Sommerville Chapter 6
The High-Level Structure of a Software Intensive System

Architectural Design

Architectural Parallels

- Architects are the technical interface between the customer and the contractor building the system
- A bad architectural design for a building cannot be rescued by good construction
  - The same is true for software
- There are specialist types of building and software architects
- There are schools or styles of building and software architecture

Architectural Styles

What is Architecture?

- [Garlan & Perry]: Software architecture is "the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time."

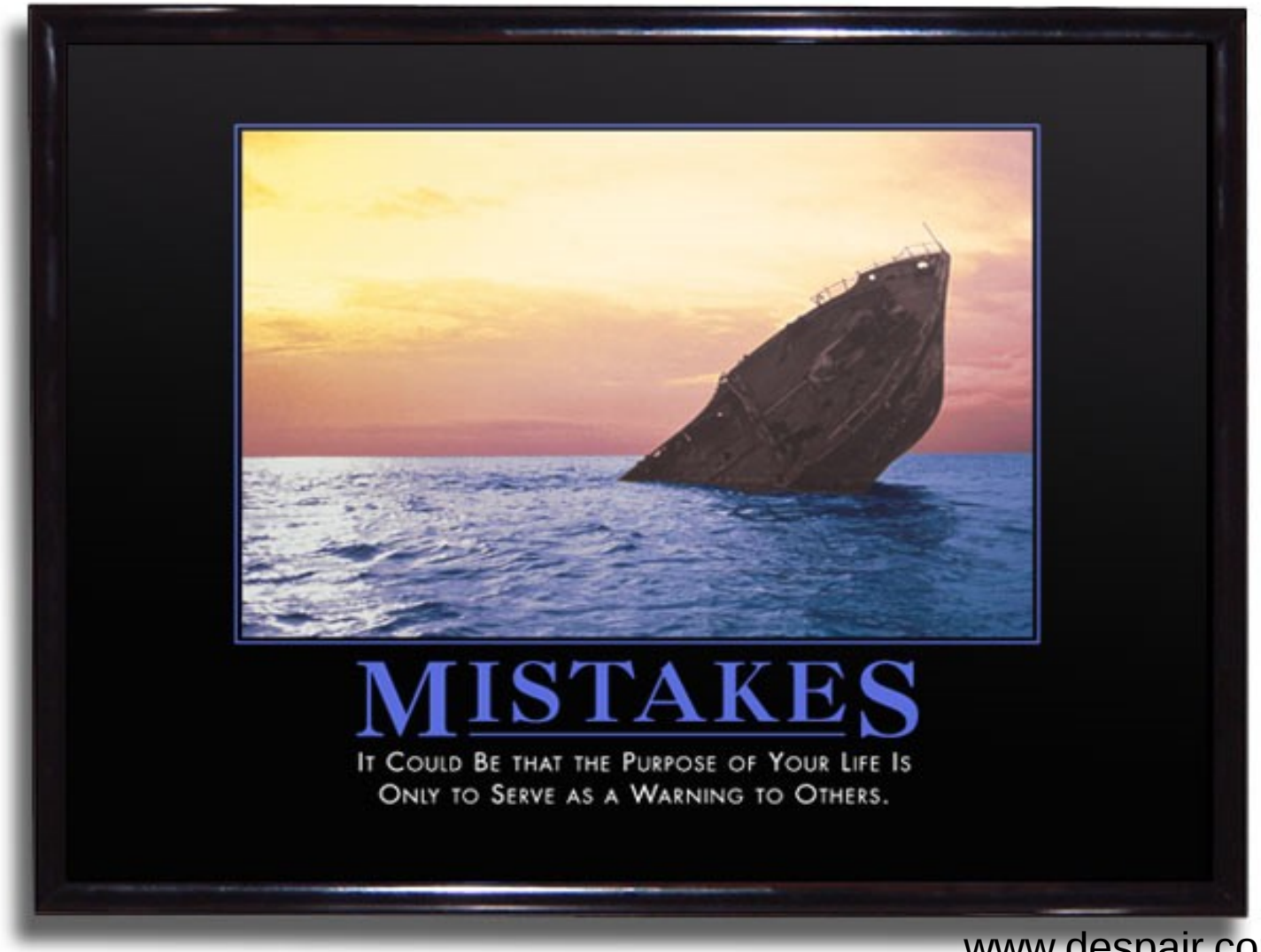
- [Bass, Clements, and Kazman]: The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

- [IEEE 1471]: Software architecture is the **fundamental organization** of a system, embodied in its **components**, their **relationships** to one another and the environment, and the **principles** governing its design and evolution.

Why Should You Care?

- Good Things are Well Architected
  - Internet
  - World Wide Web
  - Airplanes
  - ATMs
- Good Architecture is hard!

Why Should You Care?



www.despair.com

What is Architecture informally?

- Software architecture is primarily concerned with **partitioning** large systems into smaller ones that can be created separately, that individually have business value, and that can be straightforwardly integrated with one another and with existing systems.

Mike Whalen

Architectural Design Process

- System structuring
  - The system is decomposed into several principal sub-systems and communications between these sub-systems are identified
- Control modeling
  - 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 whose operation is independent of the services provided by other sub-systems
- A module is a system component that provides services to other components but would not normally be considered as a separate system

Architectural Qualities

Performance

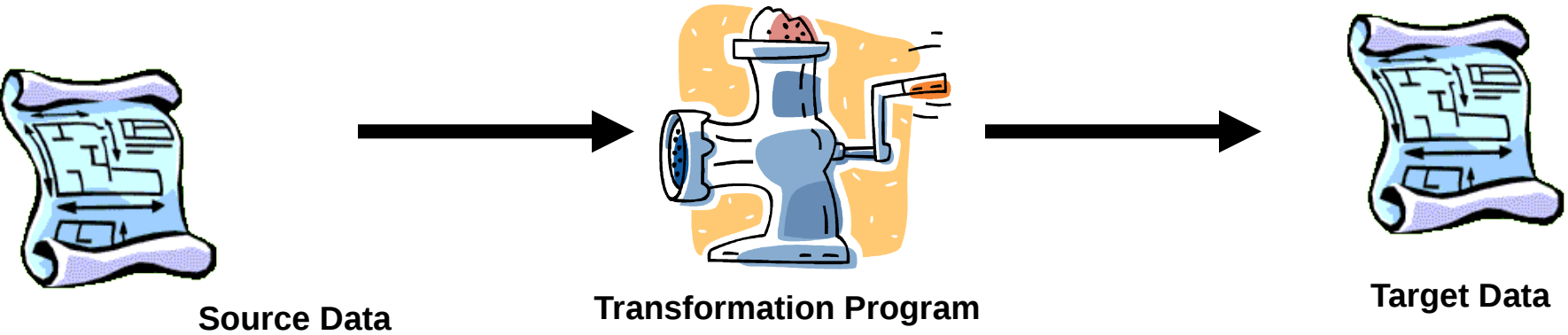
Ease of Maintenance

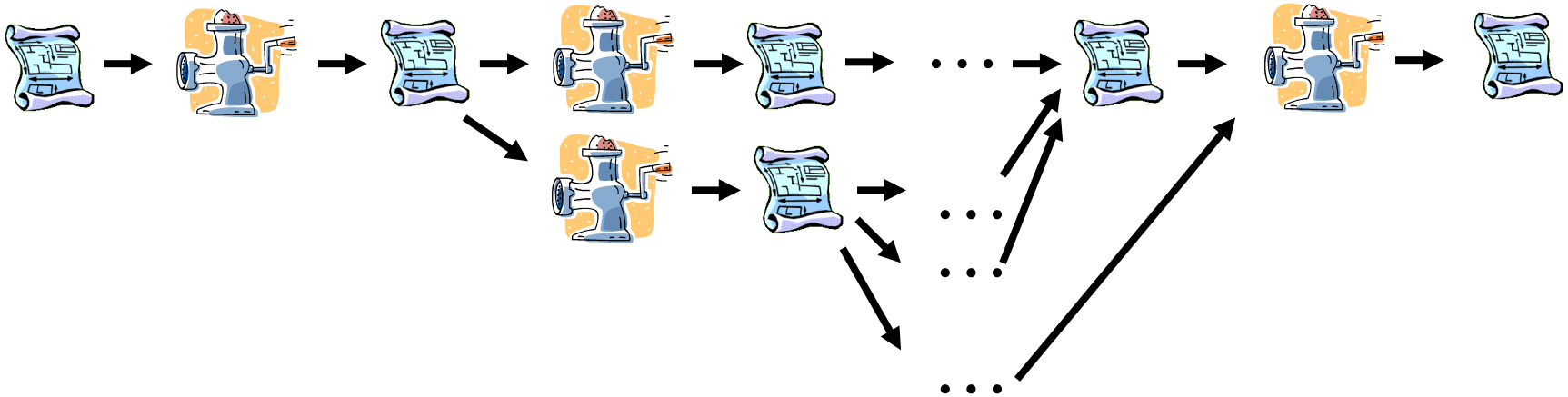
Testability

Security

Usability

Pipe and Filter Architecture





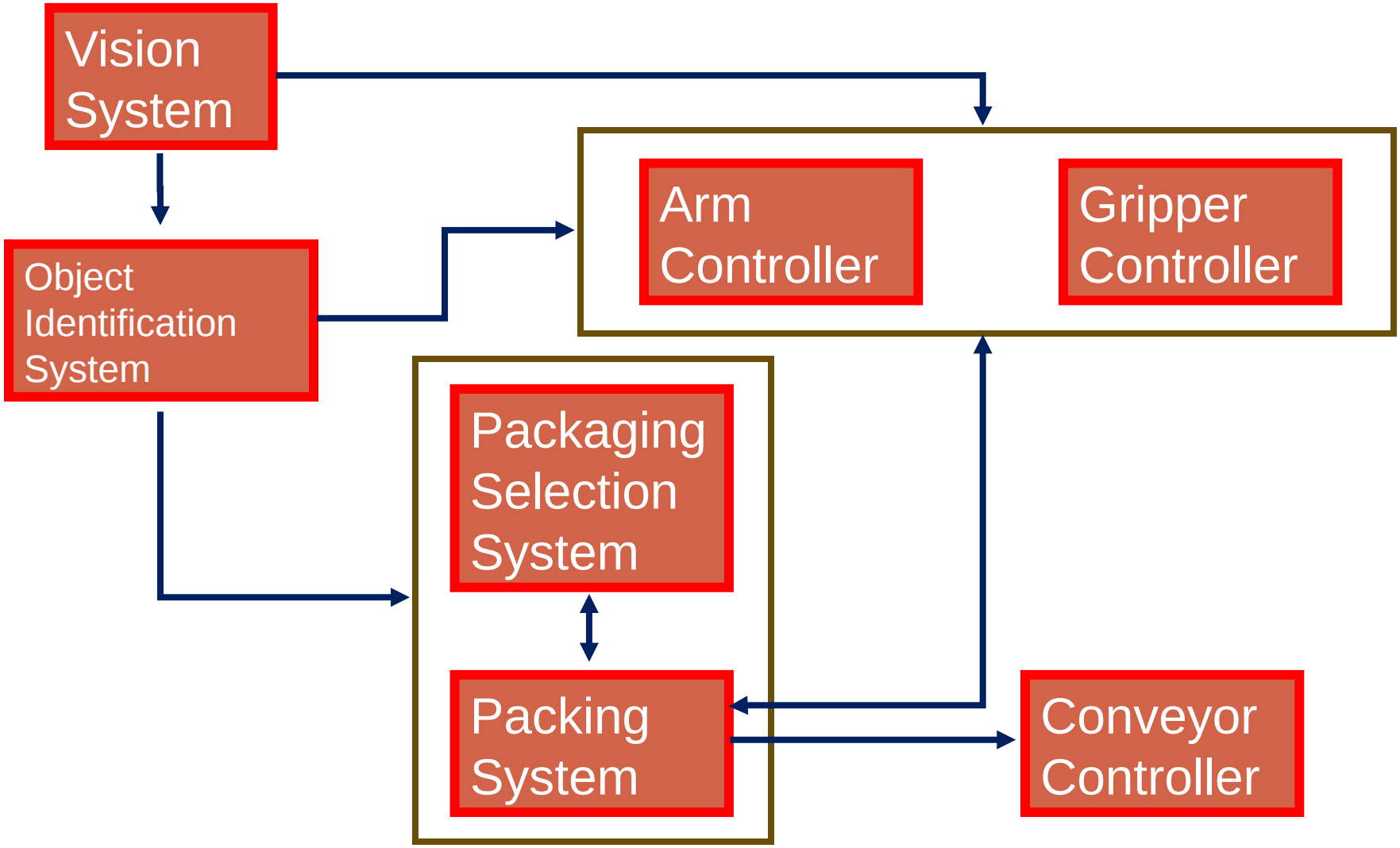
Architectural Models

- Structure, control and modular decomposition may be based on a particular model or architectural style
- However, most systems are heterogeneous in that different parts of the system are based on different models
- The architectural model used affects the performance, robustness, distributability and maintainability of the system

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

Packing Robot Control System

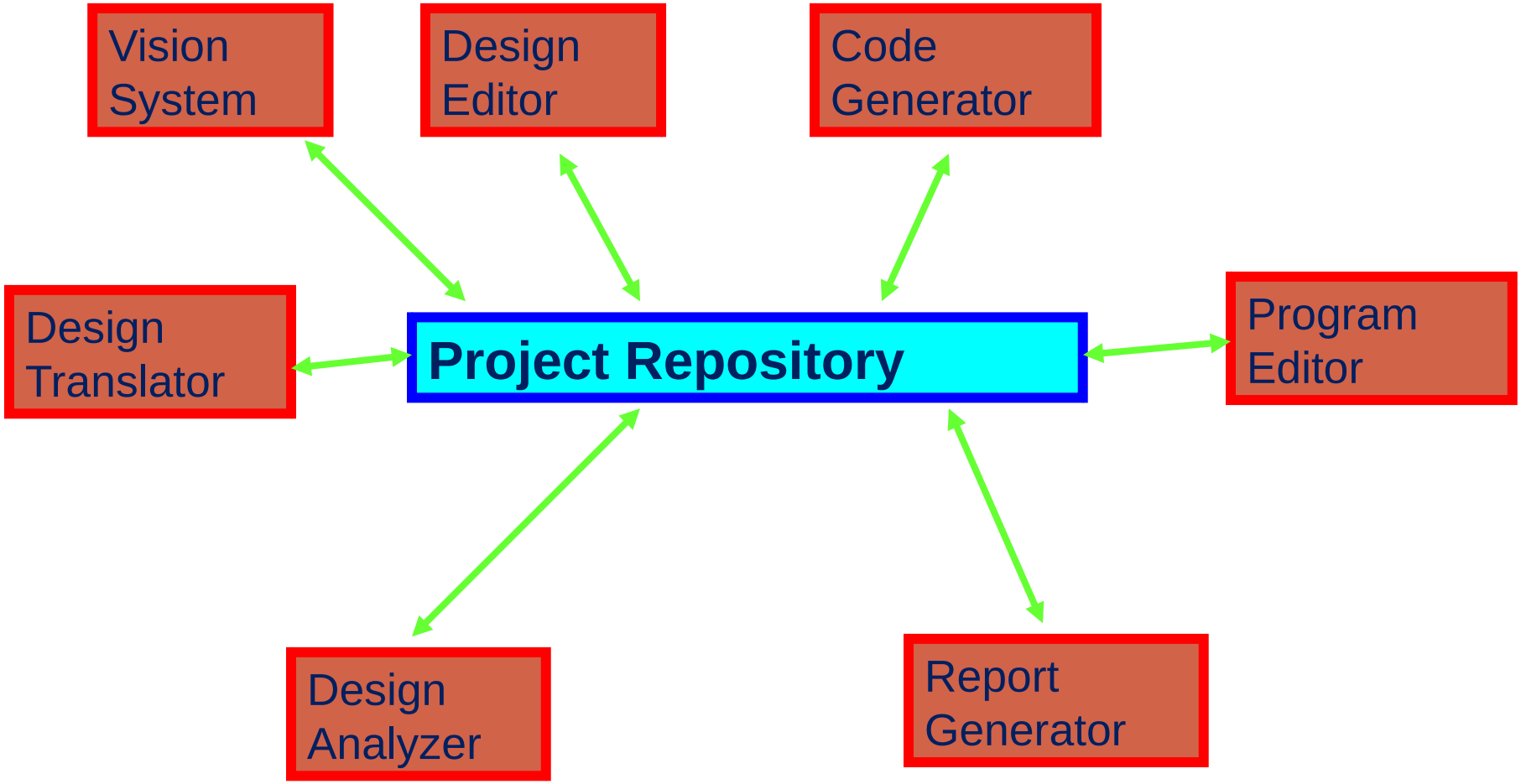


The Repository Model

- 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

CASE Toolkit Architecture



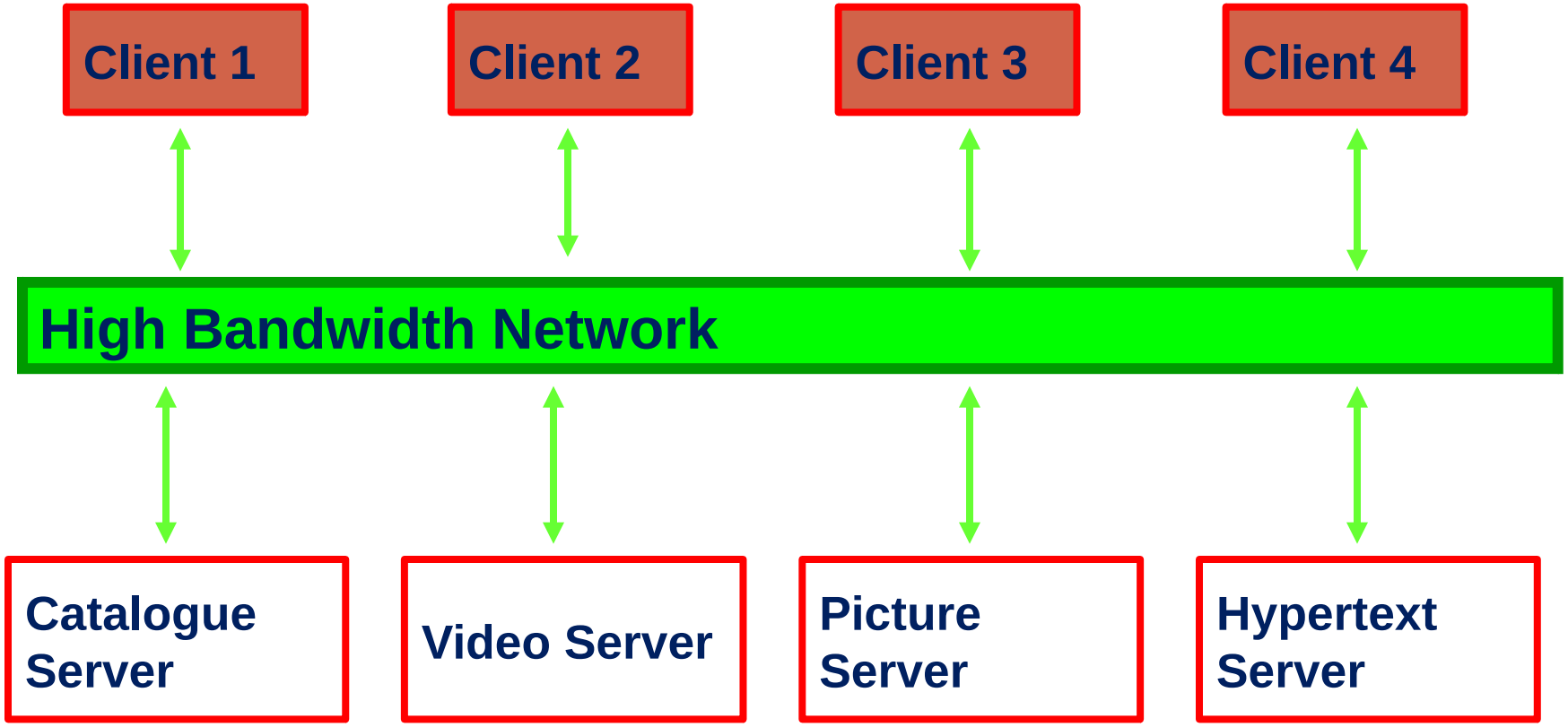
Repository Characteristics Model

- Advantages
  - Efficient way to share large amounts of data
  - Sub-systems need not be concerned with how data is produced
  - Centralized management e.g., backup, security, etc
  - Sharing model is published as the repository schema
- Disadvantages
  - Sub-systems must agree on a repository data model
  - Inevitably a compromise
  - Data evolution is difficult and expensive
  - No scope for specific management policies
  - 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



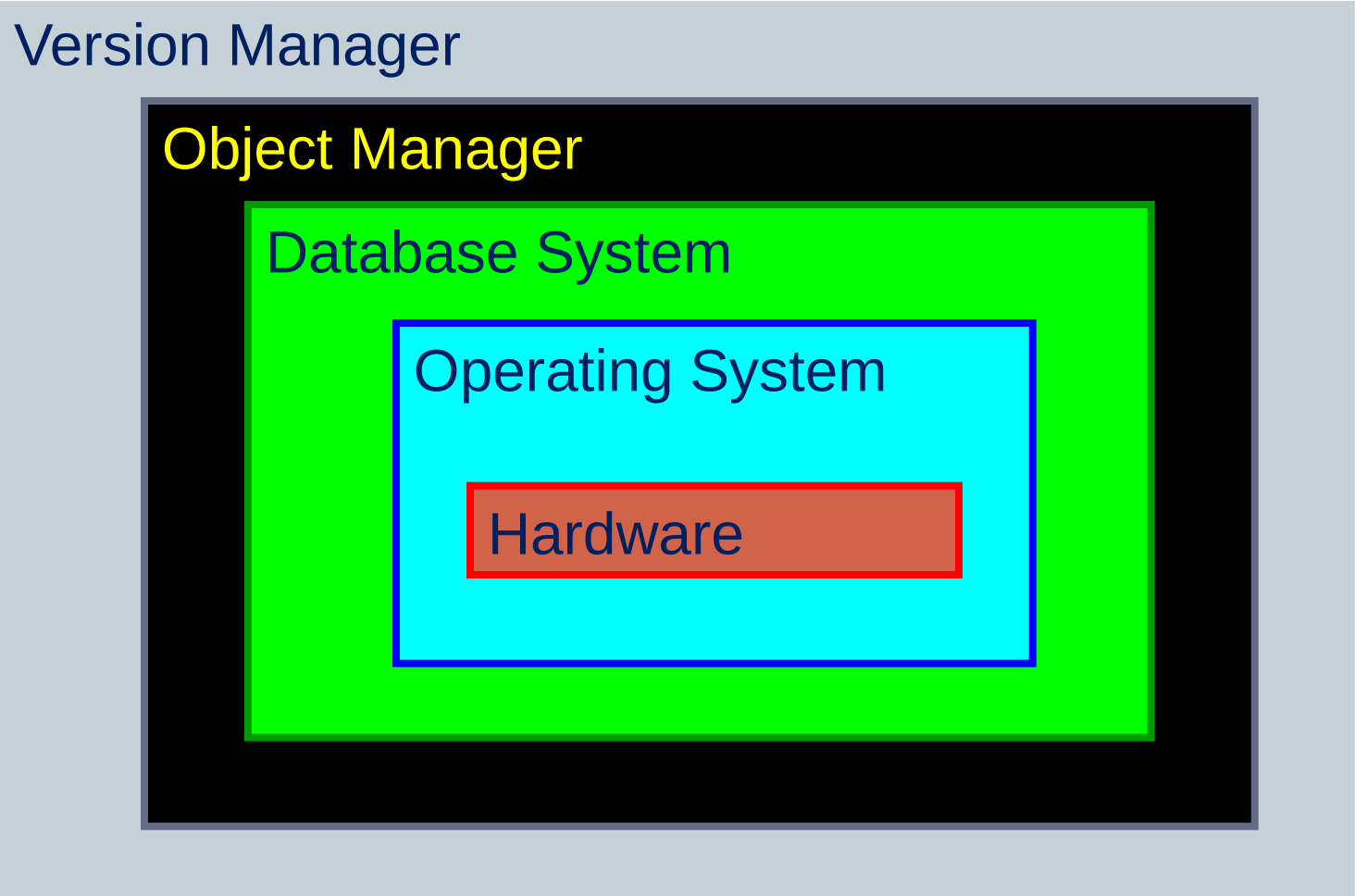
Client-Server Characteristics

- 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 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

Abstract Machine Model

- Used to model the interfacing of sub-systems
- Organizes 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



Control Models

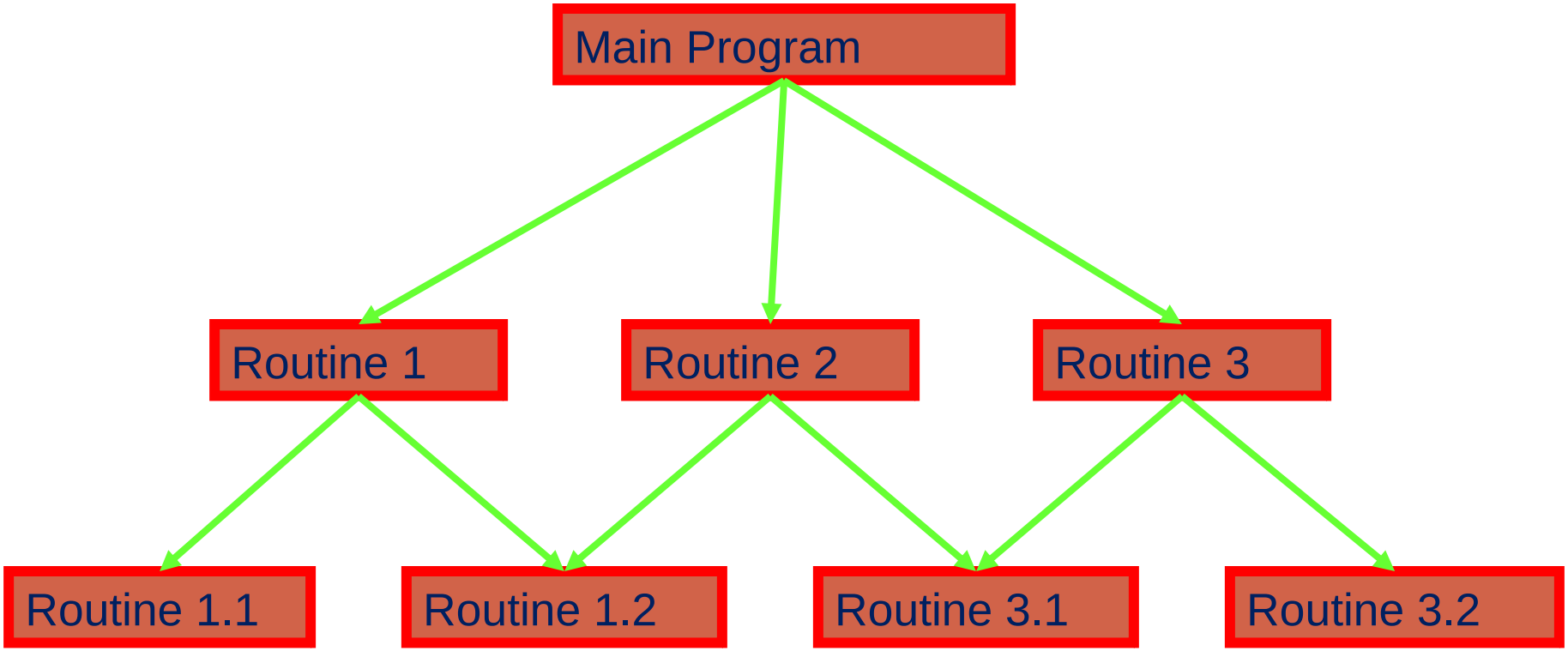
- Are concerned with the control flow between sub-systems
  - Distinct from the system decomposition model

- Centralized 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

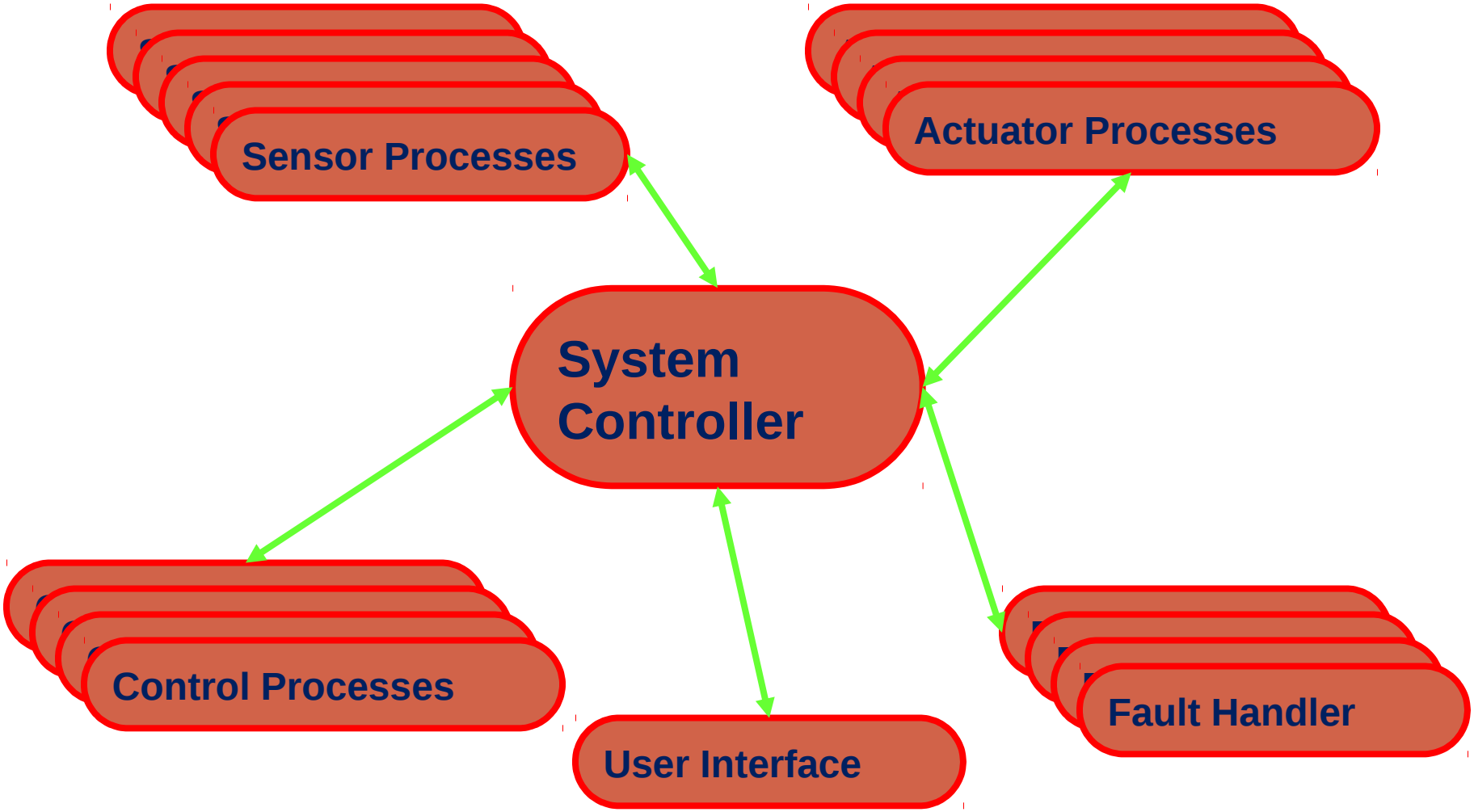
Centralized Control

- A control sub-system takes responsibility for managing the execution of other sub-systems
- 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

Call-Return Model



Real-Time System Control



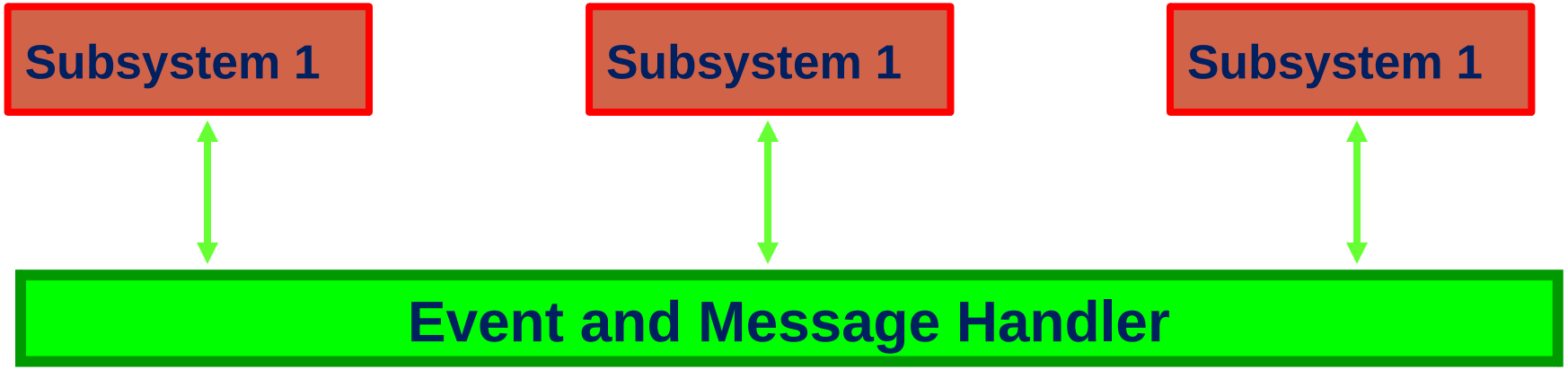
Event-Driven Systems

- 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, for example, spreadsheets

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

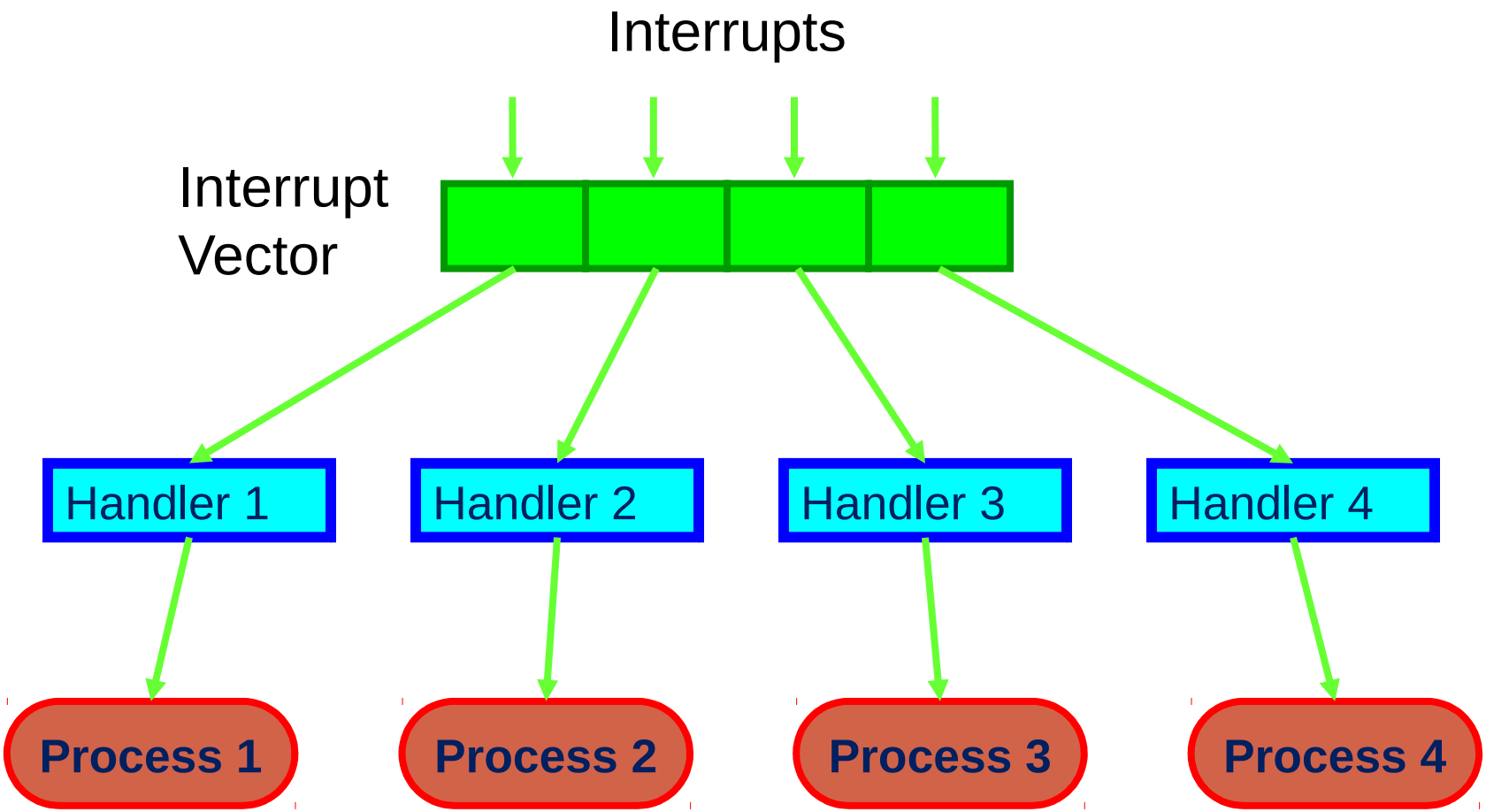
Selective Broadcasting



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

- Another structural level where sub-systems are decomposed into modules
- Two modular decomposition models covered in this class
  - 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

Key Points

- The software architect is responsible for deriving a structural system model, a control model and a sub-system decomposition model
- Large systems rarely conform to a single architectural model
- **System decomposition models** include repository models, client-server models and abstract machine models
- **Control models** include centralized control and event-driven models

Key Points

- Modular decomposition models include data-flow and object models
- Domain specific architectural models are abstractions over an application domain
  - They may be constructed by abstracting from existing systems or may be idealized reference models

Key Points

- **Modular decomposition models** include data-flow and object models
- Domain specific architectural models are abstractions over an application domain
  - They may be constructed by abstracting from existing systems or may be idealized reference models