16.35

Aerospace Software Engineering

Software Architecture

Prof. Kristina Lundqvist
Dept. of Aero/Astro, MIT

Architectural Design

- Establishing the overall structure of a software system

Software Architecture

- "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. By "externally visible" properties, we are referring to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on."
  - Bass, Clements, and Kazman, Software Architecture in Practice, 1998 http://www.sei.cmu.edu/ata/symposium00/Symposium_archrep_slides11/

`http://www.sei.cmu.edu/architecture/definitions.html`

Topics Covered

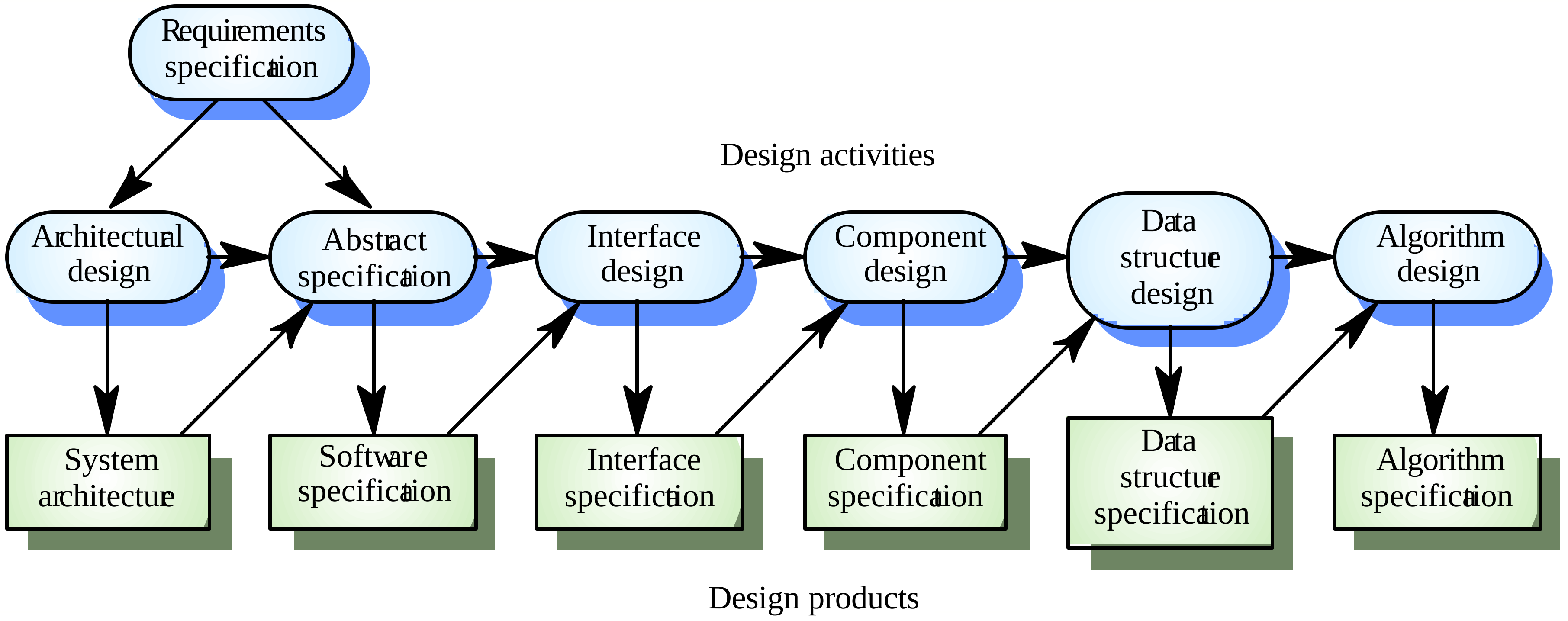
- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

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*

Architectural Design

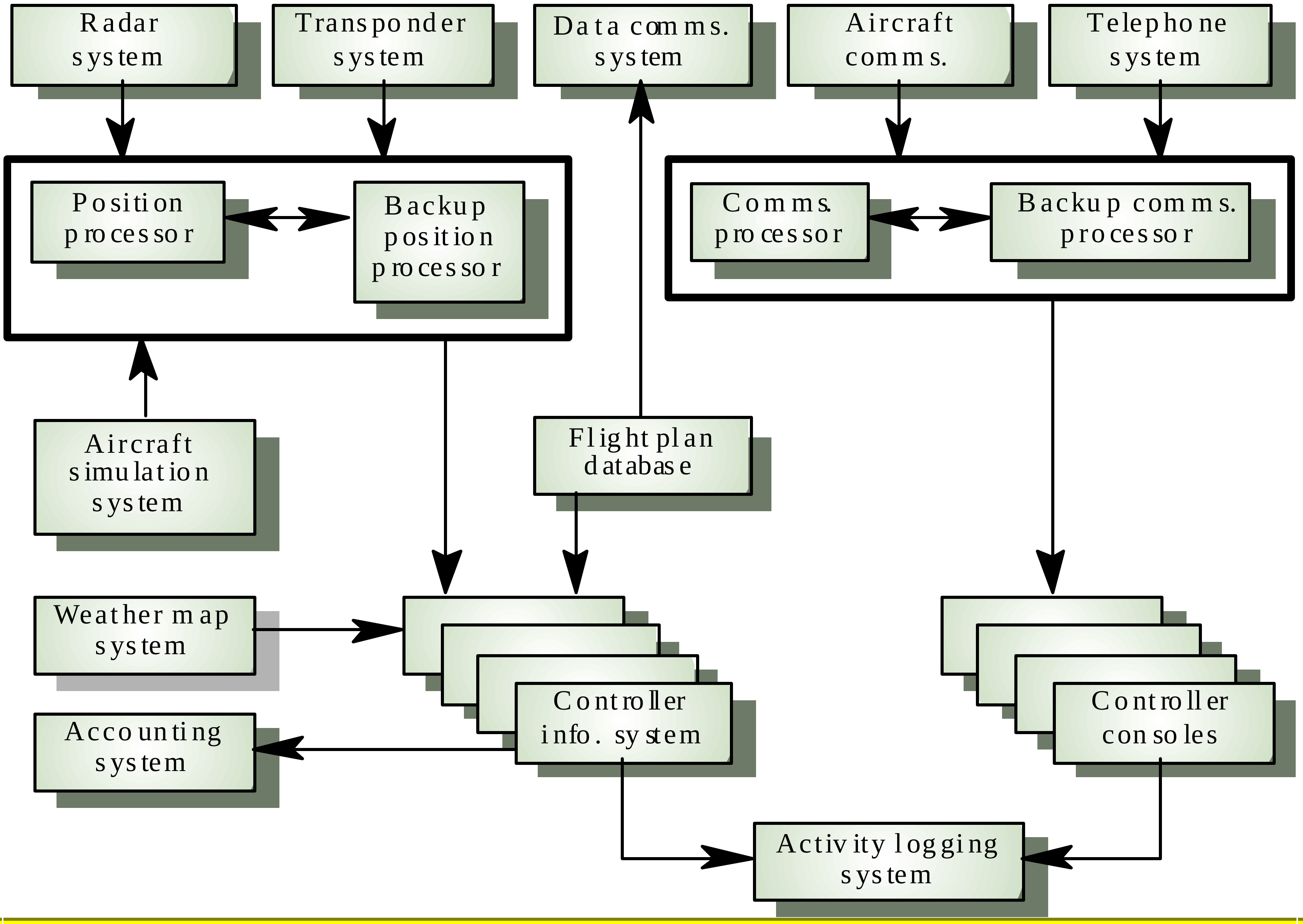
- An early stage of the system design process
- Represents the link between specification and design processes
- It involves identifying major system components and their communications


Requirements specification
Design activities
Architectural design
Abstract specification
Interface design
Component design
Data structure design
Algorithm design
System architecture
Software specification
Interface specification
Component specification
Data structure specification
Algorithm specification
Design products

Architectural Design

- An early stage of the system design process
- Represents the link between specification and design processes
- It involves identifying major system components and their communications

ATC System Architecture



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

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 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

Why Document Software Architectures?

- Marching orders for development teams
- Contract between components, component teams
- Basis for pre-implementation analysis
- Blueprint for maintainers
- Familiarization materials for all stakeholders
- Insurance against personnel turnover

How do you Document Software Architectures?

- UML (Unified Modeling Language)
- Visio and/or Power Point
- Not very well at all ...

- What information do you write down?

High-Quality Architecture Documentation

- Documentation should be written from the point of view of the reader, not the writer
  - Documentation should be organized for ease of reference, not ease of reading
  - Mark what you do not know with “TBD” rather than leaving it blank
- Avoid repetition. Everything in one place.
- Avoid unintentional ambiguity
  - Explain your notation. Beware boxes and lines
- Use a standard organization
- Record rationale
- Keep it current
- Review it for fitness of purpose

Documenting Architecture

- Primarily a matter of:
  - Documenting the relevant structures
  - Documenting trans-structure information
- Examples of structures:
  - Dataflow
  - Process
  - Module

Architecture Attributes

- Performance
  - Localise operations to minimise sub-system communication
- Security
  - Use a layered architecture with critical assets in inner layers
- Safety
  - Isolate safety-critical components
- Availability
  - Include redundant components in the architecture
- Maintainability
  - Use fine-grain, self-contained components

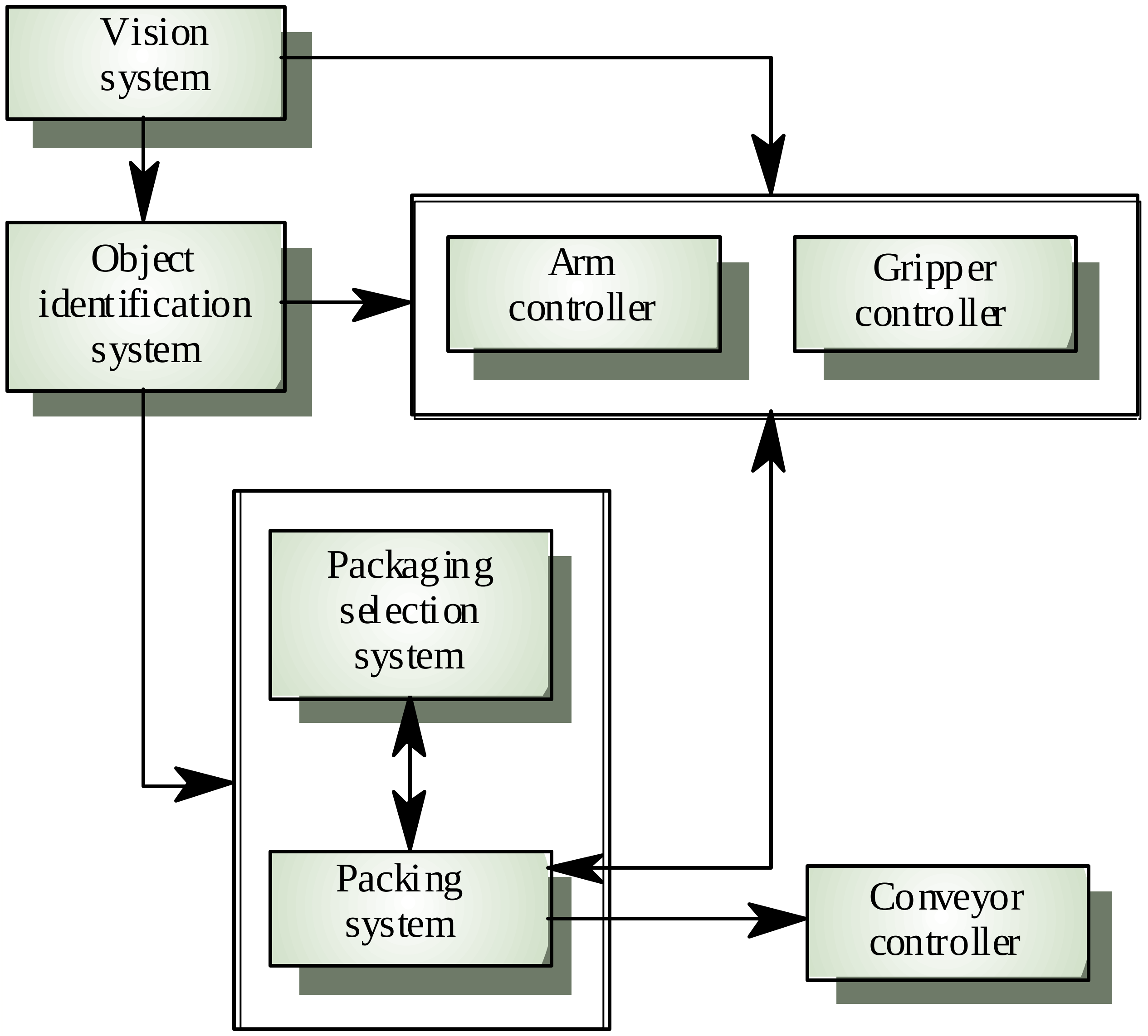
Topics Covered

- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

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

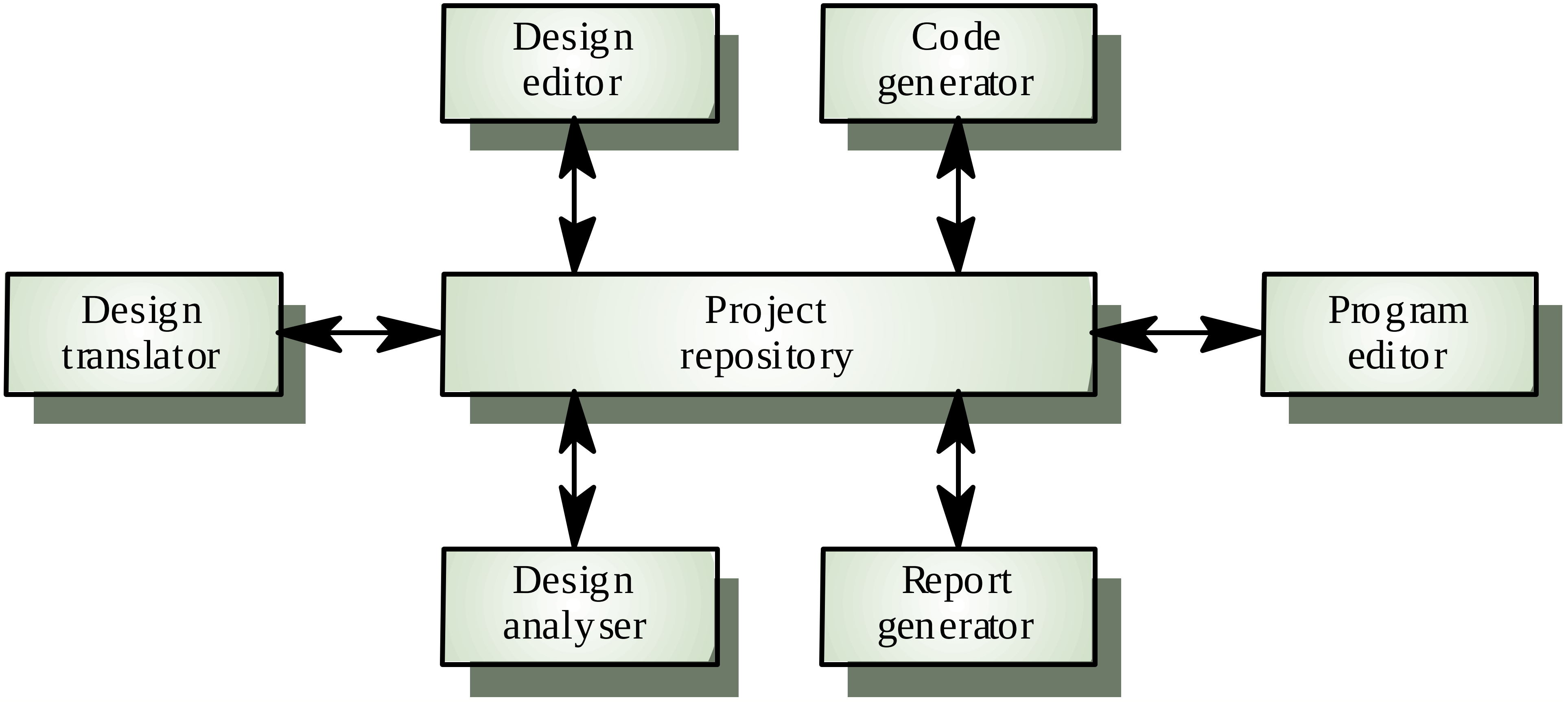
Packing Robot Control System



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 Toolset Architecture



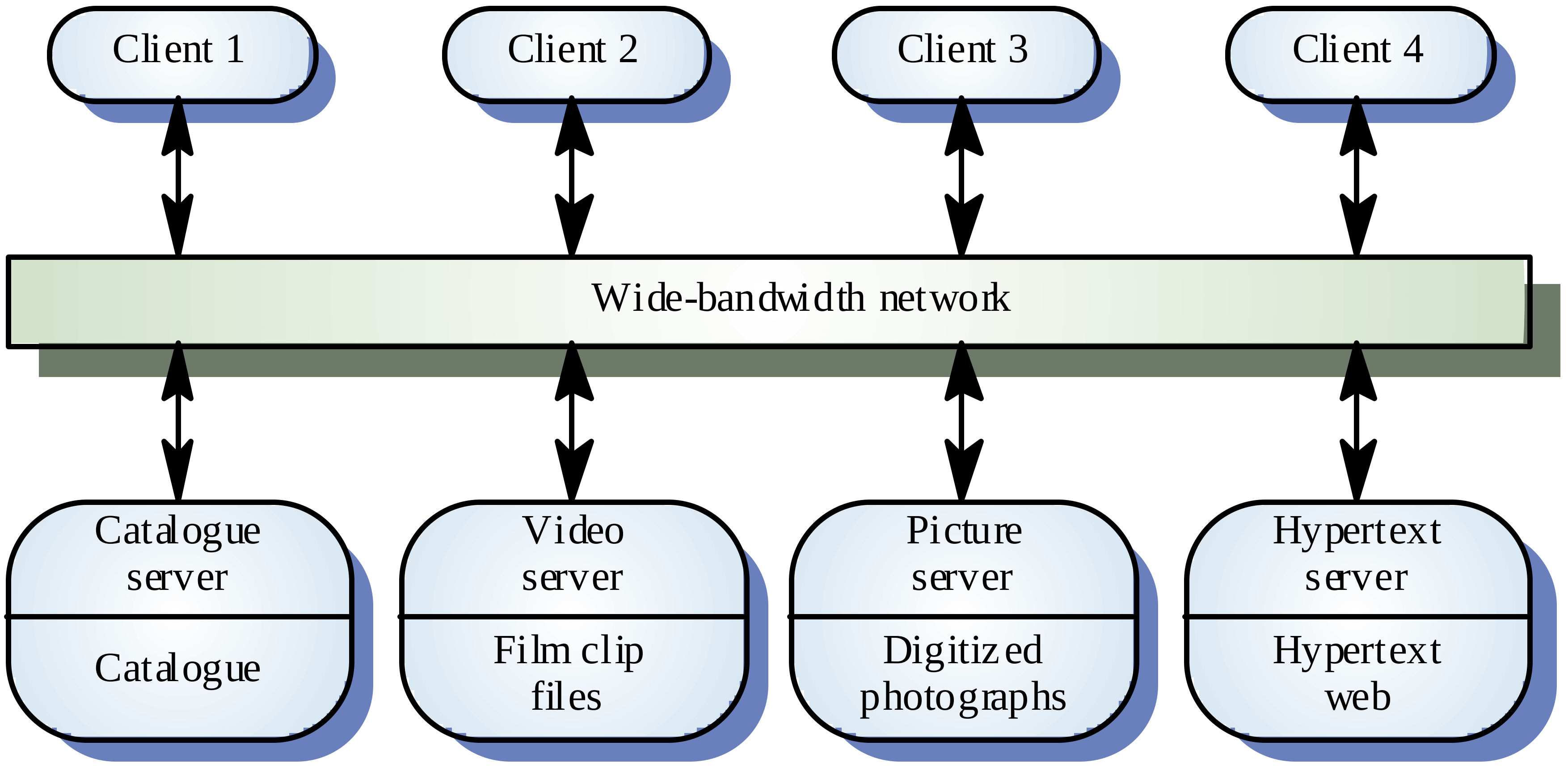
Repository Model Characteristics

- Advantages
  - Efficient way to share large amounts of data
  - Sub-systems need not be concerned with how data is produced
  - Centralised 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



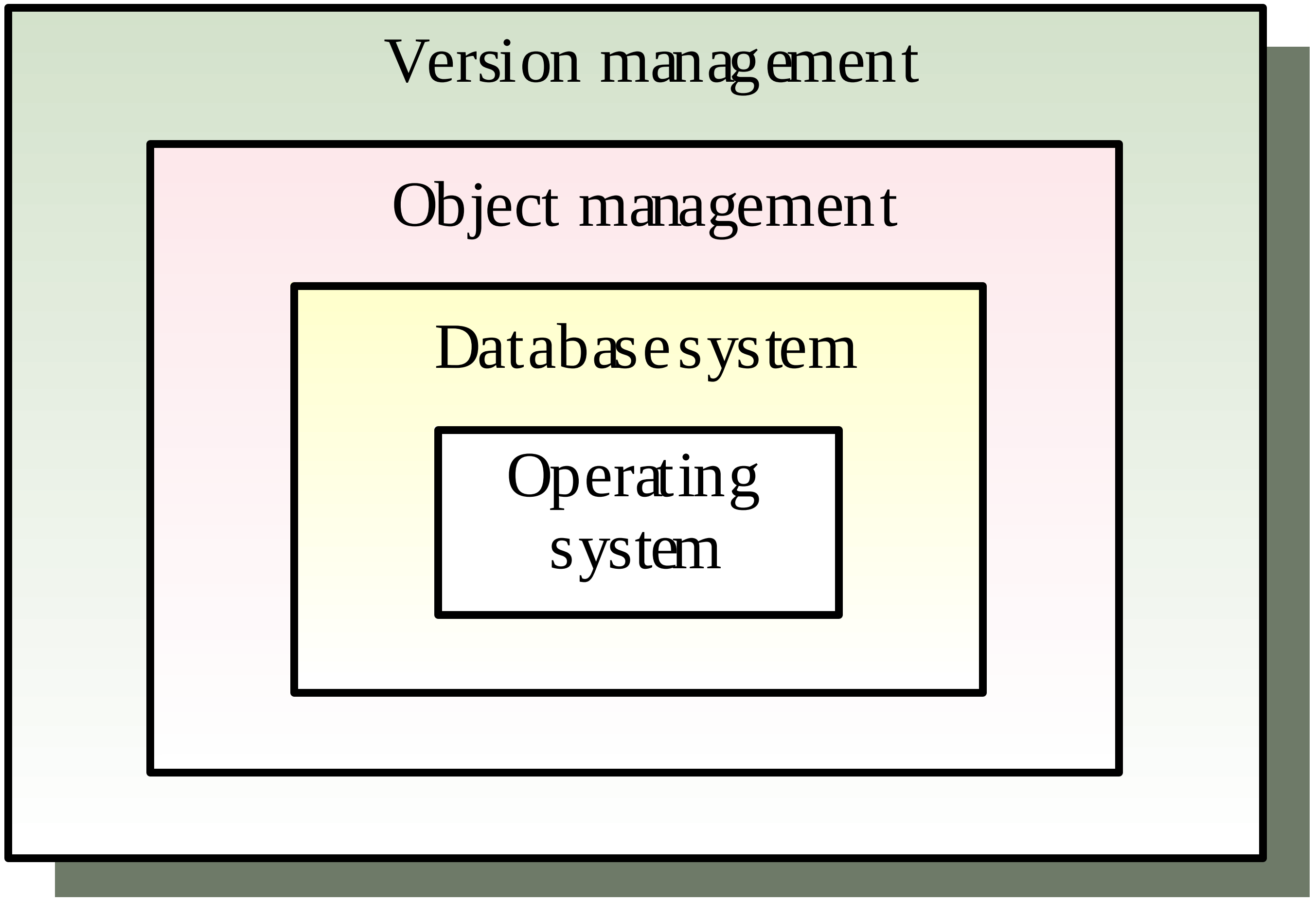
Client-server Characteristics

- 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 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 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



Topics Covered

- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

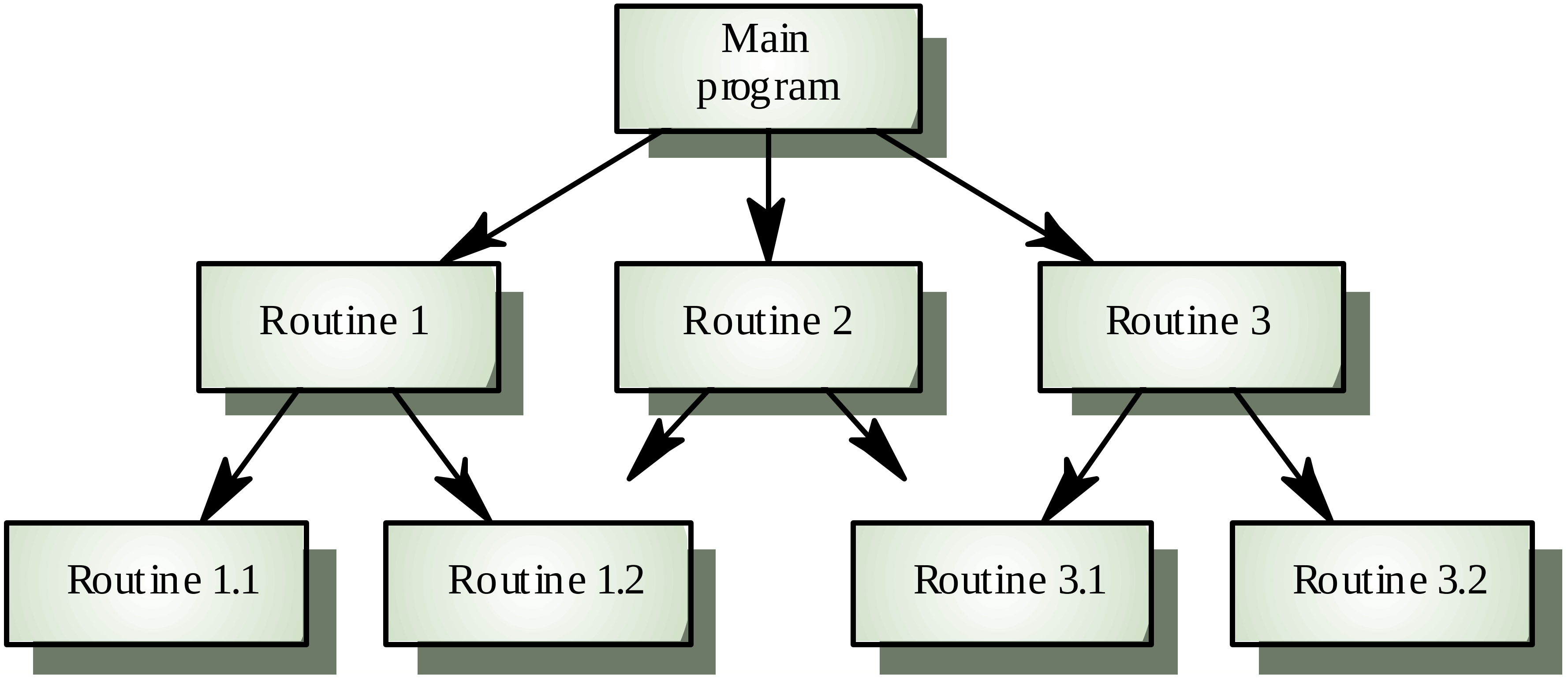
Control Models

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

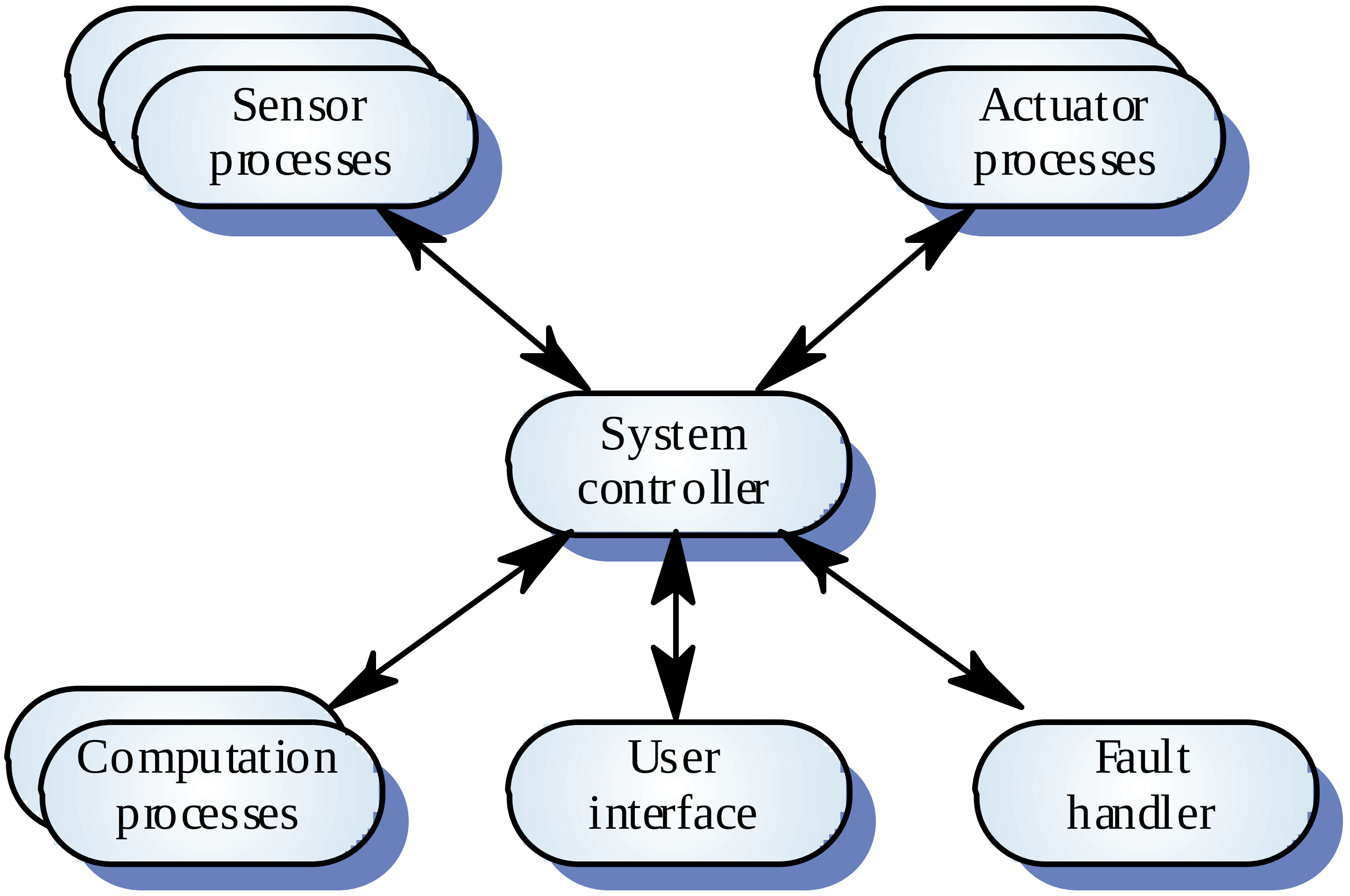
Centralised Control

- 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.
- Manager model
  - One system component controls the stopping, starting and coordination of other system processes.

Call-return Model



Real-time System Control



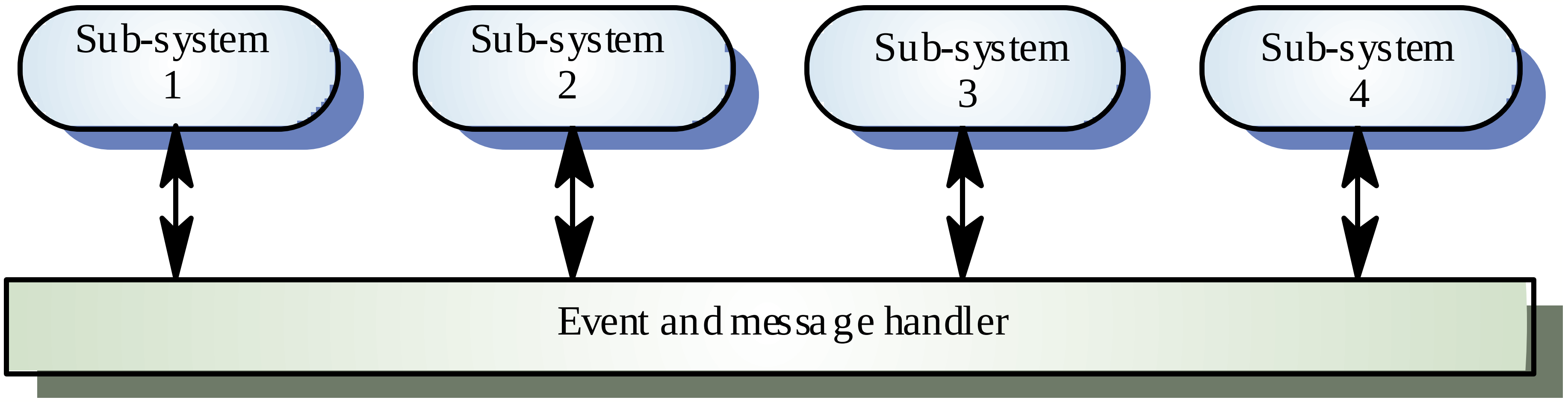
Event-driven Systems

- Driven by externally generated events
  - where the timing of the event is outside the control of the sub-systems which process the event
- Two principal event-driven models
  - Broadcast models
  - Interrupt-driven models
- Other event driven models include spreadsheets and production systems

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

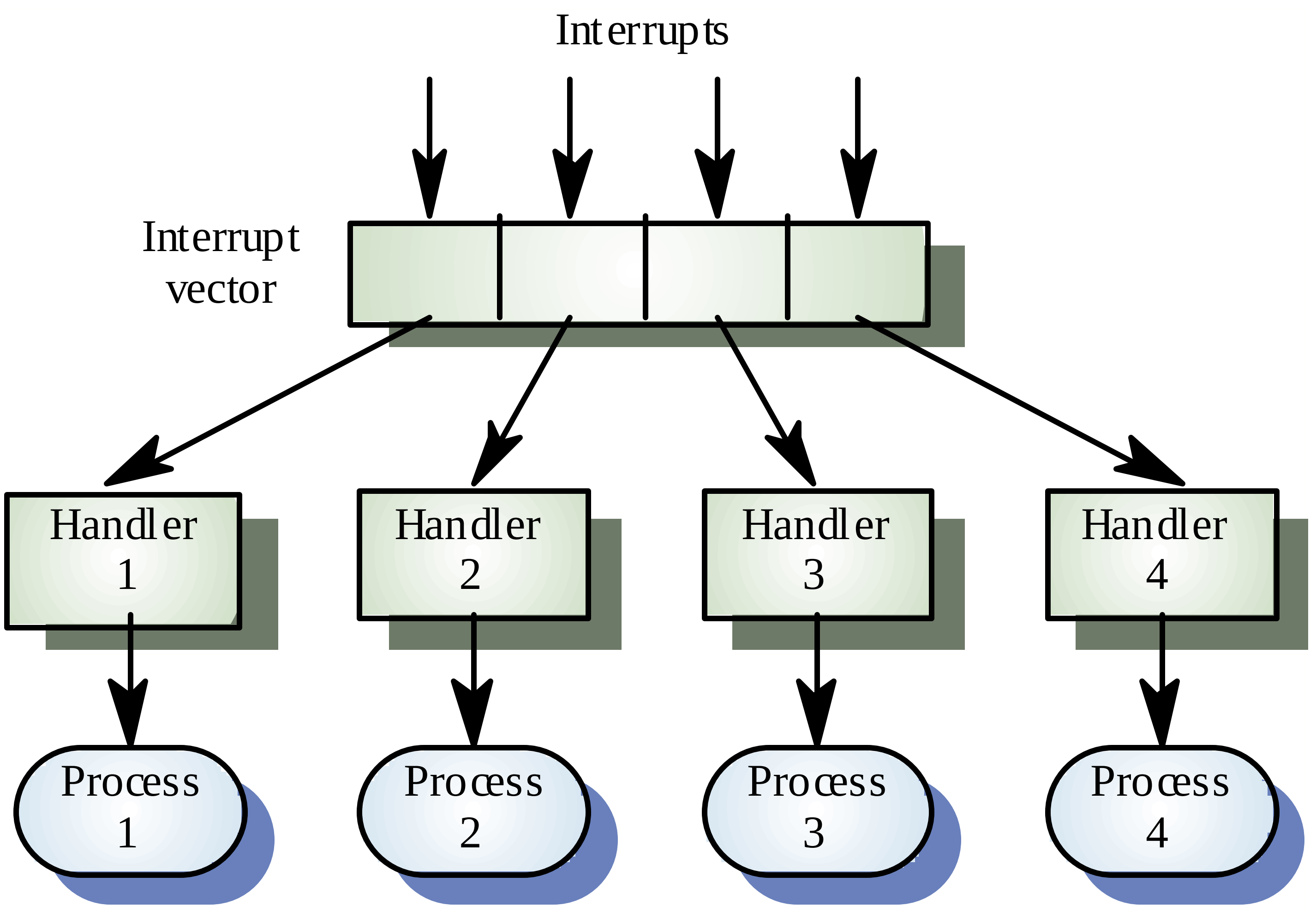
Selective Broadcasting



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



Topics Covered

- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

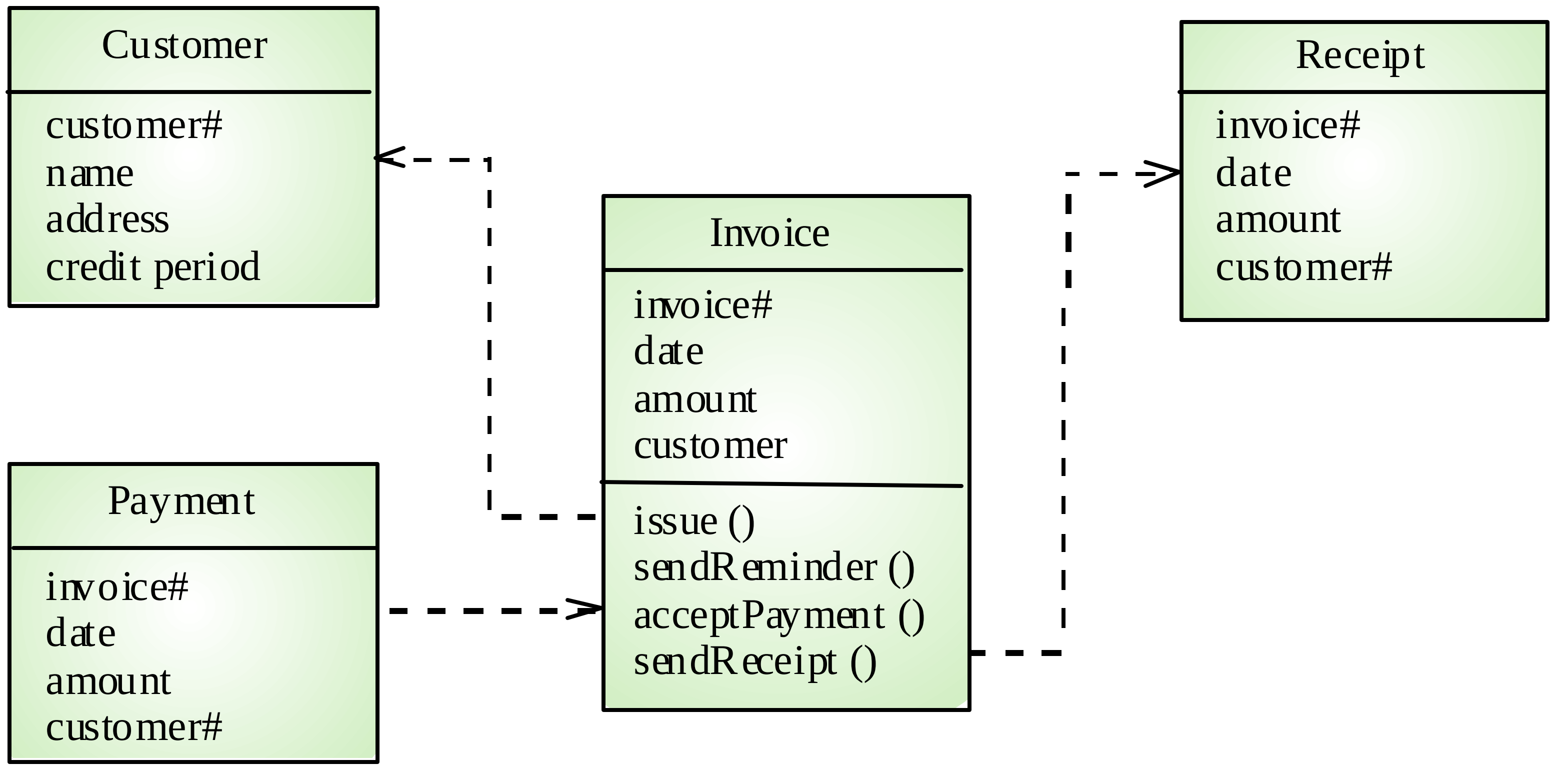
Modular Decomposition

- 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.
- 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

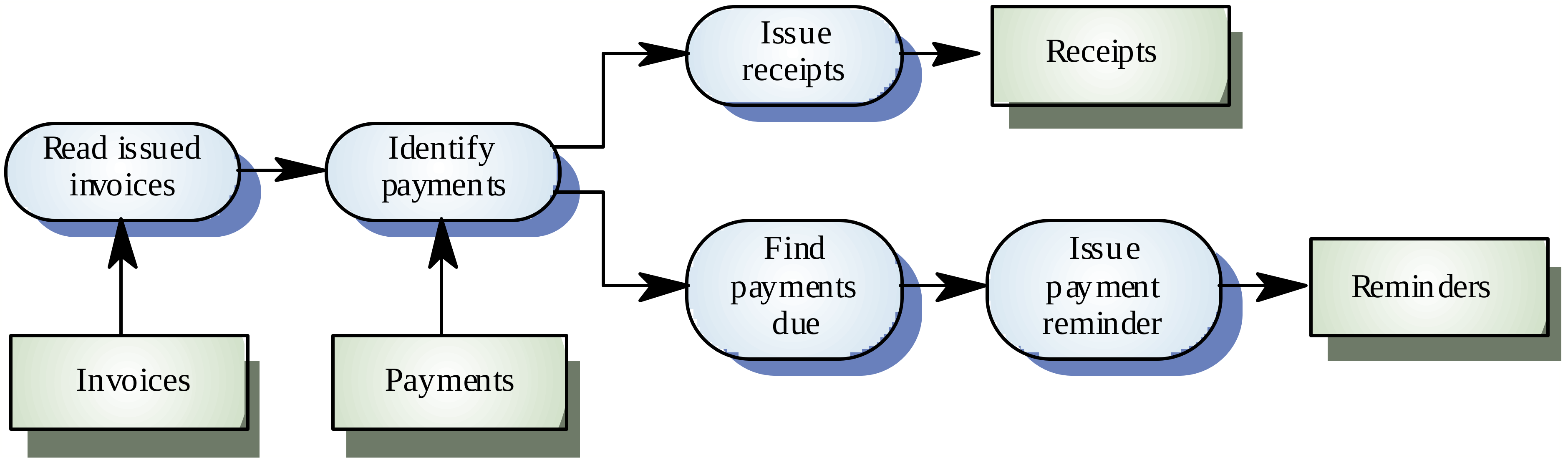
Invoice Processing System



Data-flow Models

- Functional transformations process their inputs to produce outputs
- 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



Topics Covered

- System structuring
- Control models
- Modular decomposition
- Domain-specific architectures

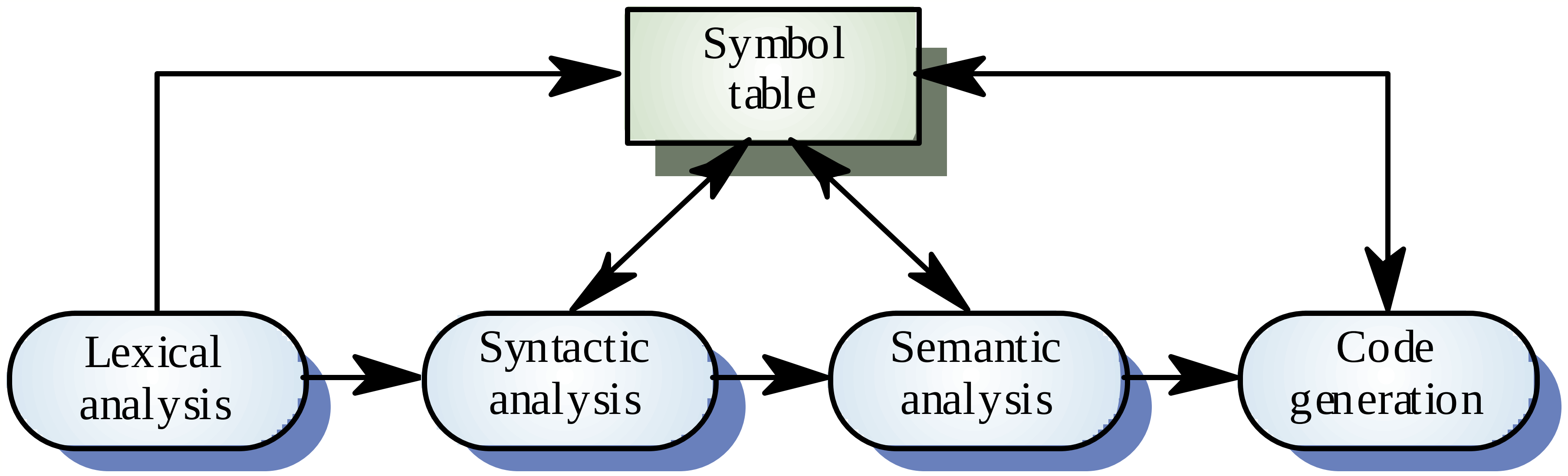
Domain-specific Architectures

- Architectural models which are specific to some application domain
- Two types of domain-specific model
  - **Generic models** which are abstractions from a number of real systems and which encapsulate the principal characteristics of these systems
  - **Reference models** which are more abstract, idealised model. Provide a means of information about that class of system and of comparing different architectures
- Generic models are usually bottom-up models; Reference models are top-down models

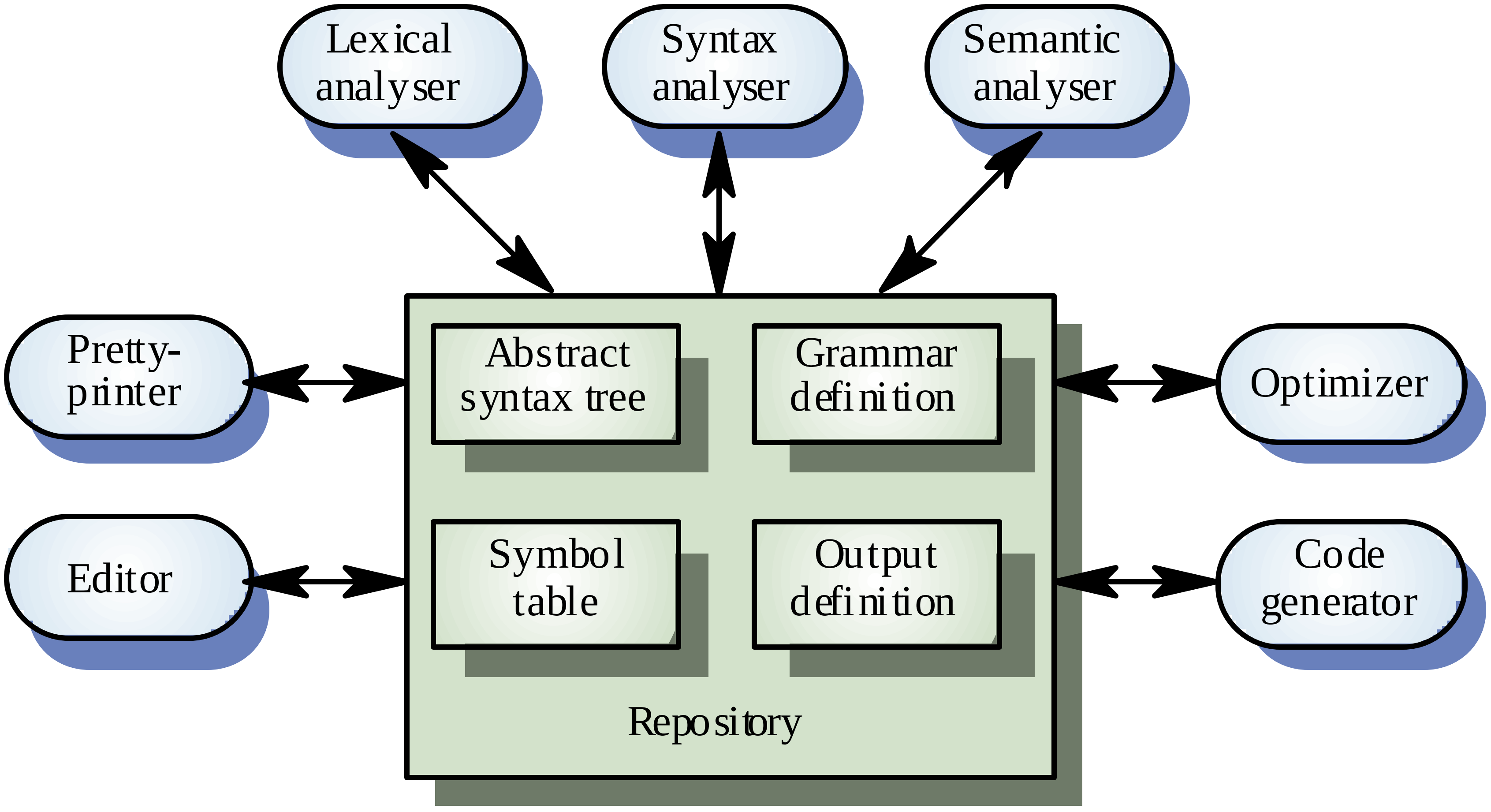
Generic Models

- Compiler model is a well-known example although other models exist in more specialised application domains
  - Lexical analyser
  - Symbol table
  - Syntax analyser
  - Syntax tree
  - Semantic analyser
  - Code generator

Compiler Model



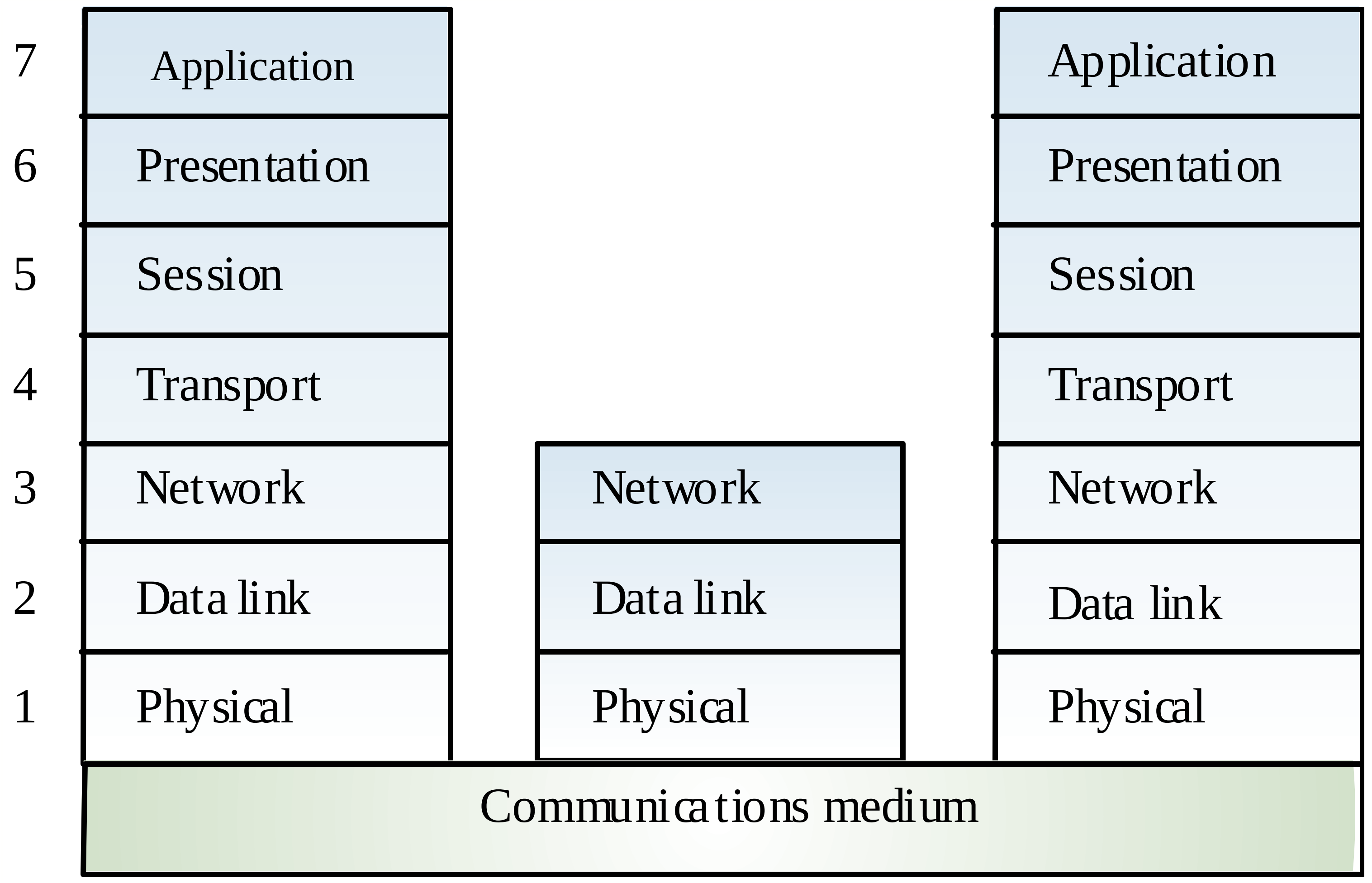
Language Processing System



Reference Architectures

- Reference models are derived from a study of the application domain rather than from existing systems
- May be used as a basis for system implementation or to compare different systems. It acts as a standard against which systems can be evaluated
- OSI model is a layered model for communication systems

OSI Reference Model



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 centralised 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 idealised reference models

…

- Friday group meetings in 33-308
  - 12-12:30 The green team
  - 2:30-3 Phifer, Sidelnik, Oulette, Chang (?)
  - 3-3:30 Mandic, Nyenke, da Silva, Riedel
  - 3:30-4 Qu, Stringfellow, Guevara, Kambouchev
- Turn in problem sets to folder on server:
  - \\aero-astro\16.35
- Feed back on the Ada Compendium to Malia
- CD