Chap. 13
System Design and Architecture

Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill

Topics Covered

- What Do We Mean by Architecture?
- Architectural Style

System Architecture

- Software architects undertake the following activities:
  - Big picture is addressed
  - Subsystems and major components are identified
  - Any inherent concurrency is identified
  - A data management strategy is selected
  - A strategy and standards for human–computer interaction are chosen

Architectural Styles

- Ways to design systems that conform to the prevailing fashion
  - How to divide a system into subsystems
    - Layering and partitioning
    - Model-View-Controller (MVC)

Subsystems

- A subsystem typically groups together elements of the system that share some common properties
- Examples
  - Human-computer interface
  - Data management
  - Campaign management
  - Staff management

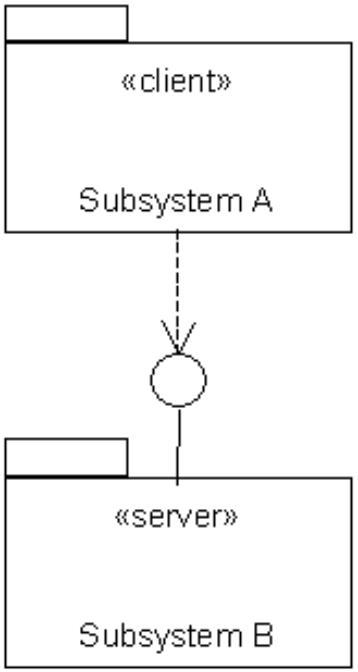
Subsystems

- The subdivision of an information system into subsystems has the following advantages
  - It produces smaller units of development
  - It helps to maximize reuse at the component level
  - It helps the developers to cope with complexity
  - It improves maintainability
  - It aids portability

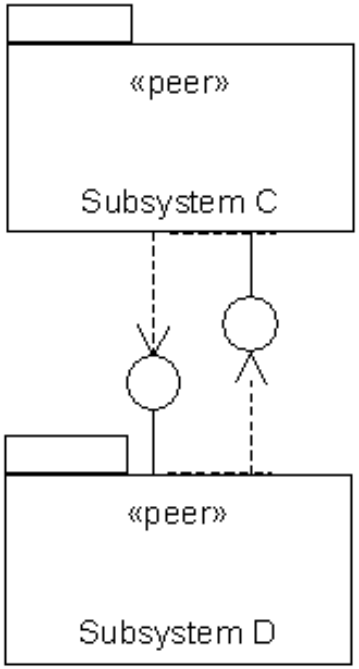
Subsystems

- Each subsystem provides services for other subsystems, and there are two different styles of communication that make this possible
- These are known as *client–server* and *peer-to-peer* communication

Styles of communication between s ubsystems



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



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

Client–server communication

- Client–server communication requires the client to know the interface of the server subsystem, but the communication is only in one direction
- The client subsystem requests services from the server subsystem and not vice versa

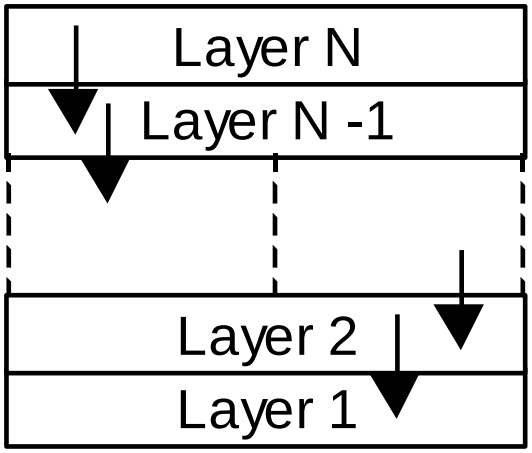
Peer-to-peer communication

- Peer-to-peer communication requires each subsystem to know the interface of the other, thus coupling them more tightly
- The communication is two way since either peer subsystem may request services from the other

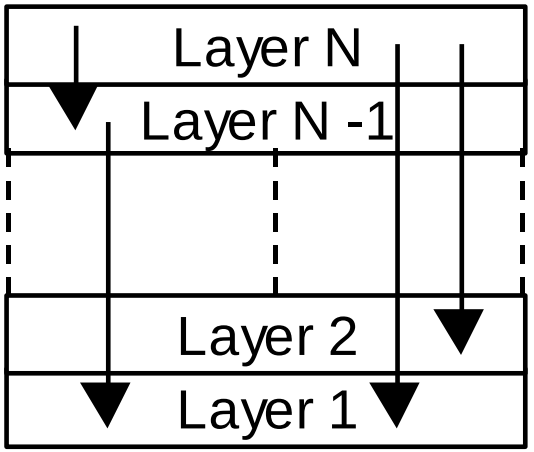
Layering and Partitioning

- Two general approaches to the division of a software system into subsystems
  - *Layering*—so called because the different subsystems usually represent different levels of abstraction
  - *Partitioning*, which usually means that each subsystem focuses on a different aspect of the functionality of the system as a whole
- Both approaches are often used together on one system

Schematic of a Layered Architecture



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



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

Layered Architecture

- Closed architecture
  - Keeps the encapsulation of the layers
    - minimizes dependencies between the layers
    - reduces the impact of a change to the interface of any one layer
  - Reduce the system performance
- Open architecture
  - produces more compact code
    - access services of all lower level layers directly by any layer above them without the need for extra program code to pass messages through each intervening layer
  - Improve the system performance
  - breaks the encapsulation of the layers

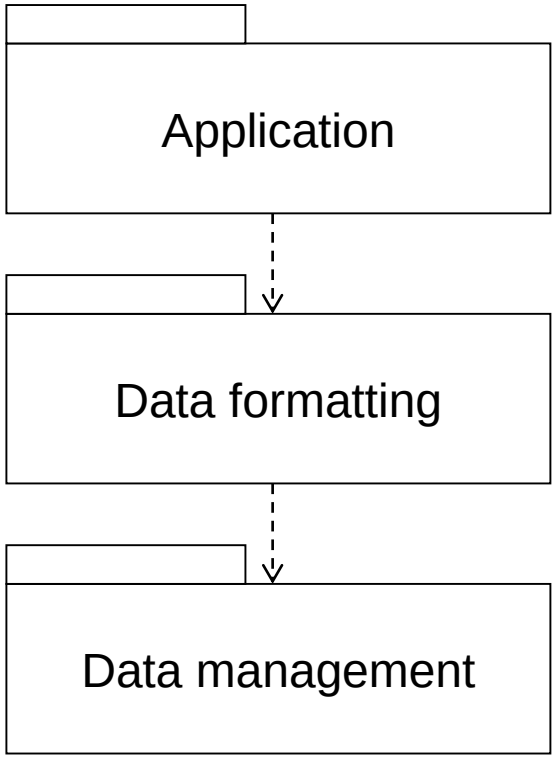
OSI 7 Layer Model

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.

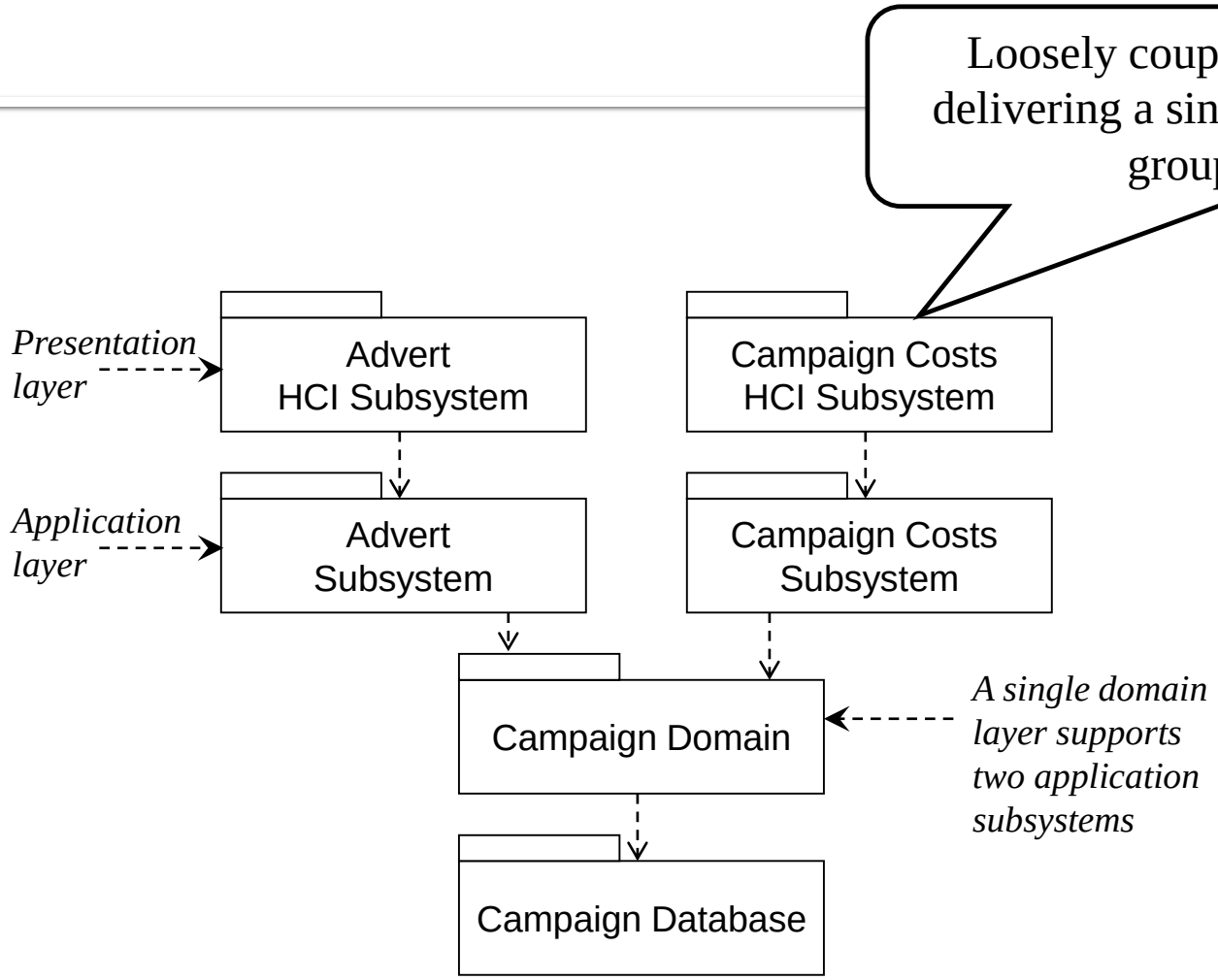
Applying a Layered Architecture

- Issues that need to be addressed include:
  - maintaining the stability of the interfaces of each layer
  - the construction of other systems using some of the lower layers
  - variations in the appropriate level of granularity for subsystems
  - the further sub-division of complex layers

Simple Layered Architecture.



Partitioned Subsystems



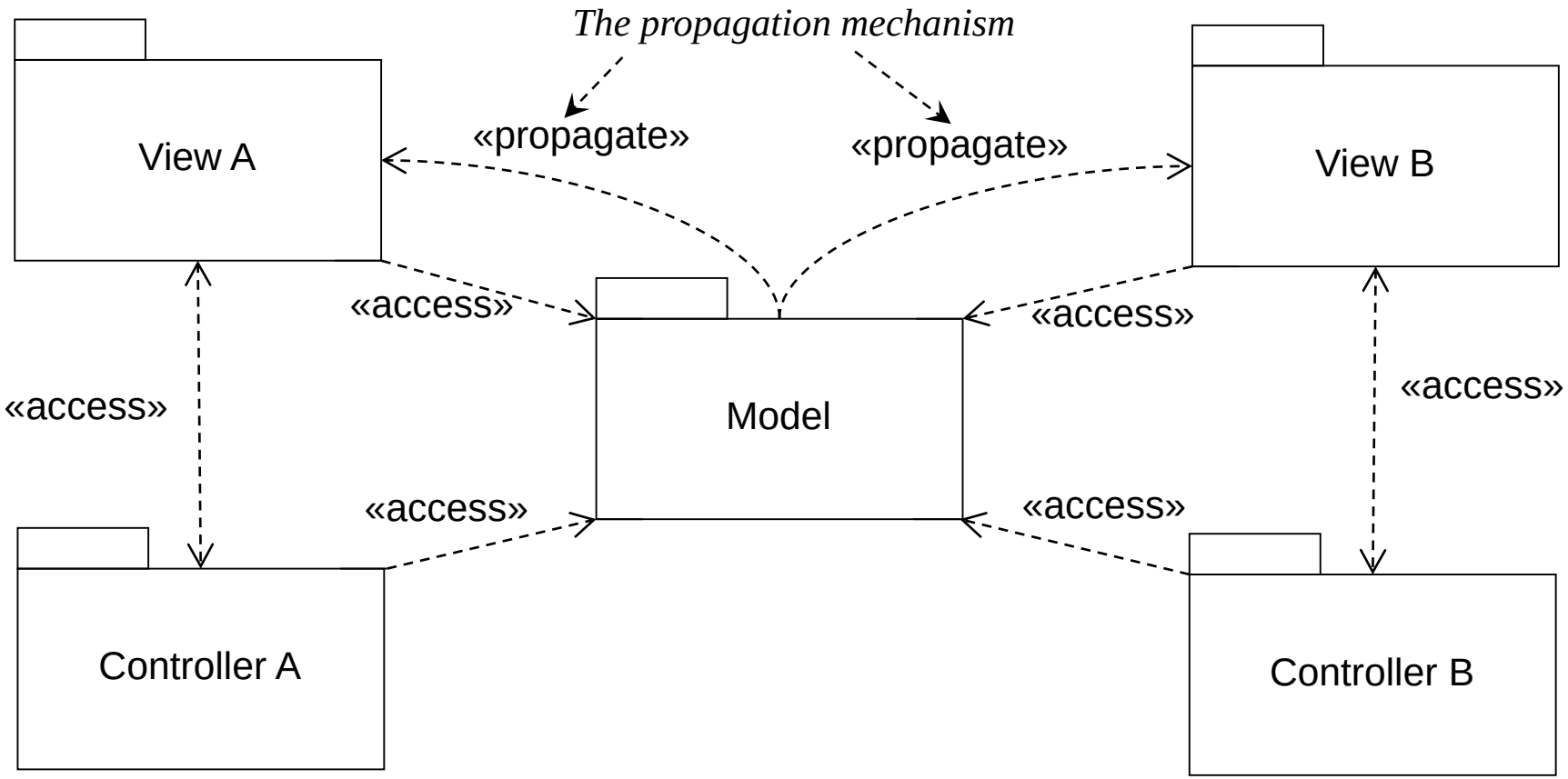
Four layer architecture applied to part of the Agate campaign management system

Difficulties

- Some of the difficulties that need to be resolved for designing applications
  - The same information should be capable of presentation in different formats in different windows
  - Changes made within one view should be reflected immediately in the other views
  - Changes in the user interface should be easy to make
  - Core functionality should be independent of the interface to enable multiple interface styles to co-exist

Model-View-Controller

Many interactive systems use MVC architecture



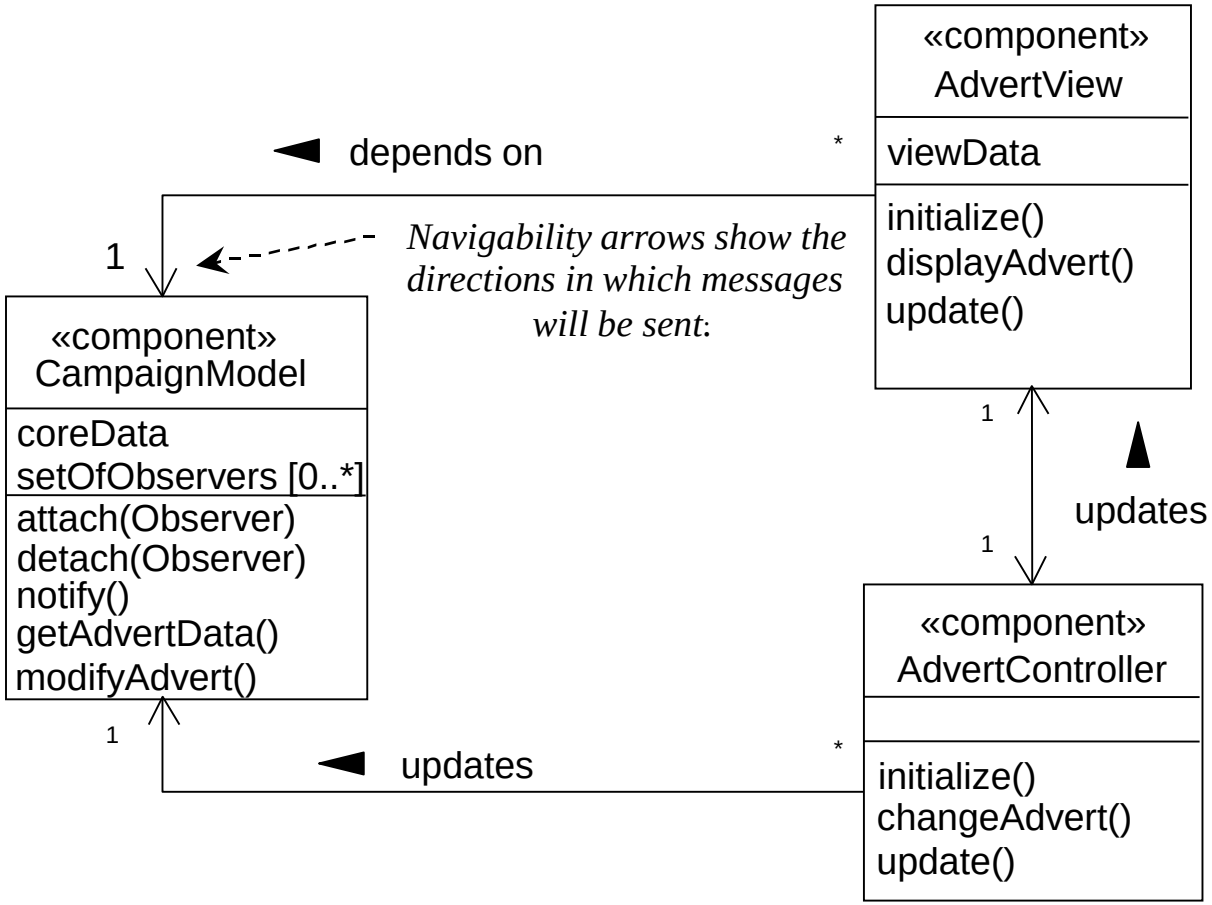
Model-View-Controller

- *Model*—provides the central functionality of the application and is aware of each of its dependent view and controller components.
- *View*—corresponds to a particular style and format of presentation of information to the user. The view retrieves data from the model and updates its presentations when data has been changed in one of the other views. The view creates its associated controller.

Model-View-Controller

- *Controller*—accepts user input in the form of events that trigger the execution of operations within the model. These may cause changes to the information and in turn trigger updates in all the views ensuring that they are all up to date.
- *Propagation Mechanism*—enables the model to inform each view that the model data has changed and as a result the view must update itself. It is also often called the dependency mechanism.

MVC applied to Agate



MVC Component Interaction