all in one place.
- **Disadvantages**
  - 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 the repository.
  - Distributing the repository across several computers may be difficult.

Client-server architecture

- **Distributed system model which shows how data and processing are 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.**
- **When used**
  - Used when function and data in a shared database has to be accessed from a range of locations.

A client–server architecture for a film library

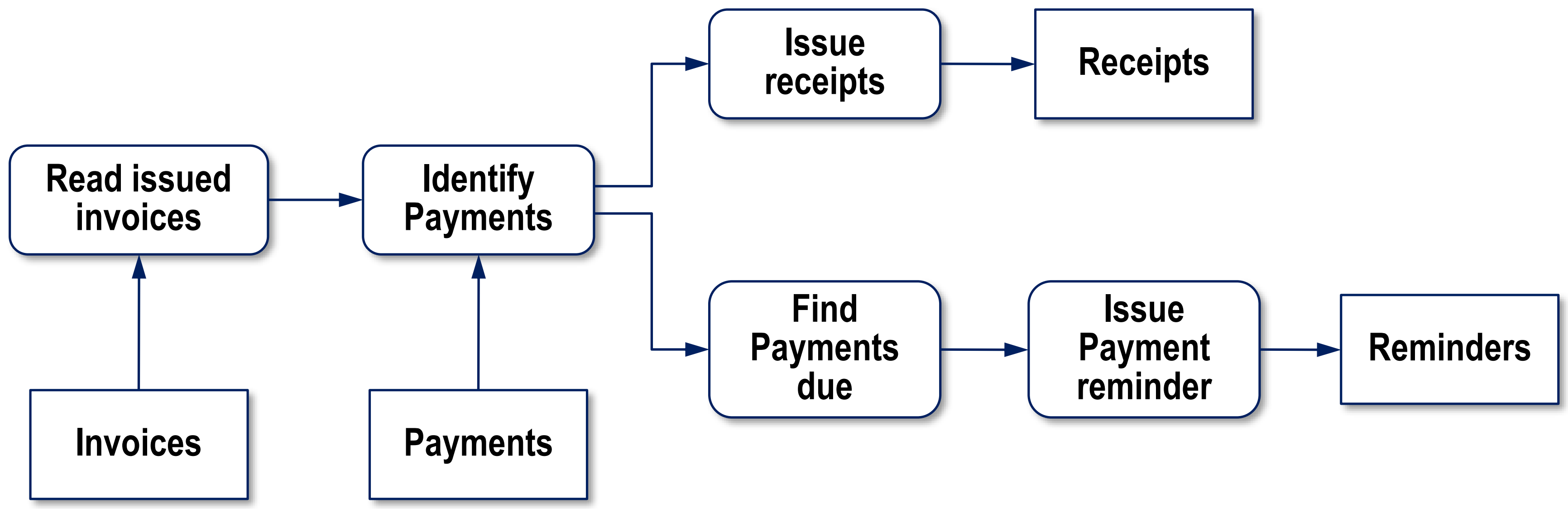


- **Advantages**
  - Servers can be distributed across a network. General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.
- **Disadvantages**
  - Each service is a single point of failure so susceptible to denial of service attacks or server failure.
  - Performance may be unpredictable because it depends on the network as well as the system.
  - May be management problems **if** servers are owned by different organizations.

Pipe and filter architecture

- **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.**
- **When used**
  - Commonly used in data processing applications (both batch- and transaction-based) where inputs are processed in separate stages to generate related outputs.

예) the pipe and filter architecture

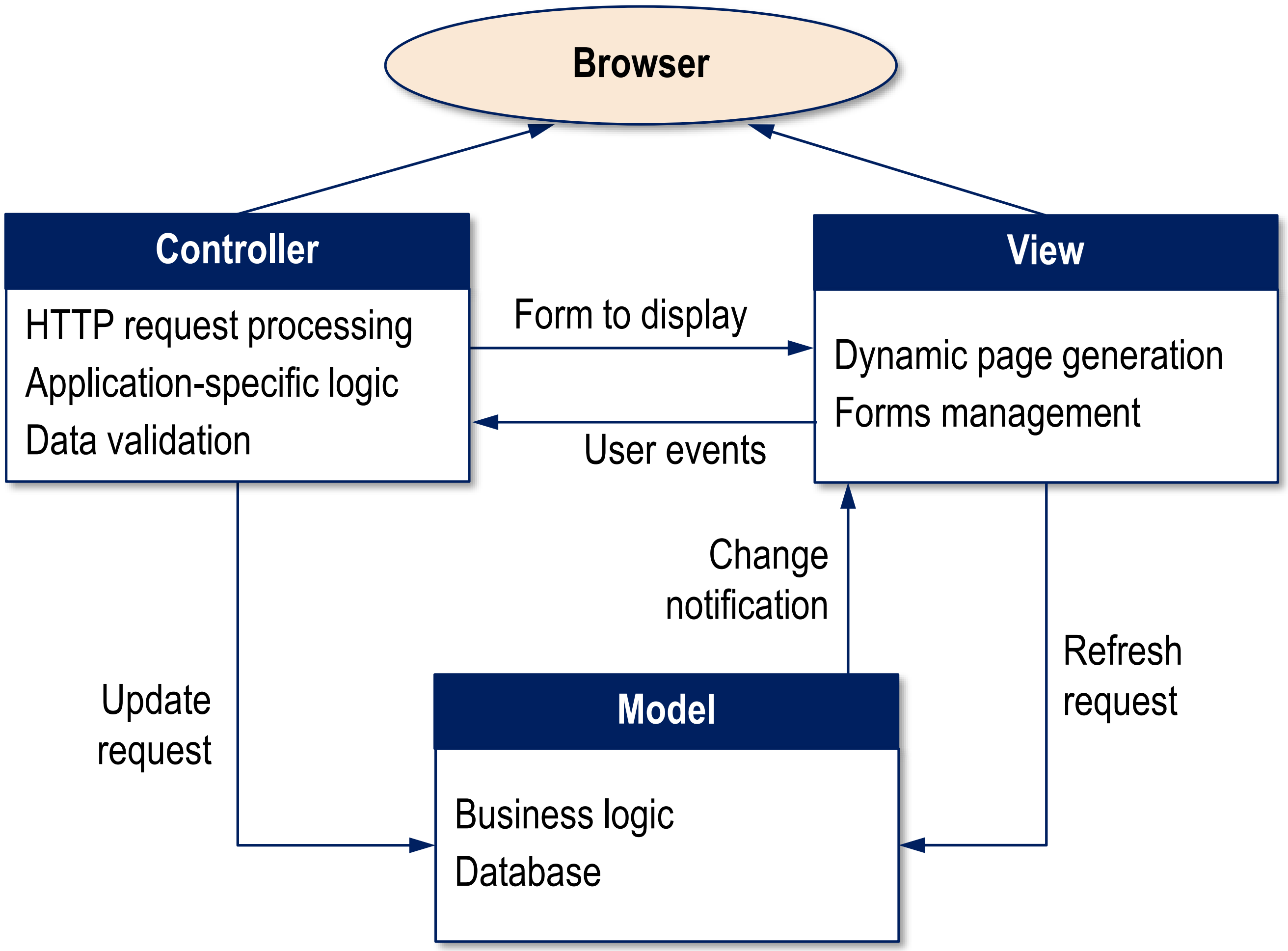


- **Advantages**
  - Workflow style matches the structure of many business processes.
  - Can be implemented as either a sequential or concurrent system.
- **Disadvantages**
  - The format for data transfer has to be agreed upon between communicating transformations.
  - Each transformation must parse its input and unparse its output to the agreed form. This increases system overhead and may mean that it is impossible to reuse functional transformations that use incompatible data structures.

The Model-View-Controller (MVC) architecture

- **Separates presentation and interaction from the system data.**
- **The system is structured into three logical components that interact with each other.**
  - The **Model component** manages the system data and associated operations on that data.
  - The **View component** defines and manages how the data is presented to the user.
  - The **Controller component** manages user interaction (e.g., key presses, mouse clicks, etc.) and passes these interactions to the View and the Model.
- **When used**
  - Used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.

예) Web application architecture

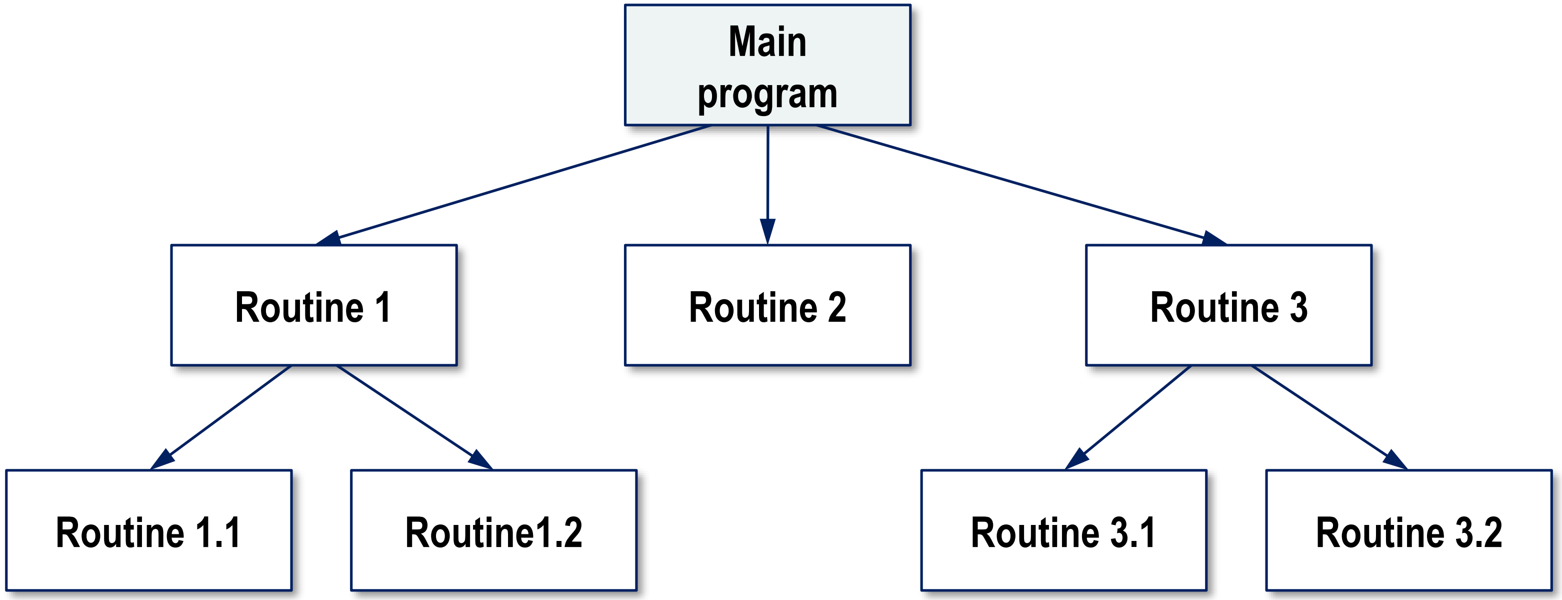


- **Advantages**
  - Allows the data to **change independently** of its representation and vice versa.
  - Supports presentation of **the same data in different ways**
- **Disadvantages**
  - Can involve additional code and code complexity when the data model and interactions are simple.

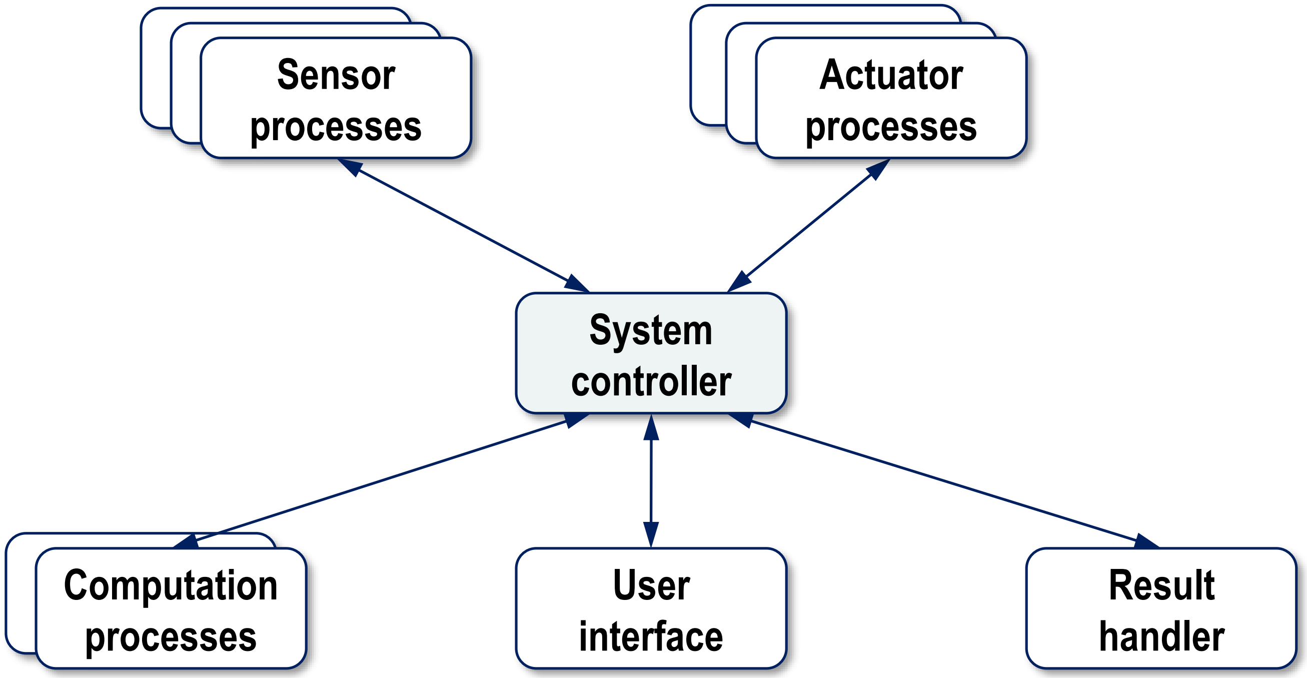
Control styles

- **Centralised control: One sub-system has overall responsibility for control and starts and stops other sub-systems.**
  - Call-return model
  - Manager model
- **Event-based control: Each sub-system can respond to externally generated events from other sub-systems.**
  - Broadcast models.
  - Interrupt-driven models.

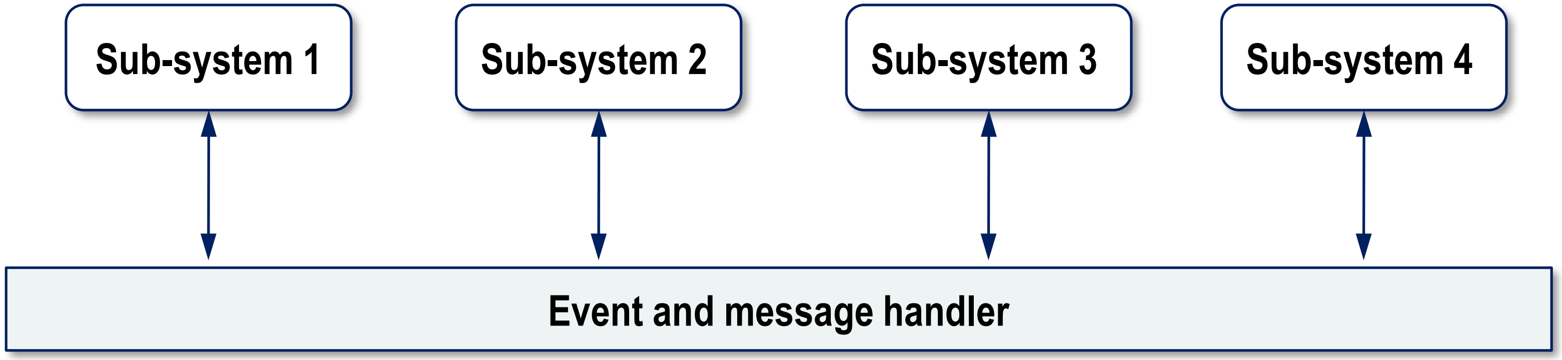
- **Call-return model**
  - Top-down subroutine model where control starts at the top of a subroutine hierarchy and moves downwards. Applicable to sequential systems.



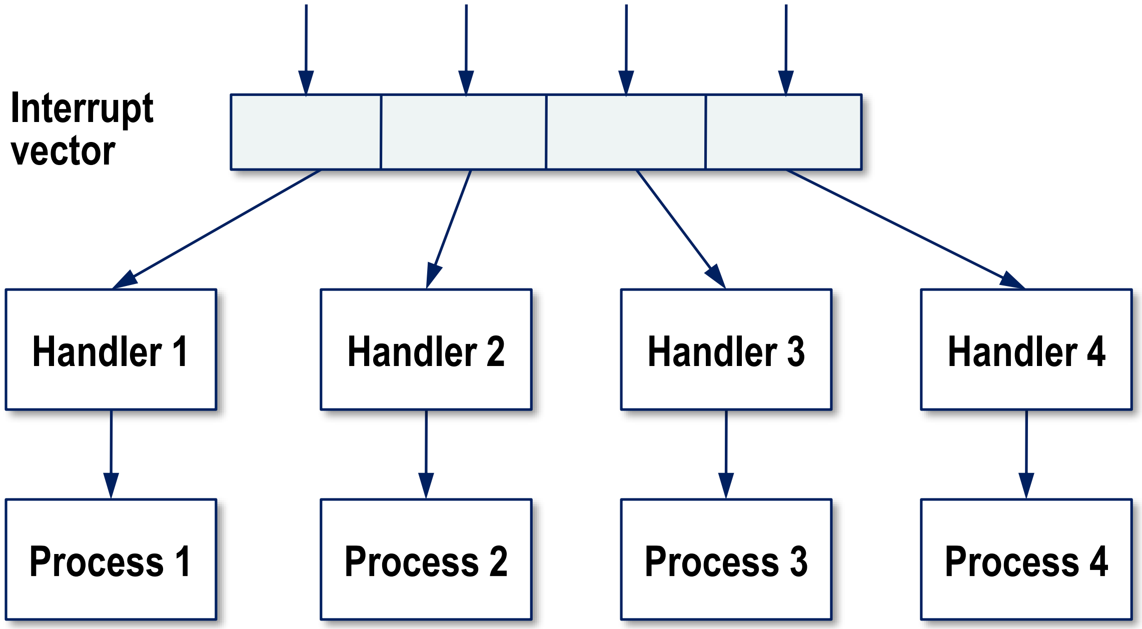
- **Manager model**
  - Applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.



- **Broadcast models.**
  - An event is broadcast to all sub-systems.



- **Interrupt-driven models.**
  - Used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing.


Interrupt vector
Handler 1
Handler 2
Handler 3
Handler 4
Process 1
Process 2
Process 3
Process 4