XI. Architectural Design

Objectives

- To introduce architectural design and to discuss its importance
- To introduce importance of different views
- To introduce the concepts of patterns and frameworks
- To explain the architectural design decisions that have to be made
- To introduce three complementary architectural styles covering organisation, decomposition and control
- To discuss reference architectures are used to communicate and compare architectures

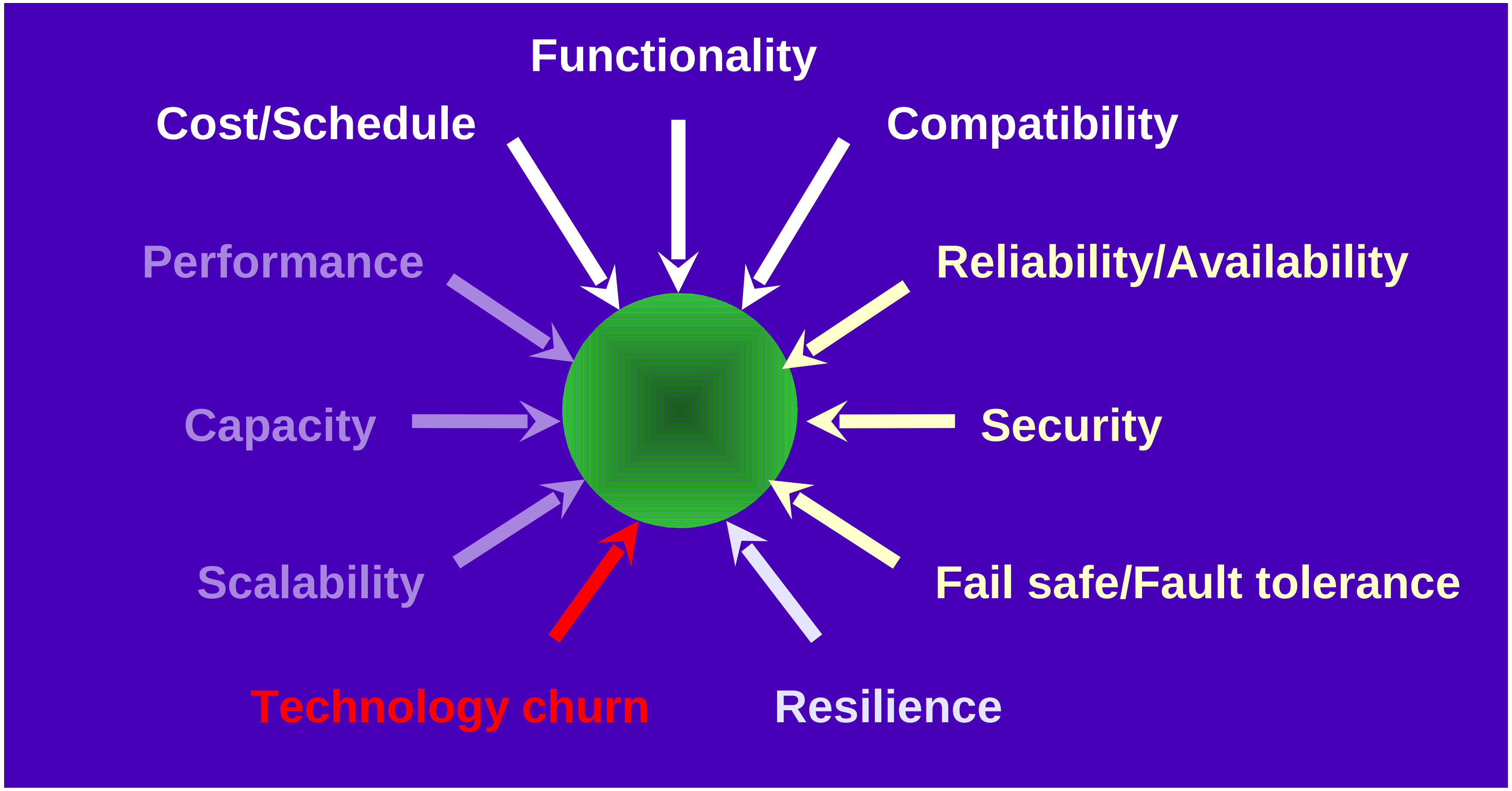
Software architecture

- 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**.

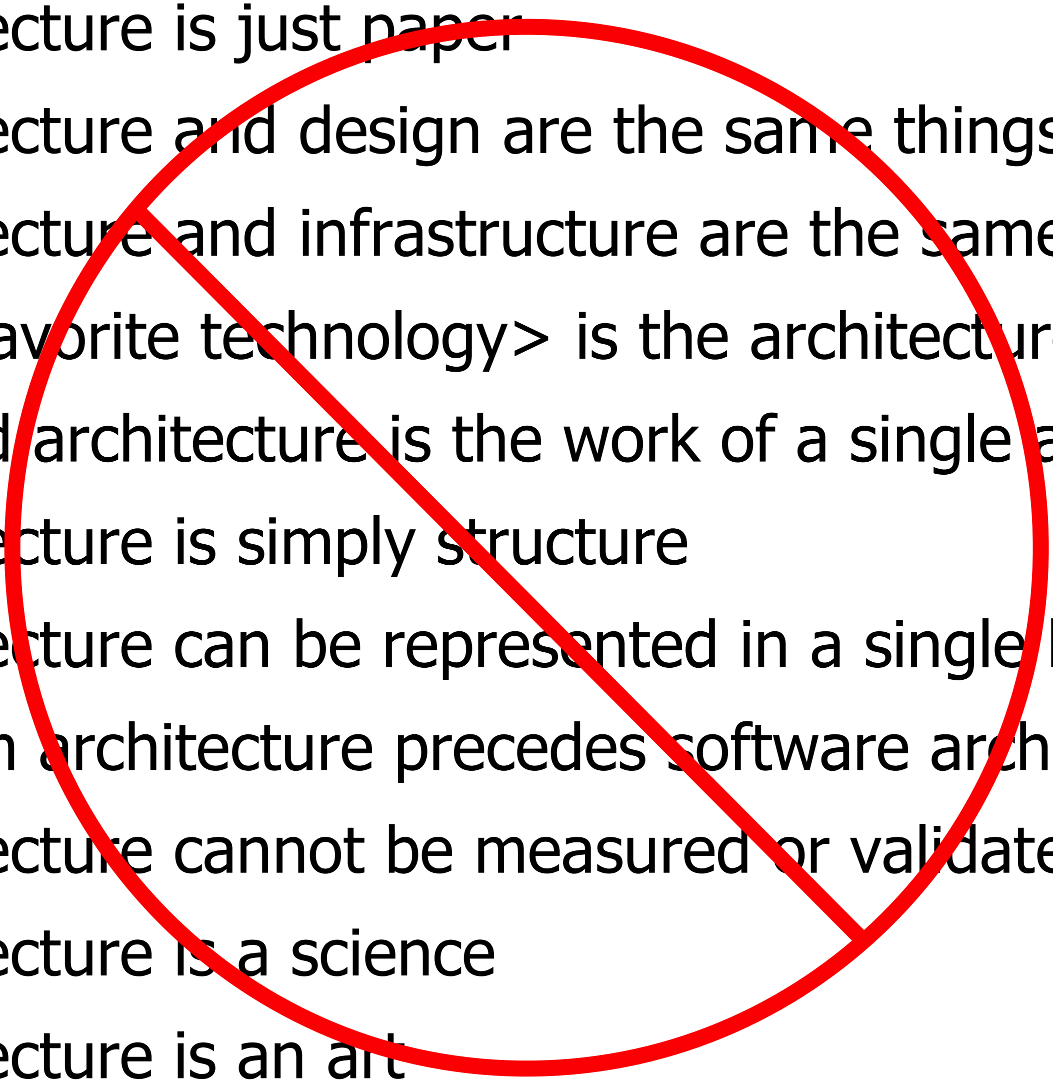
Architectural design decisions

- Is there a generic application architecture that can be used?
- How will the system be distributed?
- What architectural styles are appropriate?
- What approach will be used to structure the system?
- How will the system be decomposed into modules?
- What control strategy should be used?
- How will the architectural design be evaluated?
- How should the architecture be documented?

Forces In Software



Misconceptions about architecture



- Architecture is just paper
- Architecture and design are the same things
- Architecture and infrastructure are the same things
- <my favorite technology> is the architecture
- A good architecture is the work of a single architect
- Architecture is simply structure
- Architecture can be represented in a single blueprint
- System architecture precedes software architecture
- Architecture cannot be measured or validated
- Architecture is a science
- Architecture is an art

Architecture is just paper

- A system’s architecture ultimately resides in executable code
- A system’s architecture may be visualized in models
- Every system has an architecture; some architectures are made manifest and visible, many others are not

Architecture is design

- All architecture is design, but not all design is architecture
- Architecture focuses on *significant* design decisions, decisions that are both structurally and behaviorally important as well as those that have a lasting impact on the performance, reliability, cost, and resilience of the system
- Architecture involves the how and the why, not just the what

Architecture is infrastructure

- Infrastructure is an integral and important part of architecture, but there is more to architecture than just infrastructure
- Significant patterns will manifest themselves at many different levels and dimensions of a system
- Too narrow a view of architecture will lead to a very pretty infrastructure, but the wrong infrastructure for the problem at hand

<my favorite technology> is the architecture

- A given technology only serves to implement some dimension of an architecture
  - The network is the architecture
  - The database is the architecture
  - The transaction server is the architecture
  - J2EE is the architecture
- Architecture is more than just a list of products
- Technology shapes an architecture, but a resilient architecture should never be bound to all of the technologies that form it

Architecture is the work of a single architect

- Conceptual integrity is essential, but the complexity of most interesting systems leads development to be a team sport
- Fred Brooks (1975), but then Fred Brooks (1995)
- The architecture team is
  - Not a committee
  - Not a problem clearinghouse
  - Not an ivory tower
- The architecture team
  - Needs a clear leader
  - Requires a mix of specialties
  - Manifests itself at many levels in the system

Coplien & Harris, *Organizational Patterns*

Architecture is structure

- Architecture does involve structure, decomposition, and interfaces
- Architecture also involves behavior
- A system’s architecture is always projected to a given context

Architecture is flat

- Architecture is flat only in trivial systems
- Multiple stakeholders with multiple concerns lead to multiple views with multiple blueprints
- Using a single blueprint to represent all or most of system’s architecture leads to a semantic muddle
- The 4+1 view model has proven to be both necessary and sufficient for most interesting systems

Kruchten, “The 4+1 Model View”

System architecture comes first

- Software has a longer life than hardware
- Complex systems require well-informed hardware/software tradoffs, which cannot be made in a strict sequence
- Forcing a hardware-first process typically leaves to stove pipe systems

Architecture can't be measured

- The very purpose of a blueprint is to provide a tangible artifact that can be used to visualize, specify, construct, document - and reason about - a system
- A system's architecture can be used to
  - Mitigate technical risks through the release of a continuous stream of executables
  - Improve learning and understanding and communicate important decisions
  - Accelerate testing and attack integration risks
  - Set expectations
  - Break in the development environment and the team

Architecture is a science

- There exists only a modest body of knowledge about software architecture
- Scientific and analytical methods are lacking; those that do exist are hard to apply
- There is no perfect design; architecture involves the management of extreme ambiguity and contradiction
- Experience counts: the best architects are grown, not born

Petroski, *Small Things Considered*

Architecture is an art

- Even the best architects copy solutions that have proven themselves in practice, adapt them to the current context, improve upon their weaknesses, and then assemble them in novel ways with very modest incremental improvements
- The “artsy” part of software architecture is minimal
- An architectural process can be established with intentional artifacts, clear activities, and well-defined

Rechtin Maier, *Systems Architecting*

Architecture defined

- Architecture n (1563)
  - The art or science of building or constructing edifices of any kind for human use
  - The action or process of building
  - Architectural work; structure, building
  - The special method of 'style' in accordance with which the details of the structure and ornamentation of a building are arranged
  - Construction or structure generally
  - The conceptual structure and overall logical organization of a computer or computer-based system from the point of view of its use or design; a particular realization of this

Oxford English Dictionary, 2nd ed.

Physical systems

- Mature physical systems have stable architectures
  - Aircraft, cars, and ships
  - Bridges and buildings
- Such architectures have grown over long periods of time
  - Trial-and-error
  - Reuse and refinement of proven solutions
  - Quantitative evaluation with analytical methods
- Mature domains are dominated by engineering efforts
  - Analytical engineering methods
  - New materials
  - New manufacturing processes

Software-intensive system

- A system in which software is the dominant, essential, and indispensable element
  - E-commerce system
  - IT (business) system
  - Telephone switch
  - Flight control system
  - Real-time control system (e.g. industrial robot)
  - Sophisticated weapons system
  - Software development tools
  - System software (e.g. operating systems or compilers)

Architecting software is different

- No equivalent laws of physics
- Transparency
- Complexity
  - Combinatorial explosion of state space
  - Non-continuous behavior
  - Systemic issues
- Requirement and technology churn
- Low replication and distribution costs

Architecture defined

- Software architecture is what software architects do!

Beck

Architecture defined

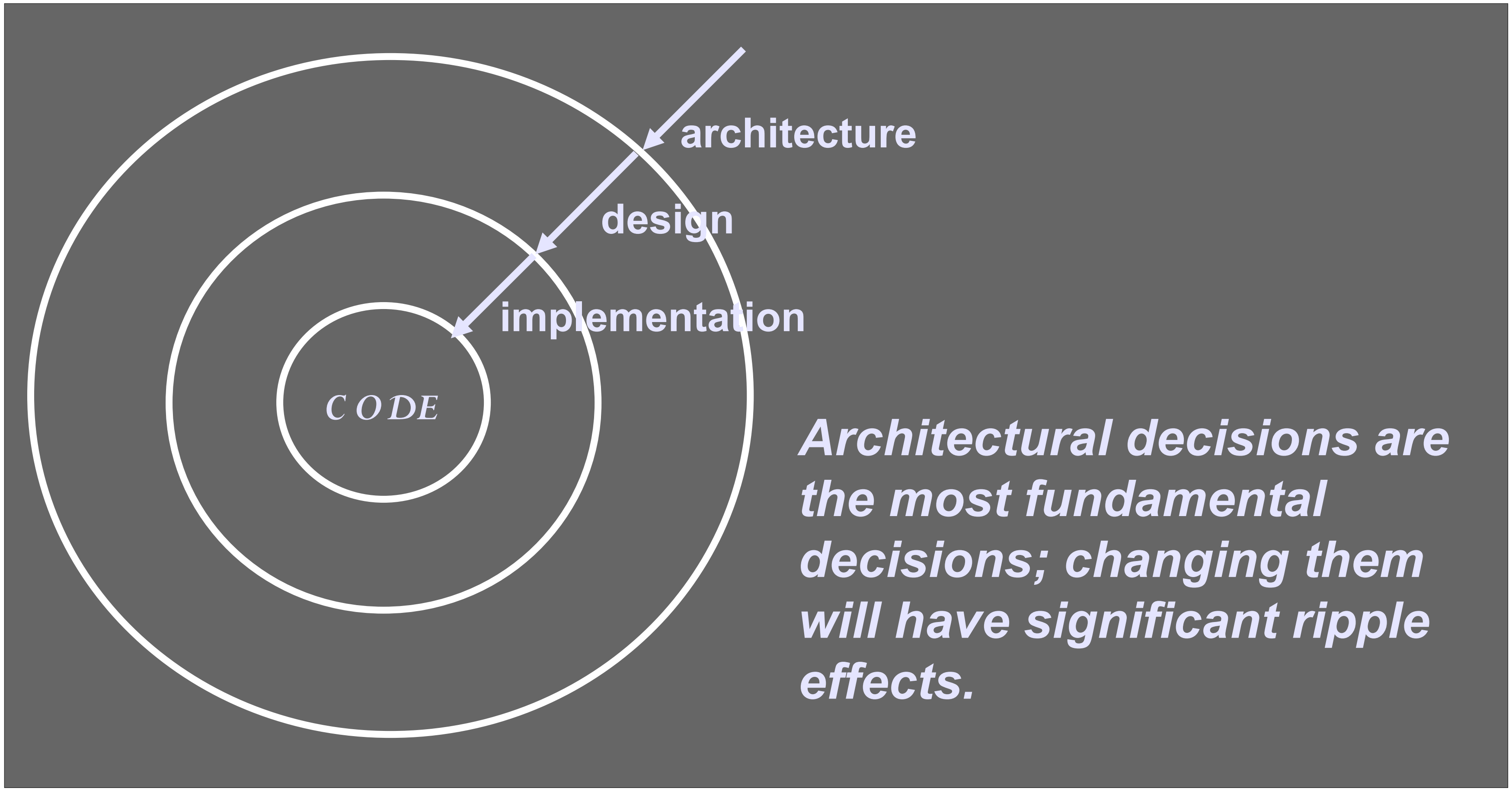
- Perry and Wolf, 1992
  - A set of architectural (or design) elements that have a particular form
- Boehm et al., 1995
  - A software system architecture comprises
    - A collection of software and system components, connections, and constraints
    - A collection of system stakeholders' need statements
    - A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements
- Clements et al., 1997
  - 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

Common elements

- Architecture defines major components
- Architecture defines component relationships (structures) and interactions
- Architecture omits content information about components that does not pertain to their interactions
- Behavior of components is a part of architecture insofar as it can be discerned from the point of view of another component
- Every system has an architecture (even a system composed of one component)
- Architecture defines the rationale behind the components and the structure
- Architecture definitions do not define what a component is
- Architecture is not a single structure -- no single structure is the architecture

Architecture defined

- Architecture establishes the context for design and implementation



Architectural decisions are the most fundamental decisions; changing them will have significant ripple effects.

Architecture defined

- IEEE 1471-2000
  - Software architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution
- Software architecture encompasses the set of significant decisions about the organization of a software system
  - Selection of the structural elements and their interfaces by which a system is composed
  - Behavior as specified in collaborations among those elements
  - Composition of these structural and behavioral elements into larger subsystems
  - Architectural style that guides this organization

Booch, Kruchten, Reitman, Bittner, and Shaw

Architectural style defined

- Style is the classification of a system's architecture according to those with similar patterns
- A pattern is a common solution to a common problem; patterns may be classified as idioms, mechanisms, or frameworks

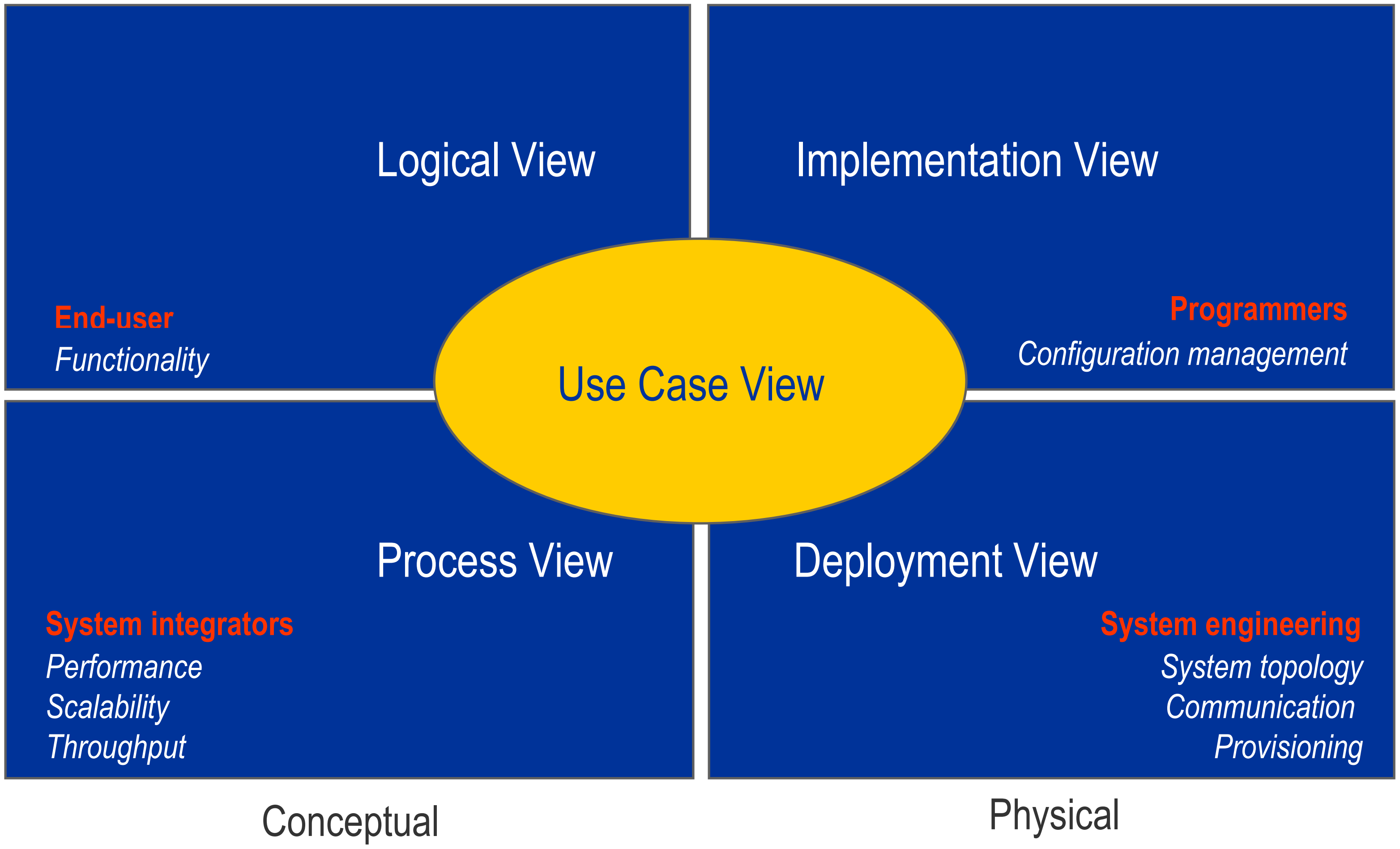
Model, views, concerns, and stakeholders

- A model is a simplification of reality, created in order to better understand the system being created; a semantically closed abstraction of a system
- A view is a representation of a whole system from the perspective of a related set of concerns
- A concern is those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders
- A stakeholder is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system

Stakeholders and views

- Architecture is many things to many different stakeholders
  - End user
  - Customer
  - Sys admin
  - Project manager
  - System engineer
  - Developer
  - Architect
  - Maintainer
  - Tester
  - Other systems
- Multiple realities, multiple views and multiple blueprints exist

Representing software architecture



Clements, et al, *Documenting Software Architectures*

Adapting views

- Not all systems require all views
  - Single process (ignore process view)
  - Small program (ignore implementation view)
  - Single processor (ignore deployment view)
- Some systems require additional views
  - Data view
  - Security view
  - Other aspects

Cross functional mechanisms

- Some structures and behaviors crosscut components
  - Security
  - Concurrency
  - Caching
  - Persistence
- Such elements usually appear as small code fragments sprinkled throughout a system
- Such elements are hard to localize using traditional approaches

Logical view

- The view of a system's architecture that encompasses the vocabulary of the problem and solution space, the collaborations that realize the system's use cases, the subsystems that provide the central layering and decomposition of the system, and the interfaces that are exposed by those subsystems and the system as a whole
- Focuses on
  - Functionality
  - Key Abstractions
  - Mechanisms
  - Separation of concerns and distribution of responsibilities

Process view

- The view of a system’s architecture that encompasses the threads and processes that form the system’s concurrency and synchronization mechanisms
- Focuses on
  - Performance
  - Scalability
  - Throughput

Implementation view

- The view of a system's architecture that encompasses the components used to assemble and release the physical system
- Focuses on
  - Configuration management

Deployment view

- The view of a system's architecture that encompasses the nodes that form the system's hardware topology on which the system executes
- Focuses on
  - Distribution
  - Communication
  - Provisioning

Use case view

- The view of a system’s architecture that encompasses the use cases that describe the behavior of the system as seen by its end users and other external stakeholders

Patterns

- A pattern is a common solution to a common problem
- A pattern codifies specific knowledge collected from experience in a domain
- A pattern resolves forces in context
- All well-structured systems are full of patterns
  - Idioms
  - Mechanisms
  - Frameworks

http://www.hillside.net

Mechanisms

- Mechanisms (design patterns) are the soul of an architecture
- Gang of Four patterns
  - Creational patterns
  - Structural patterns
  - Behavioral patterns

Gamma, et al *Design Patterns*

Frameworks

- Frameworks (architectural patterns) provide an extensible template for applications within a domain
- Shaw/Garlan patterns
  - Dataflow systems
    - Batch sequential
    - Pipes and filters
  - Call-and-return systems
    - Main program and subroutine
    - OO systems
    - Hierarchical layers
  - Independent components
    - Communicating processes
    - Event systems
  - Virtual machines
    - Interpreters
    - Rule-based systems
  - Data-centered systems
    - Databases
    - Hypertext systems
    - Blackboards

Shaw et al, *Software Architecture*

Architectural design

- 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.

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.
  - *Experiment we carried on at UCL*
- Large-scale reuse
  - The architecture may be reusable across a range of systems.

Architecture and system characteristics

- Performance
  - Localise critical operations and minimise communications. Use large rather than fine-grain components.
- Security
  - Use a layered architecture with critical assets in the inner layers.
- Safety
  - Localise safety-critical features in a small number of sub-systems.
- Availability
  - Include redundant components and mechanisms for fault tolerance.
- Maintainability
  - Use fine-grain, replaceable components.

Architectural conflicts

- Using large-grain components improves performance but reduces maintainability.
- Introducing redundant data improves availability but makes security more difficult.
- Localising safety-related features usually means more communication so degraded performance.

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.

Architectural design decisions

- Architectural design is a creative process so the process differs depending on the type of system being developed.
- However, a number of common decisions span all design processes.

Architectural styles

- The architectural model of a system may conform to a generic architectural model or style.
- An awareness of these styles can simplify the problem of defining system architectures.
- However, most large systems are heterogeneous and do not follow a single architectural style.

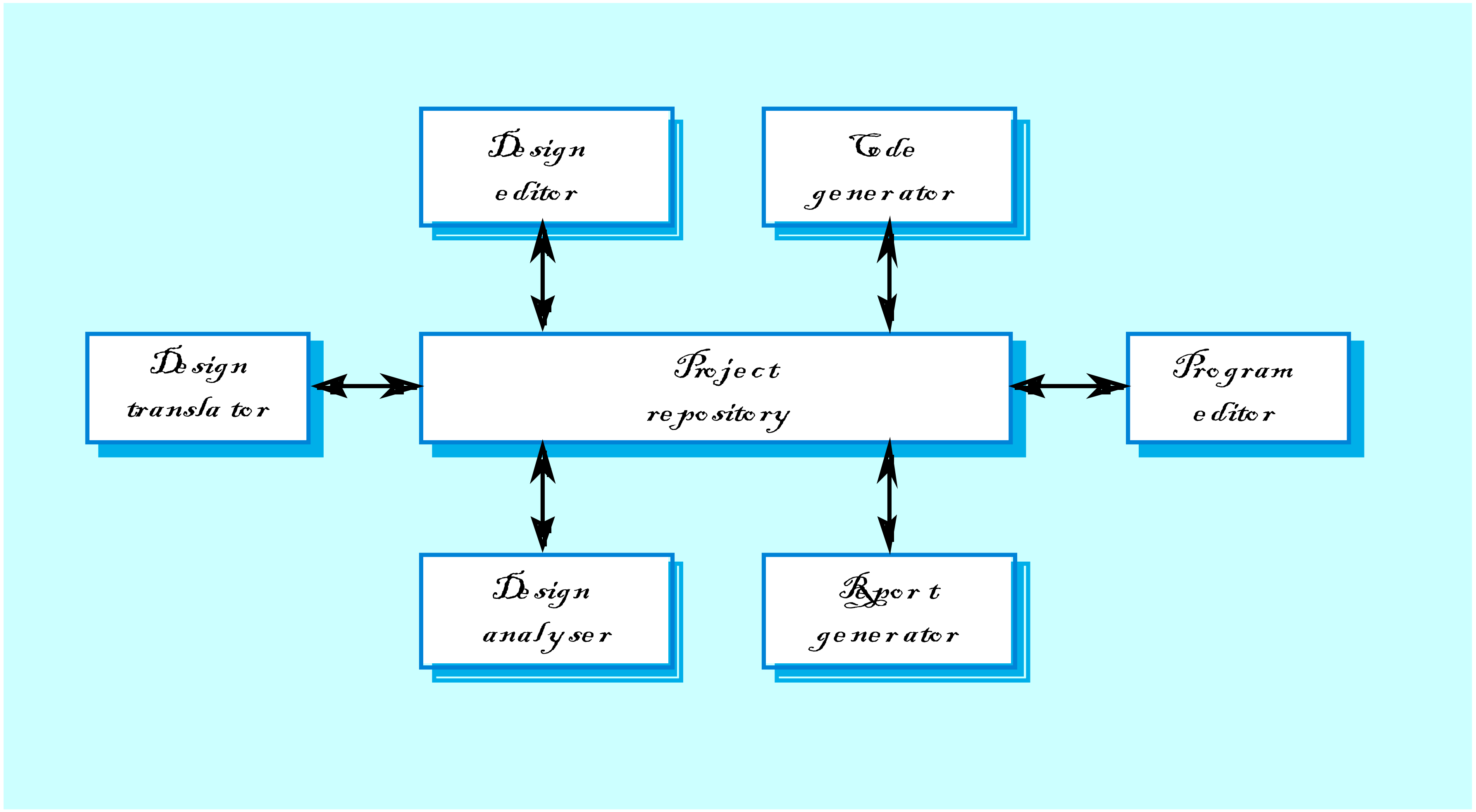
System organisation

- Reflects the basic strategy that is used to structure a system.
- Three organisational styles are widely used:
  - A shared data repository style;
  - A shared services and servers style;
  - An abstract machine or layered style.

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



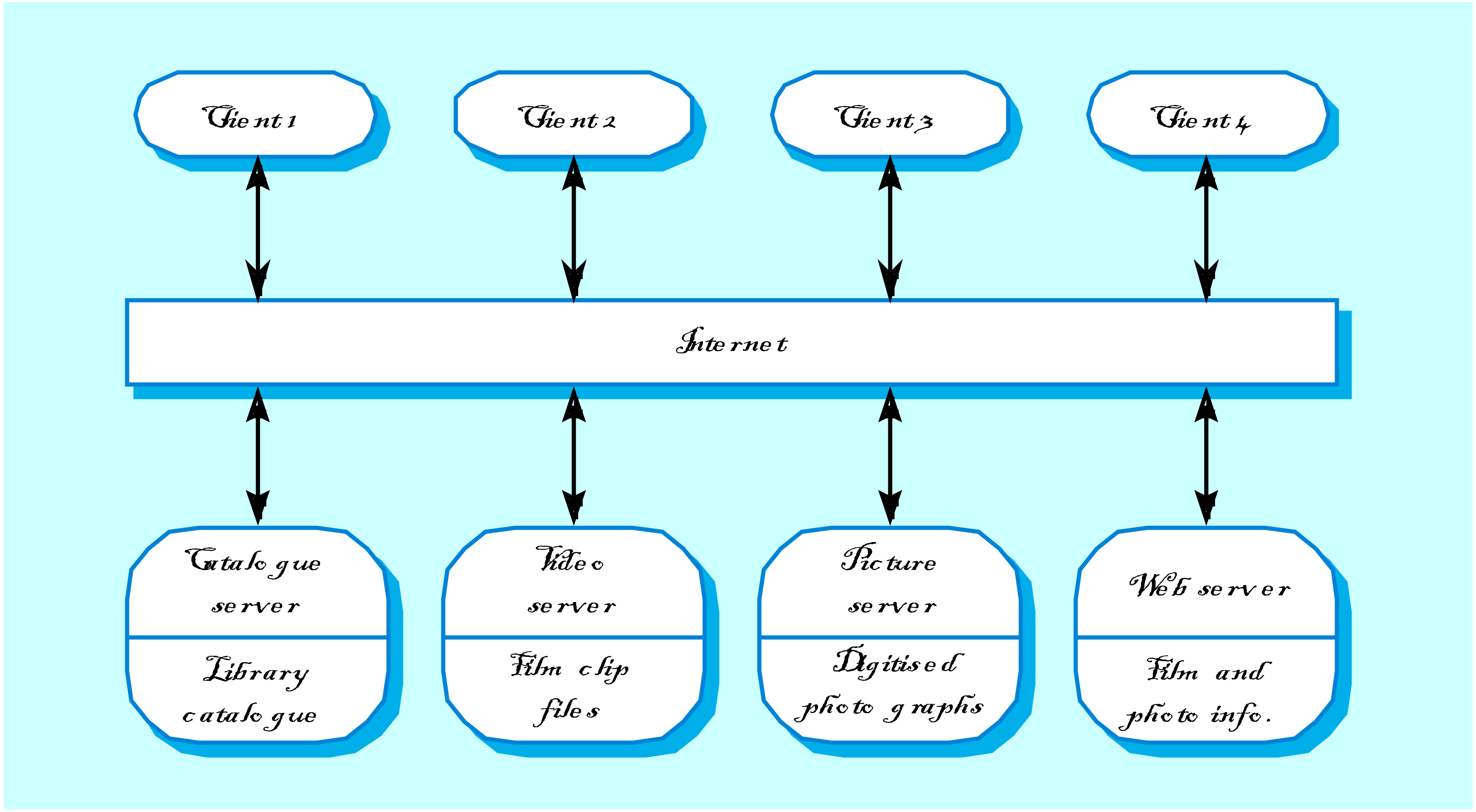
Repository model characteristics

- 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 model

- 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



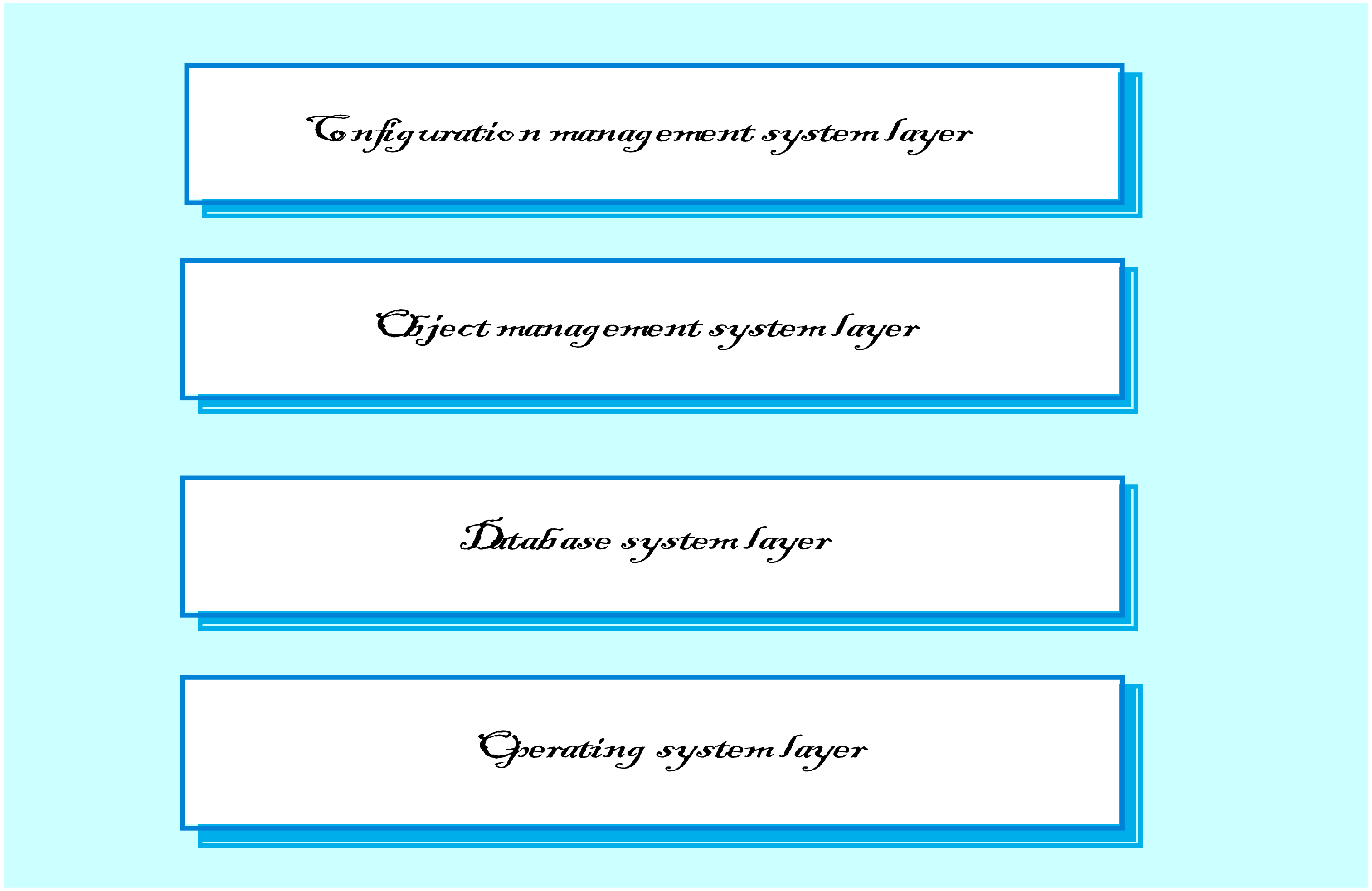
Client-server characteristics

- 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 (layered) 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 artificial to structure systems in this way.

Version management system


Configuration management system layer
Object management system layer
Database system layer
Operating system layer

Modular decomposition styles

- Styles of decomposing sub-systems into modules.
- No rigid distinction between system organisation and modular decomposition.

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.

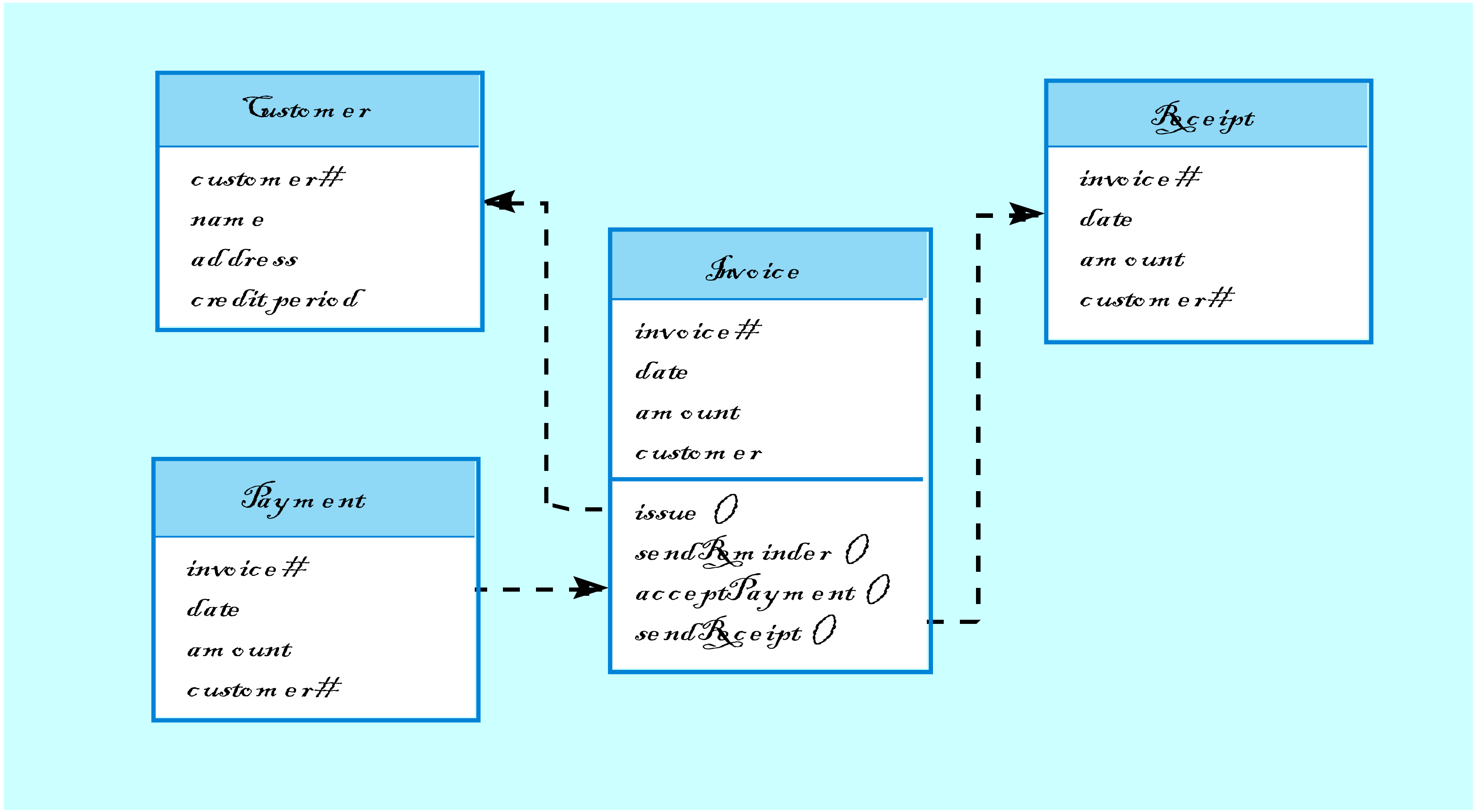
Modular decomposition

- 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 object;
  - A pipeline or 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



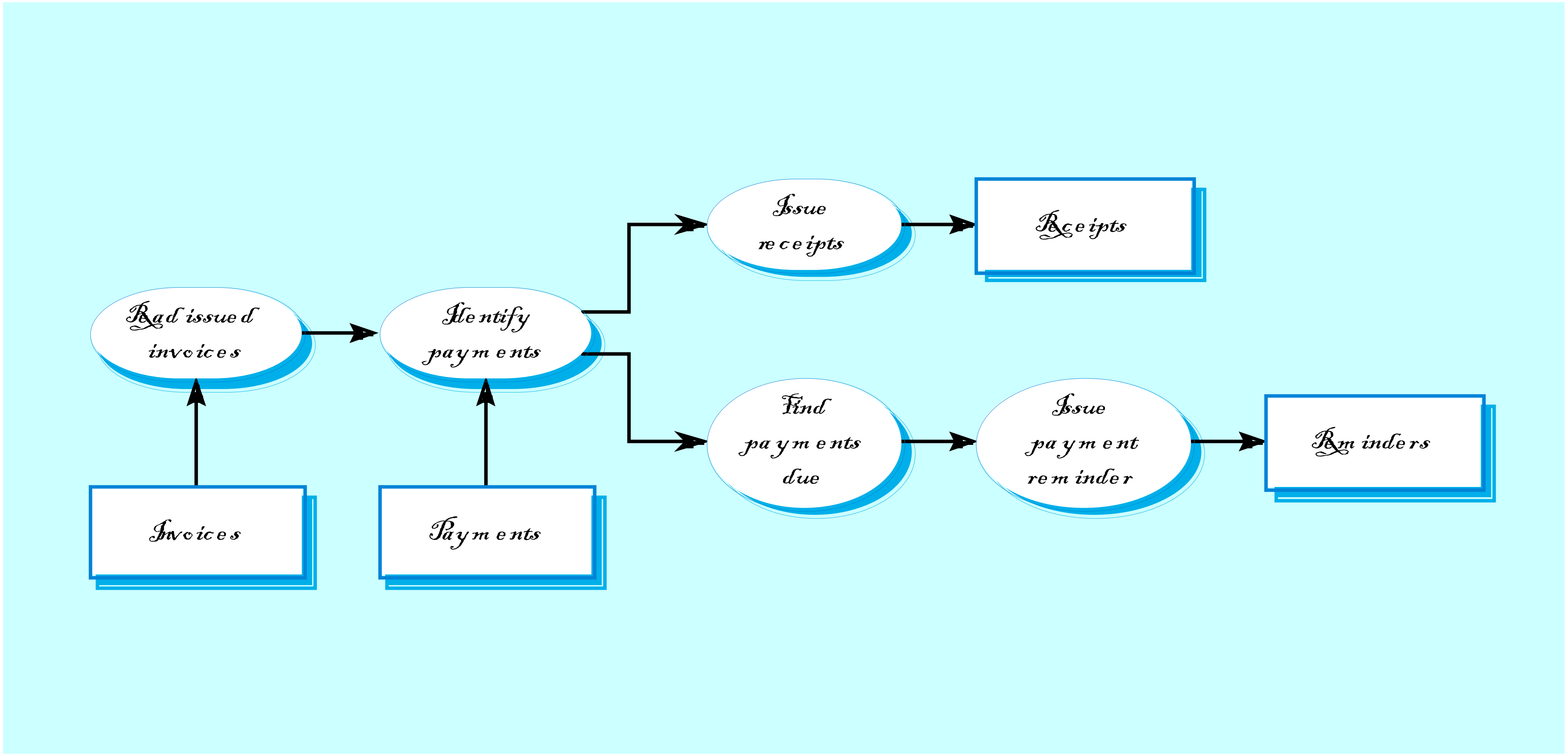
Object model advantages

- Objects are loosely coupled so their implementation can be modified without affecting other objects.
- The objects may reflect real-world entities.
- OO implementation languages are widely used.
- However, object interface changes may cause problems and complex entities may be hard to represent as objects.

Function-oriented pipelining

- 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



Pipeline model advantages

- Supports transformation reuse.
- Intuitive organisation for stakeholder communication.
- Easy to add new transformations.
- Relatively simple to implement as either a concurrent or sequential system.
- However, requires a common format for data transfer along the pipeline and difficult to support event-based interaction.

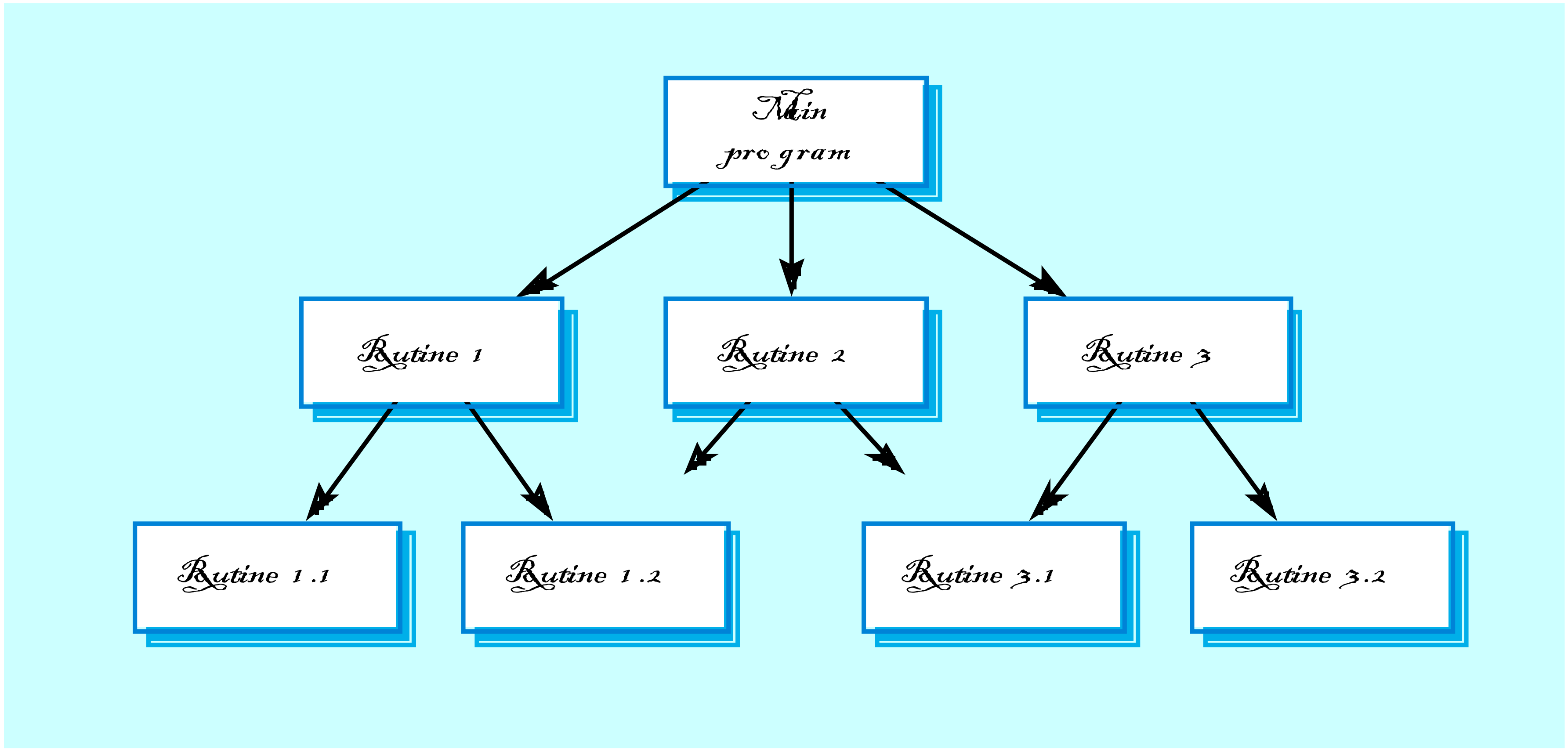
Control styles

- 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.

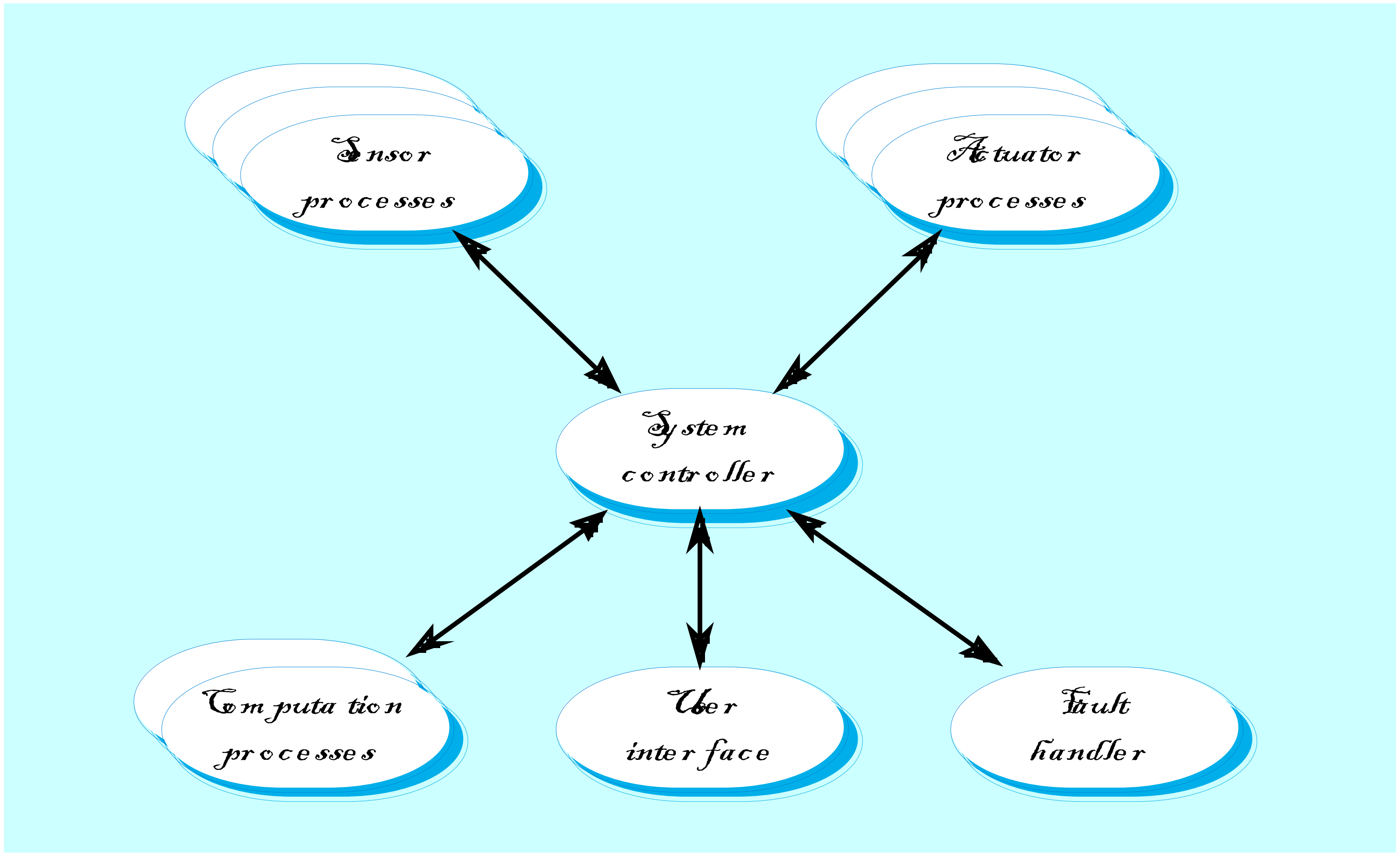
Centralised control

- 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. 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



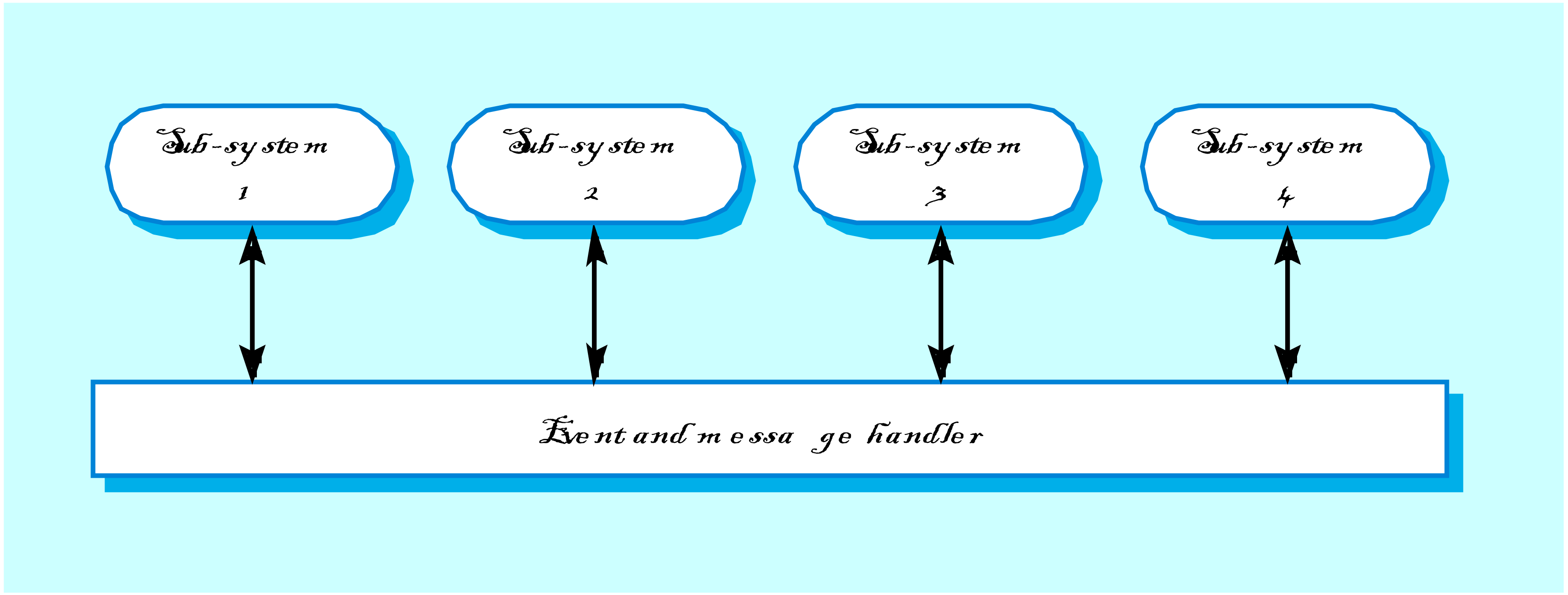
Event-driven systems

- Driven by externally generated events where the timing of the event is outwith the control of the sub-systems which process the event.
- 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.

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.

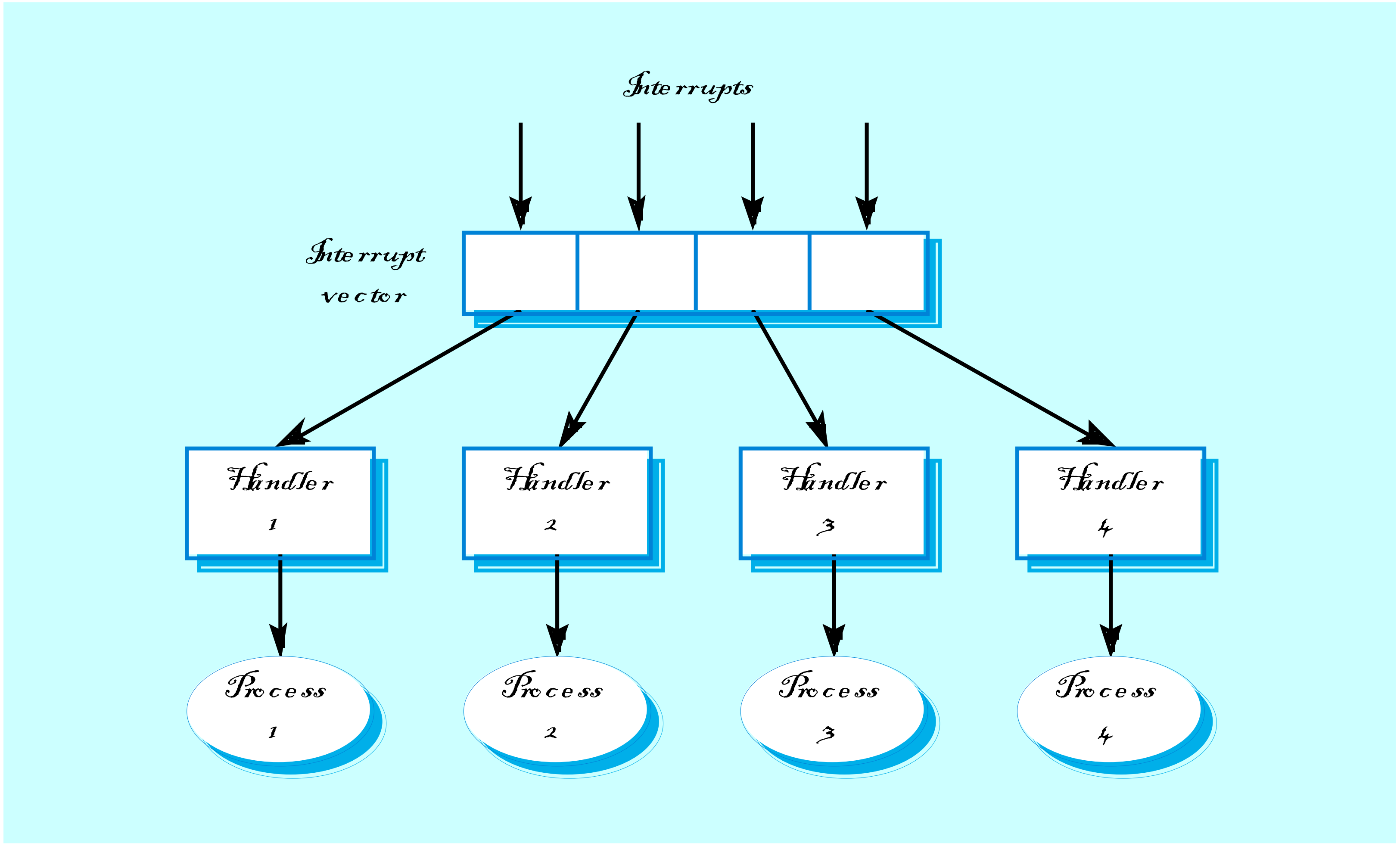
Selective broadcasting



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



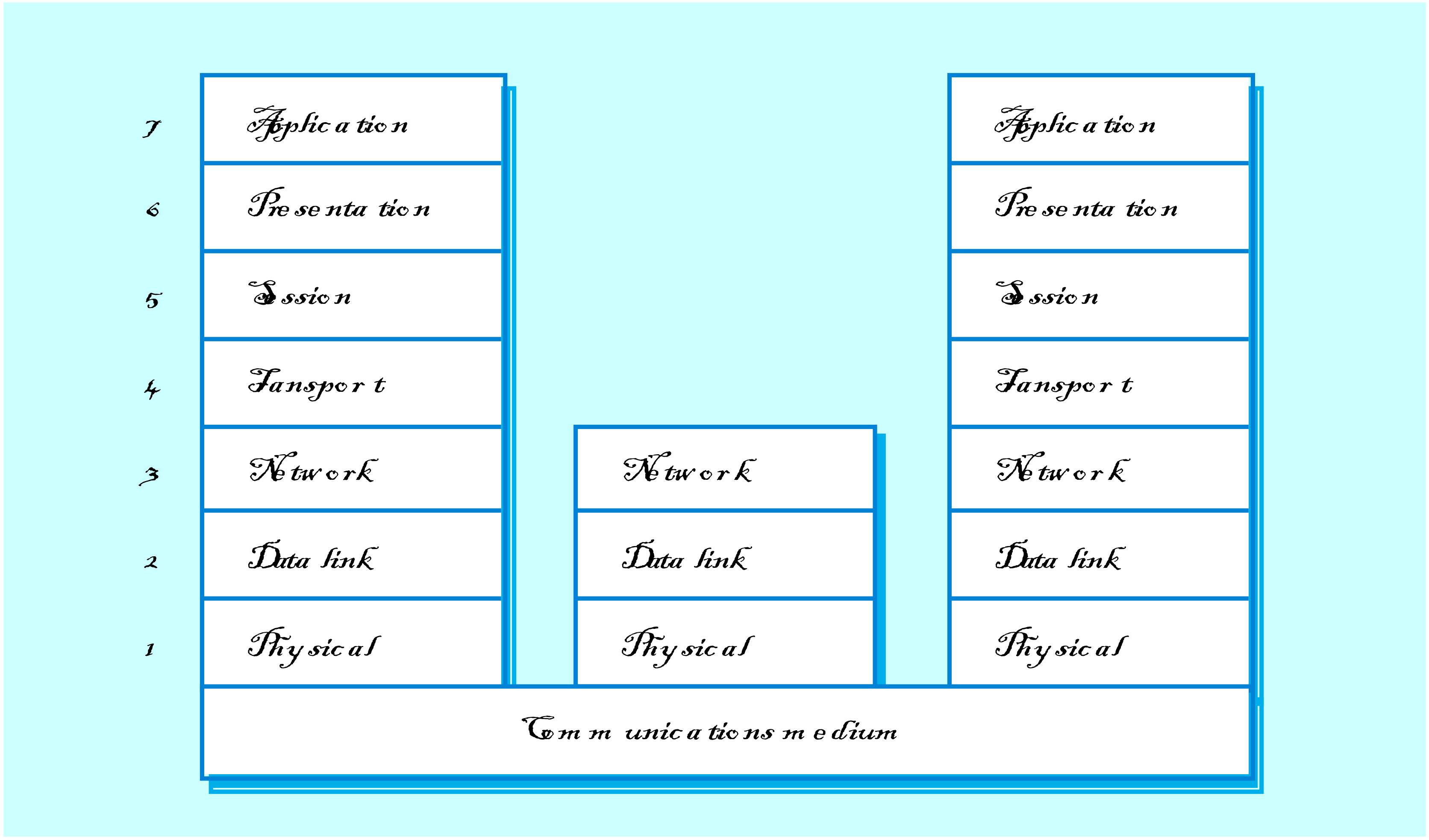
Reference architectures

- Architectural models may be 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. Covered in Chapter 13.
  - 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.

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


7
6
5
4
3
2
1
Application
Presentation
Session
Transport
Network
Data link
Physical
Network
Data link
Physical
Application
Presentation
Session
Transport
Network
Data link
Physical
Communications medium