System Design

Elements of System Design

Software Architectures

Concurrency

Processor Allocation

Data Management Issues

Development Standards

Design Tradeoffs & Implementation Issues

Elements of System Design (1)

- System architecture: overall structure, relationships among its major components and their interactions
  - Software architecture: the structure of software elements
  - Architectural decision determine success in meeting non-functional requirements
  - Poor architecture reduces reusability of designed and existing components

Elements of System Design (2)

- Activities
  - Identification of sub-systems and major components
  - Inherent concurrency
  - Allocation of sub-systems to processors
  - Data management strategy
  - HCI standards and strategy
  - Code development standards
  - Planning of control aspects
  - Test plans
  - Setting of priorities for design tradeoffs
  - Identification of implementation requirements

Software Architecture (1)

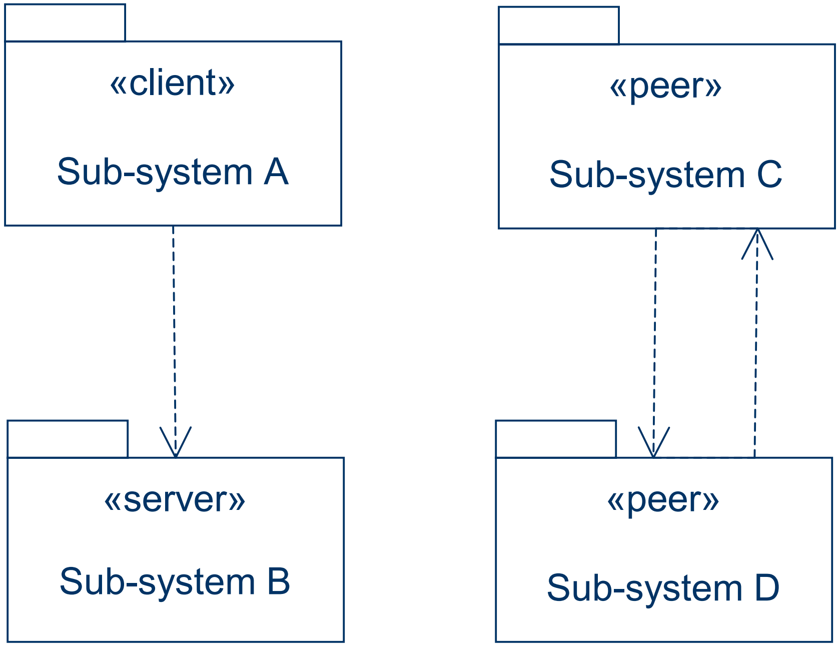
- Description of sub-systems and components and the relationships between them, typically specified in different views to show relevant functional and non-functional properties
- Aspects of software architecture
  - Conceptual architecture: components and connectors
  - Module architecture: sub-systems, modules and exports, imports
  - Code architecture: files, directories, libraries and includes, contains
  - Execution architecture: tasks, threads, object interactions and uses, calls
- Logical versus physical architecture

Software Architecture (2)

- Sub-systems group together elements of the system that share some common properties
  - A coherent set of responsibilities
- Advantages of sub-systems
  - Smaller units of deployment
  - Maximise reuse at the component level
  - Helps in copying with complexity
  - Improves maintainability
  - Aids portability

Software Architecture (3)

- Clear boundary – interface (encapsulation of internal structure)
  - Remember the contracts in contract-based design!
  - The goal is independence of sub-systems – incremental development and delivery
  - Localise changes
- Sub-system communication styles
  - Client/server easier to implement and maintain – less tighter coupling

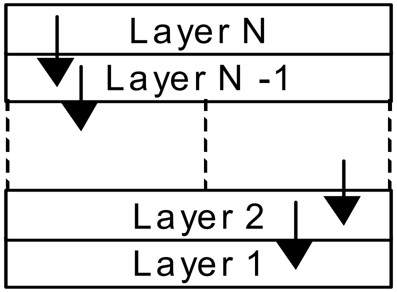


The server sub-system does not depend on the client sub-system and is not affected by changes to the client's interface.

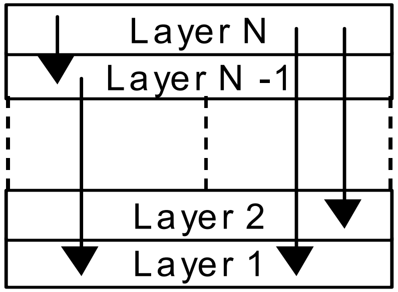
Each peer sub-system depends on the other and each is affected by changes in the other's interface.

Software Architecture (4)

- Layering and partitioning
  - Layers – different levels of abstraction
    - Open versus closed
  - Partitions – different aspects of functionality
  - Usually combined

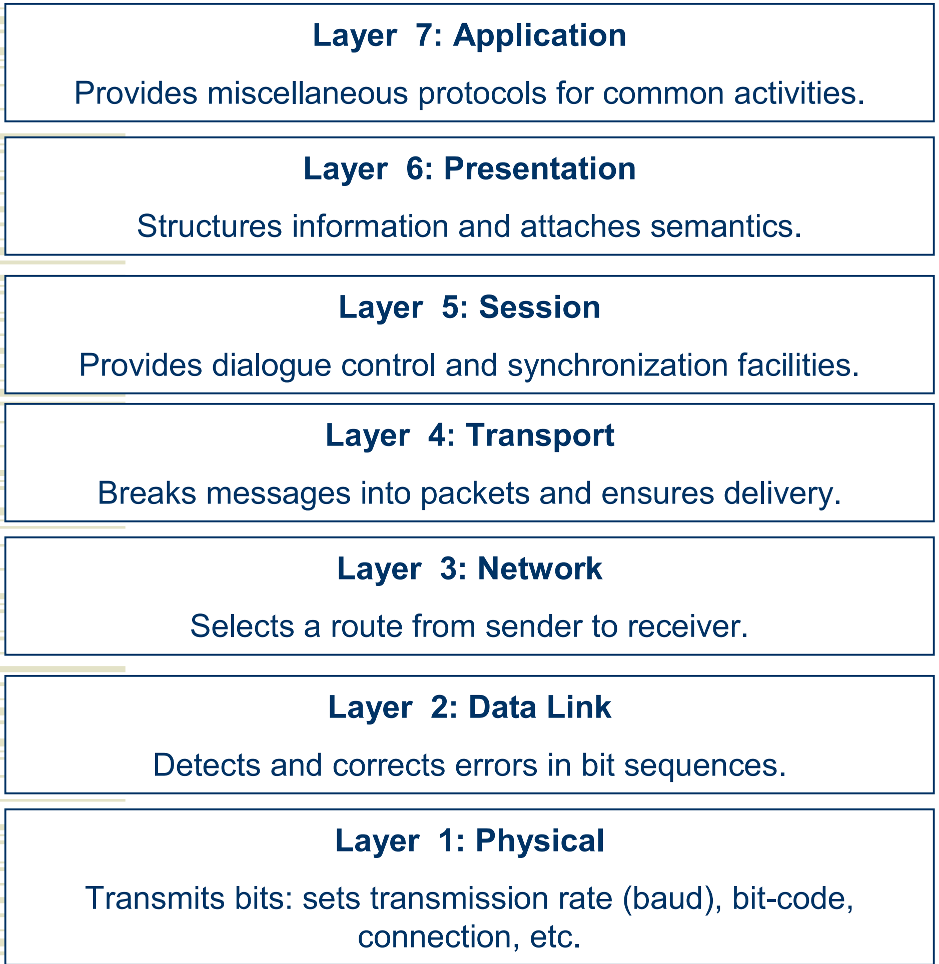


Closed architecture—messages may only be sent to the adjacent lower layer.



Open architecture—messages can be sent to any lower layer.

Software Architecture (5)



Layer 7: Application

Provides miscellaneous protocols for common activities.

Layer 6: Presentation

Structures information and attaches semantics.

Layer 5: Session

Provides dialogue control and synchronization facilities.

Layer 4: Transport

Breaks messages into packets and ensures delivery.

Layer 3: Network

Selects a route from sender to receiver.

Layer 2: Data Link

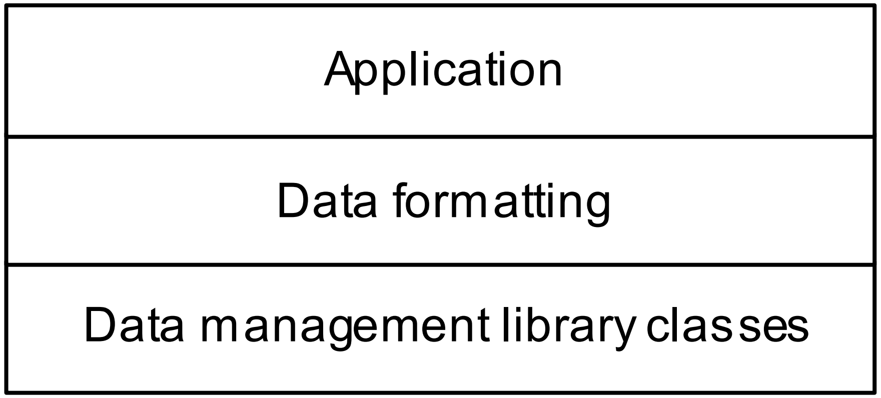
Detects and corrects errors in bit sequences.

Layer 1: Physical

Transmits bits: sets transmission rate (baud), bit-code, connection, etc.

- Issues for layered architectures
  - Layer interface stability
  - Sharing of lower layers between systems
  - Appropriate level of granularity
  - Sub-division of complex layers
  - Performance overhead of closed layer architectures

Software Architecture (5)



Application

Data formatting

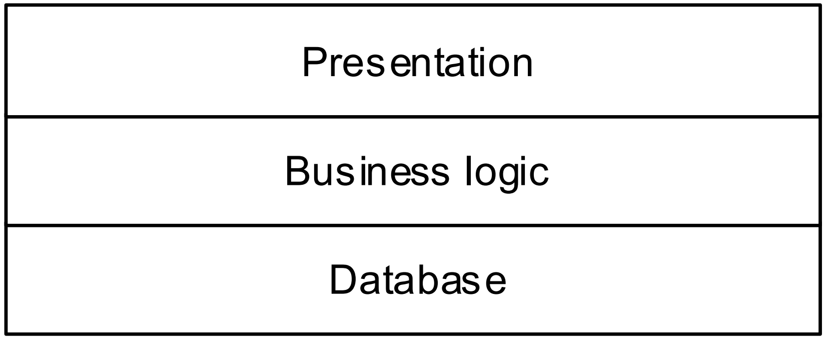
Data management library classes

- Issues for layered architectures
  - Layer interface stability
  - Sharing of lower layers between systems
  - Appropriate level of granularity
  - Sub-division of complex layers
  - Performance overhead of closed layer architectures

Software Architecture (6)

- Process for layered architecture development
  - Define criteria for grouping application functionality into layers
  - Determine number of layers
  - Name layers and assign functionality to them
  - Refine the produced structure
  - Specify the interface of each layer
  - Specify the structure of each layer – partitioning?
  - Specify the communication between layers
  - Reduce coupling between layers – strong encapsulation

Software Architecture (7)



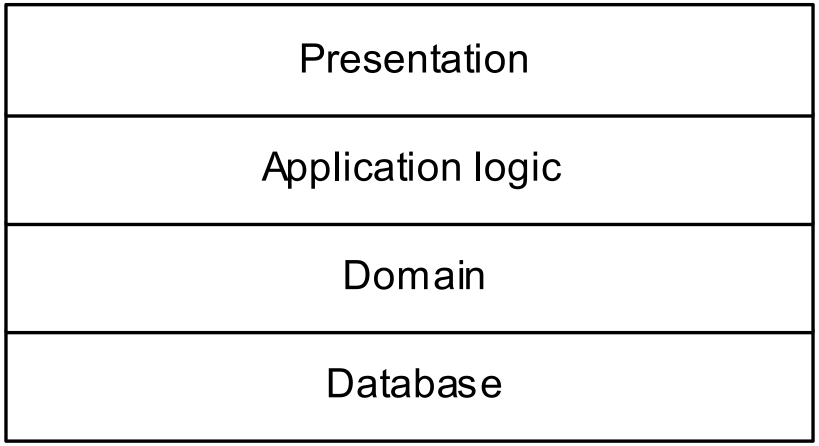
Presentation

Business logic

Database

Layers versus Tiers

Logical versus Physical division



Presentation

Application logic

Domain

Database

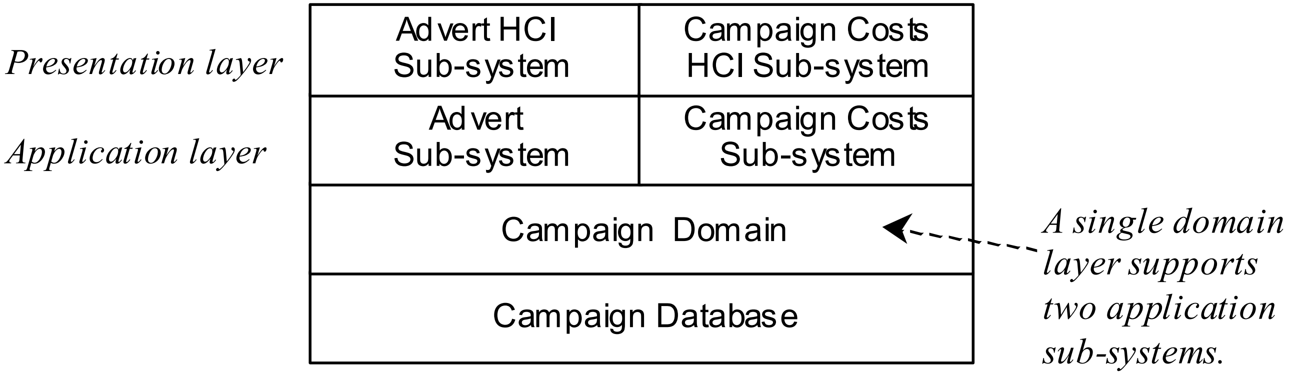
Boundary classes

Control classes

Entity classes

Software Architecture (8)

- Partitioning – different aspects of functionality



Presentation layer

Application layer

Advert HCI Sub-system

Campaign Costs HCI Sub-system

Advert Sub-system

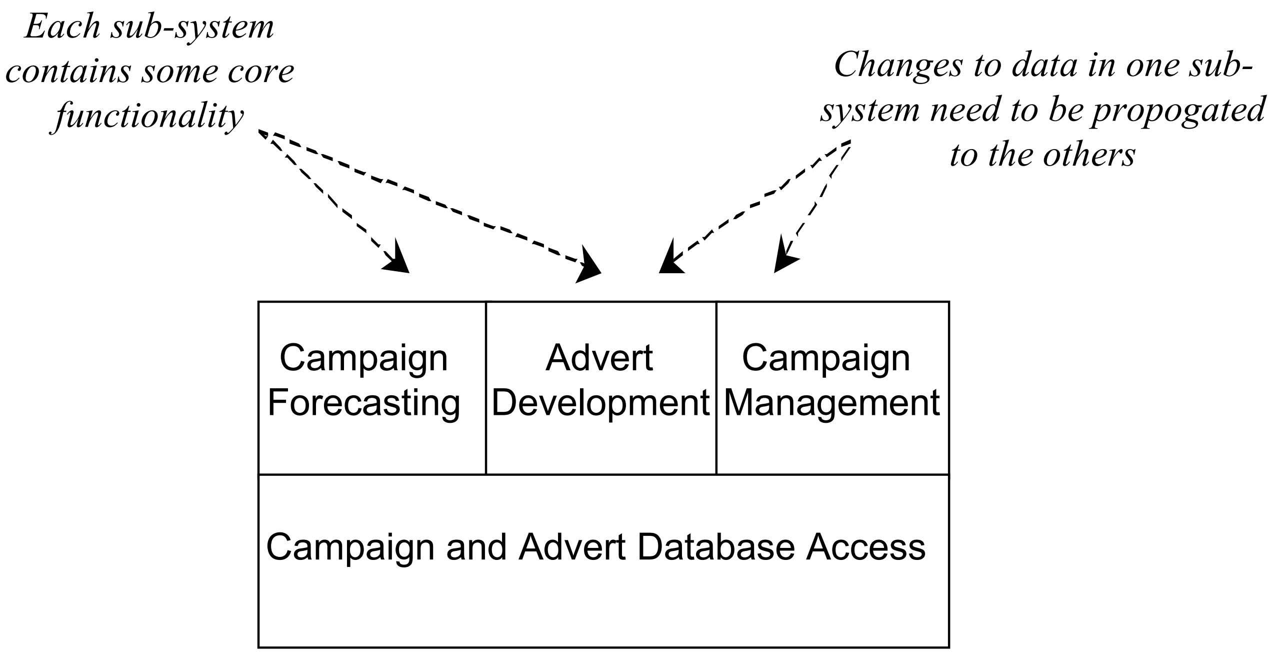
Campaign Costs Sub-system

Campaign Domain

Campaign Database

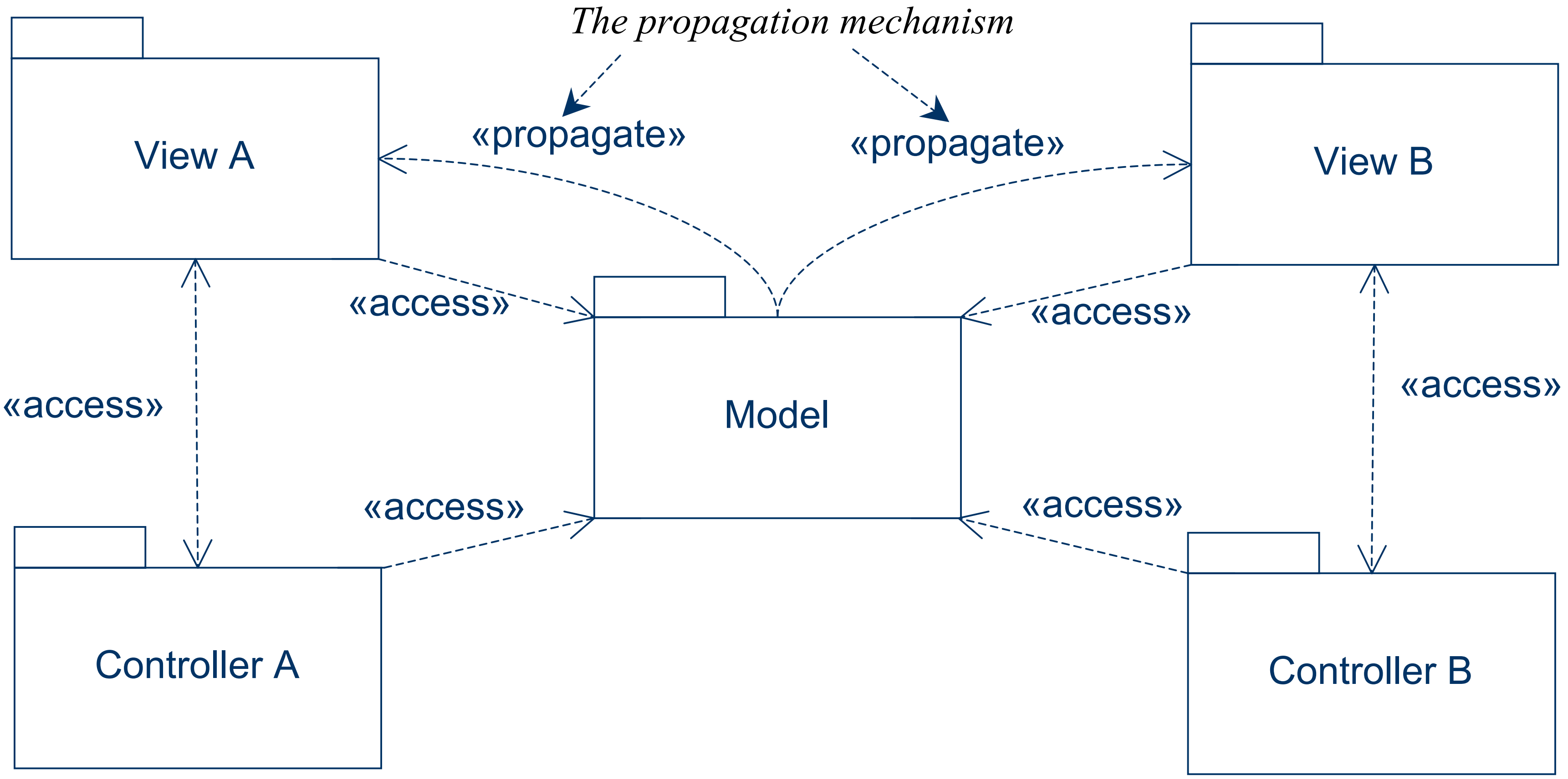
A single domain layer supports two application sub-systems.

Software Architecture (9)

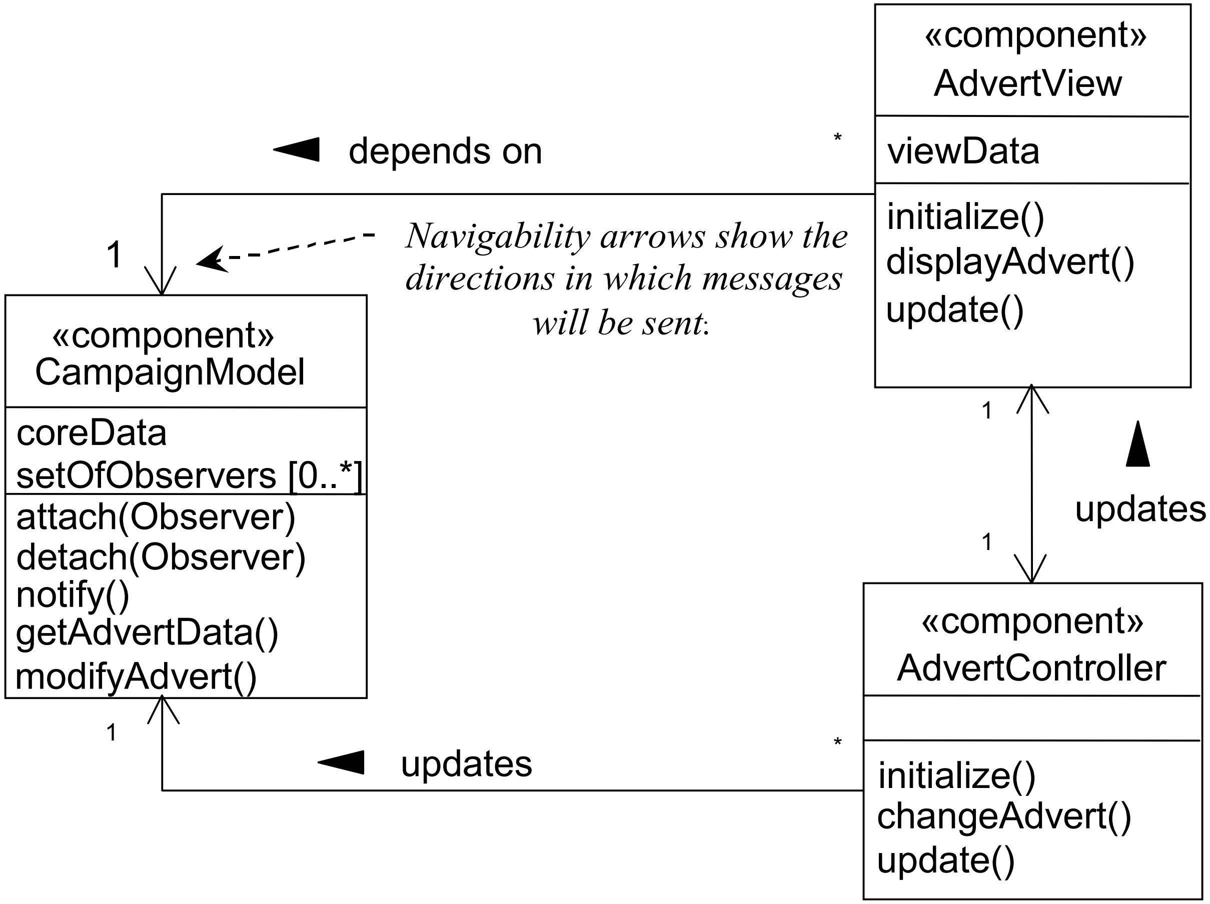

Each sub-system contains some core functionality

Changes to data in one sub-system need to be propogated to the others

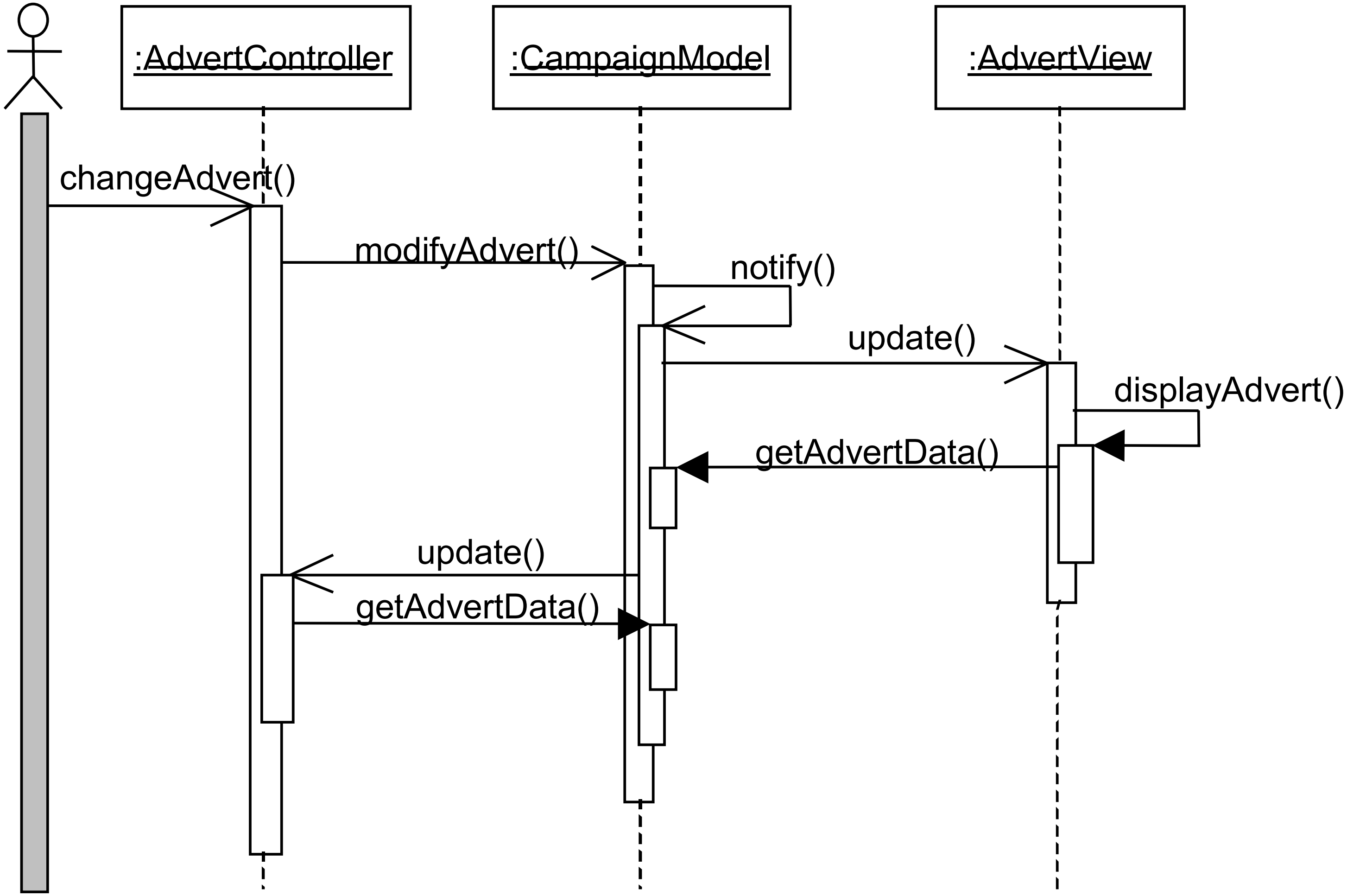
Software Architecture (10)



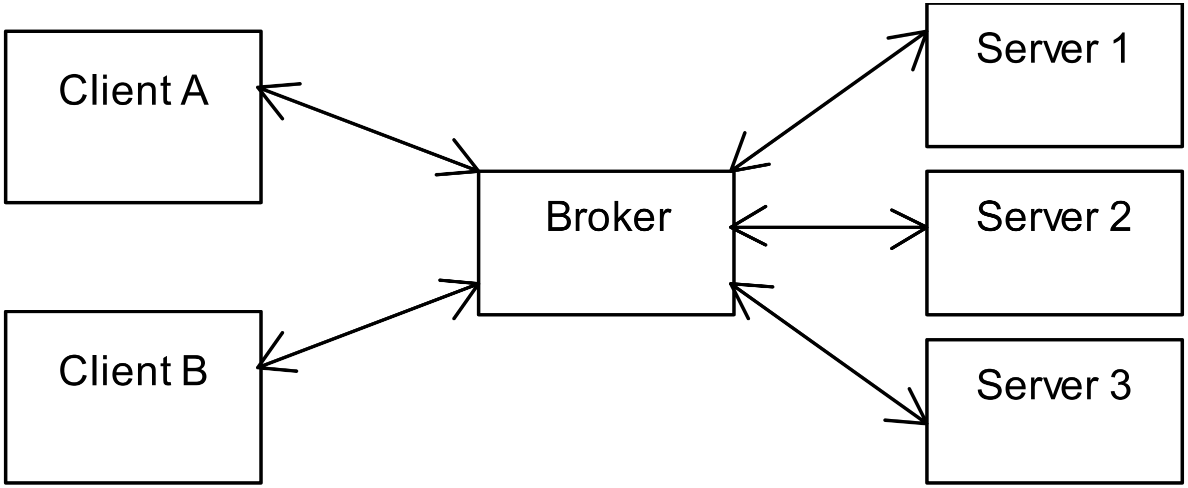
Software Architecture (11)



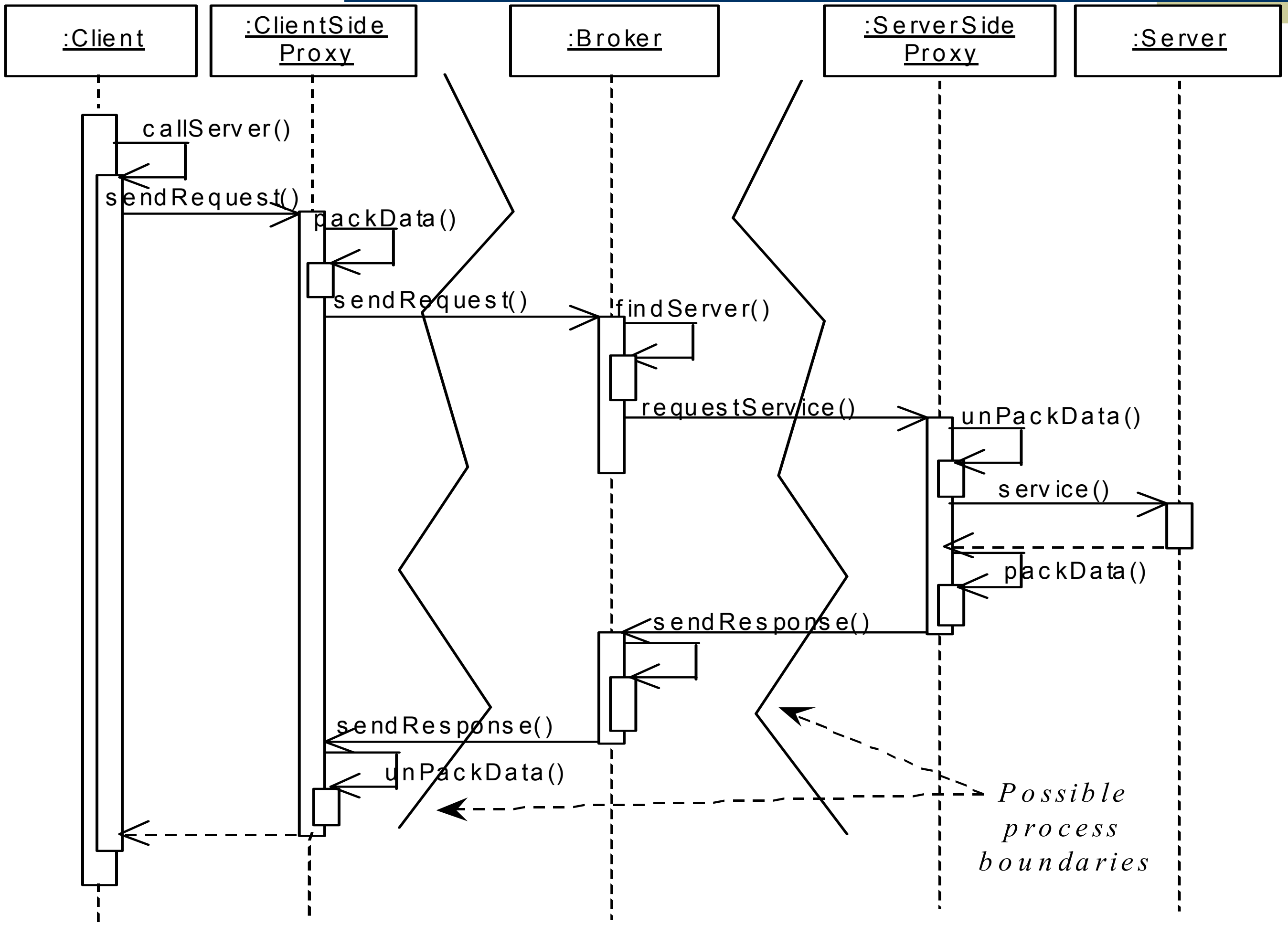
Software Architecture (12)



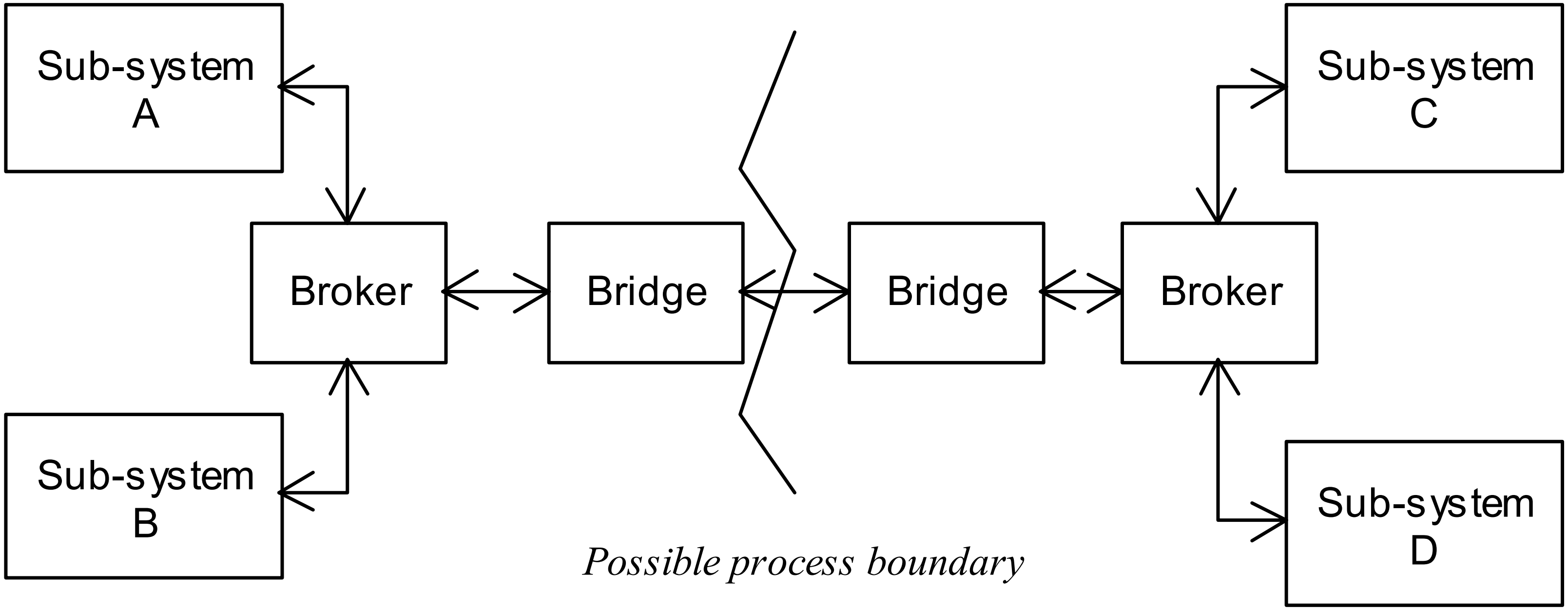
Software Architecture (13)



Software Architecture (14)



Software Architecture (15)



Possible process boundary

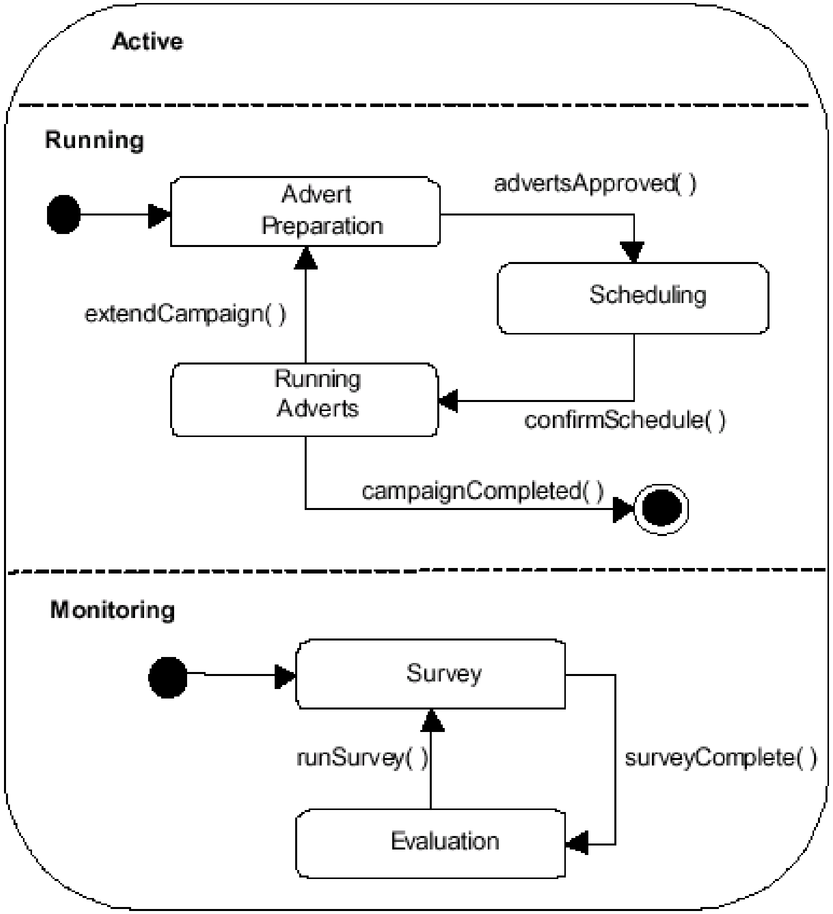
Software Architecture (16)

- Conway’s law for organisation structure and architecture
  - Development teams need to be aligned with architecture sub-systems
  - Division and coalescence of teams or sub-systems
  - Interfaces are critical

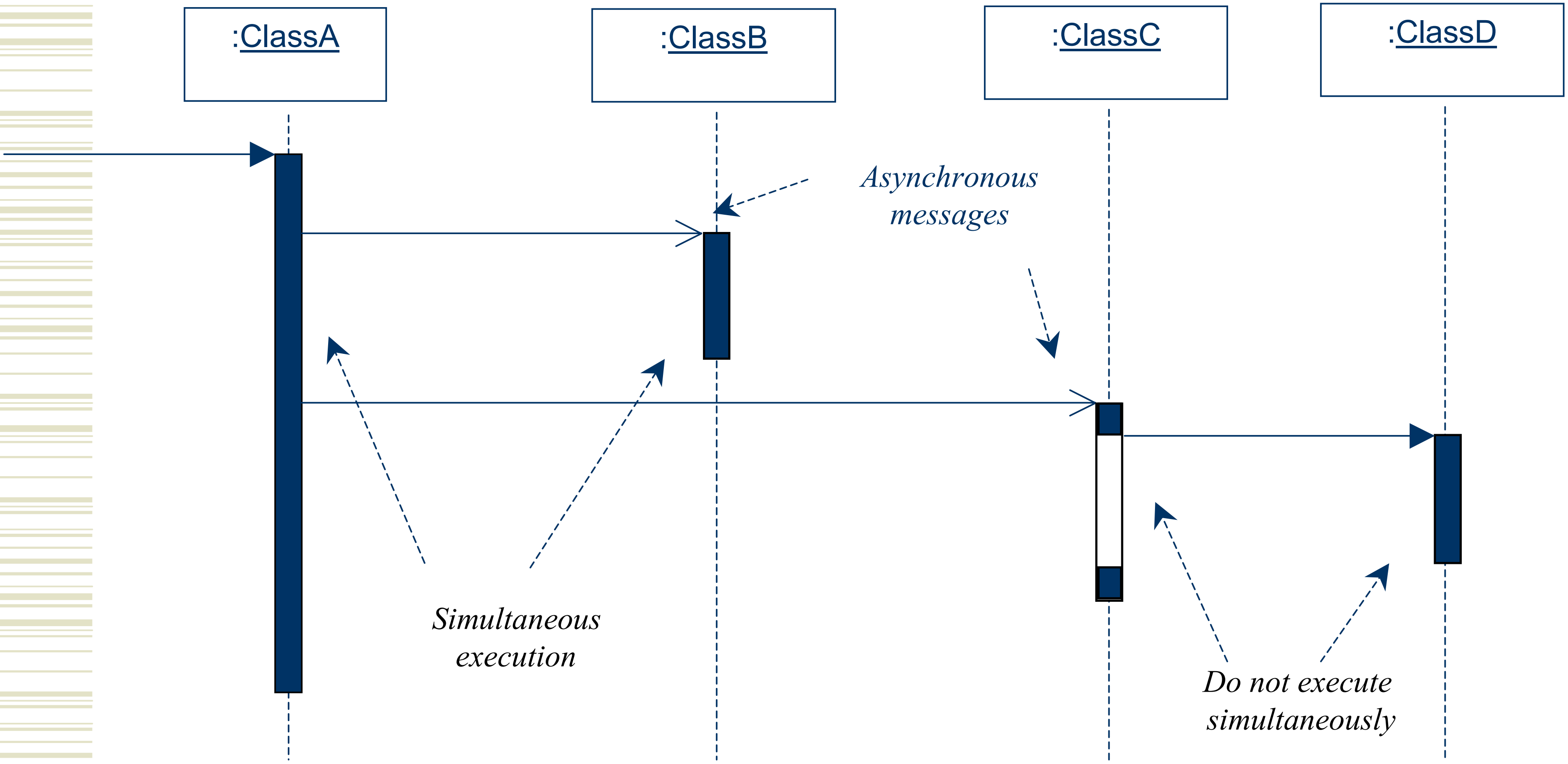
Concurrency (1)

- Logical versus physical concurrency
  - Multiple or single processors
  - Multi-user DBMS, Multitasking OS, multi-threading language
- Identifying need for concurrency
  - Use cases – simultaneous response to different events triggering different execution threads
  - Statecharts – concurrent sub-states in complex nested states
  - Sequence diagrams – simultaneous activation (method execution) for different objects

Concurrency (2)



Concurrency (3)

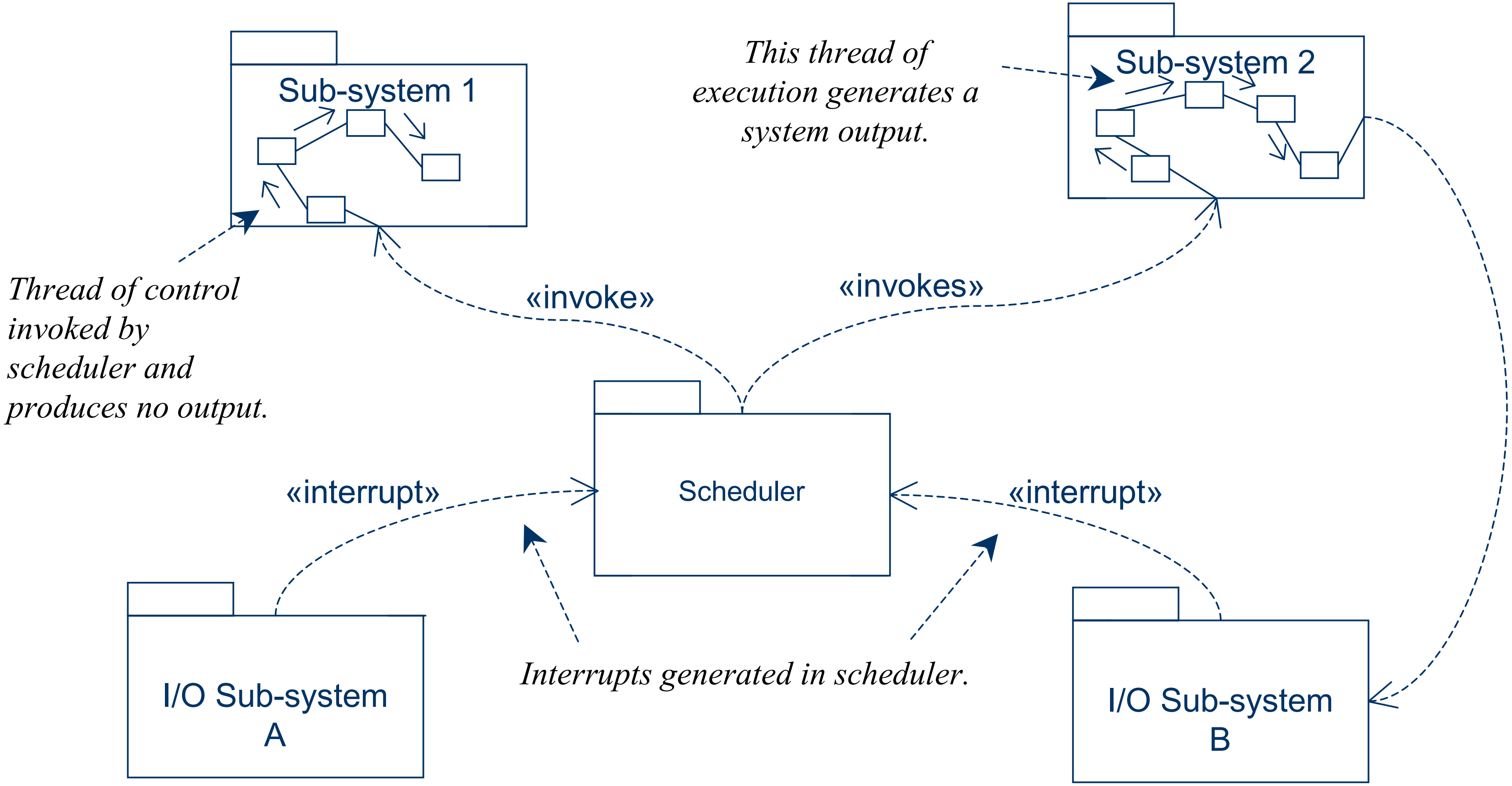


Asynchronous messages

Simultaneous execution

Do not execute simultaneously

Concurrency (4)



Thread of control invoked by scheduler and produces no output.

This thread of execution generates a system output.

Interrupts generated in scheduler.

Processor Allocation

- Divide application into subsystems
- Estimate processing requirements for each subsystem
- Determine access criteria and location requirements
- Identify concurrency requirements for the subsystems
- Allocate each subsystem to an operating platform
- Determine communication requirements between subsystems
- Specify communication infrastructure

Data Management Issues

- Files versus DBMS
  - Simple data management and fast access, but complex data storage and retrieval code versus heavyweight system with a lot of additional functionality
  - What kind of DBMS?
    - Relational, Object-Oriented, Object-Relational

Additional Considerations

- Development Standards
  - HCI guidelines
    - Good dialogue design and standardised “look and feel”
  - I/O device guidelines
    - Standard interaction interface, encapsulation access
    - Take advantage of polymorphism
  - Construction guidelines
    - Naming conventions, use of particular software features
    - Consistency is paramount!
- Prioritising design tradeoffs
  - Aim consistency of designs at different stages
  - Guidelines agreed with clients
  - There are always unanticipated cases!
- Design for implementation
  - System initialisation issues, data conversion
  - Particular care is need to maintain the integrity of the data