CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture P – W10L2

Introduction to Architecture

System Design: Architecture

- One of the major concerns in system design is architecture.

- According to Bass et al. “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” (**Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice*. 2nd Edition. Addison Wesley. 2003.**)

Select Solutions Factory Example

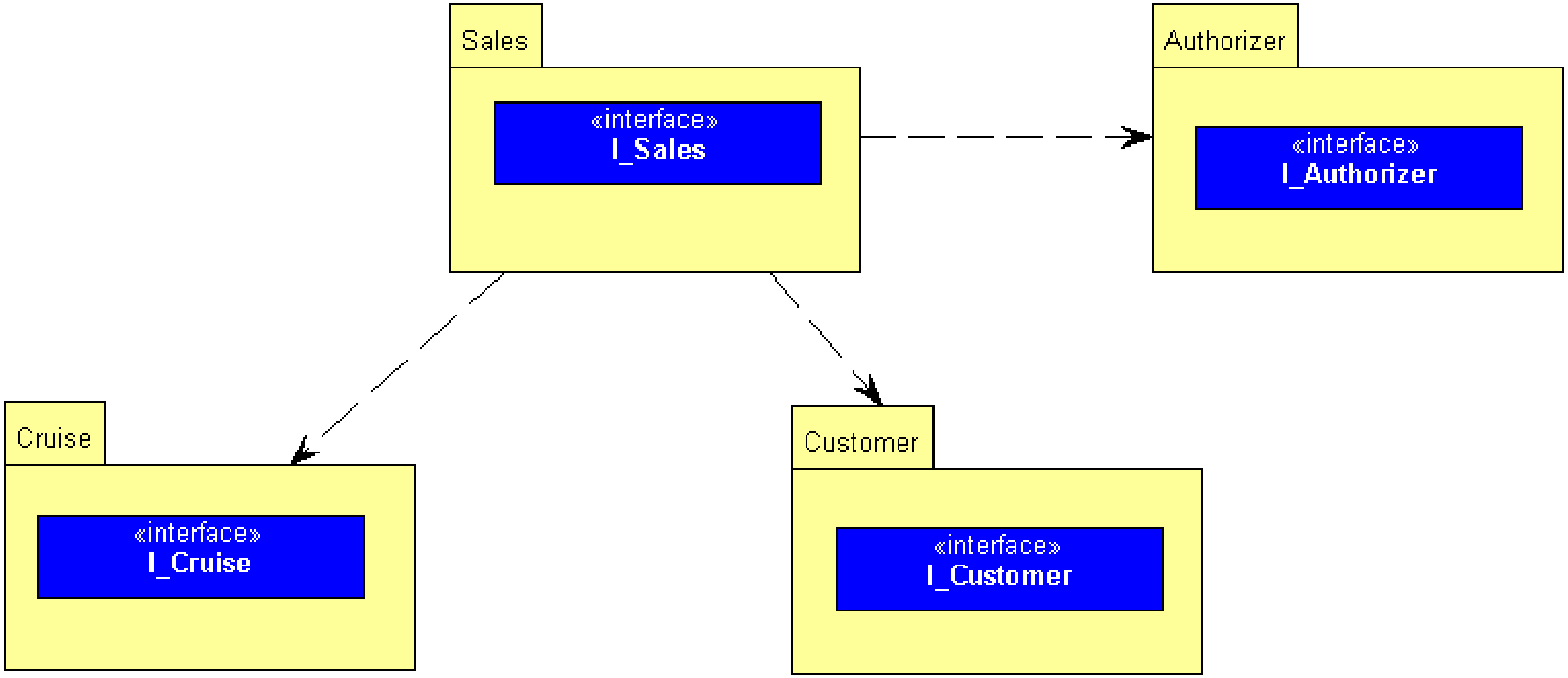


Figure 8. Business Architecture.

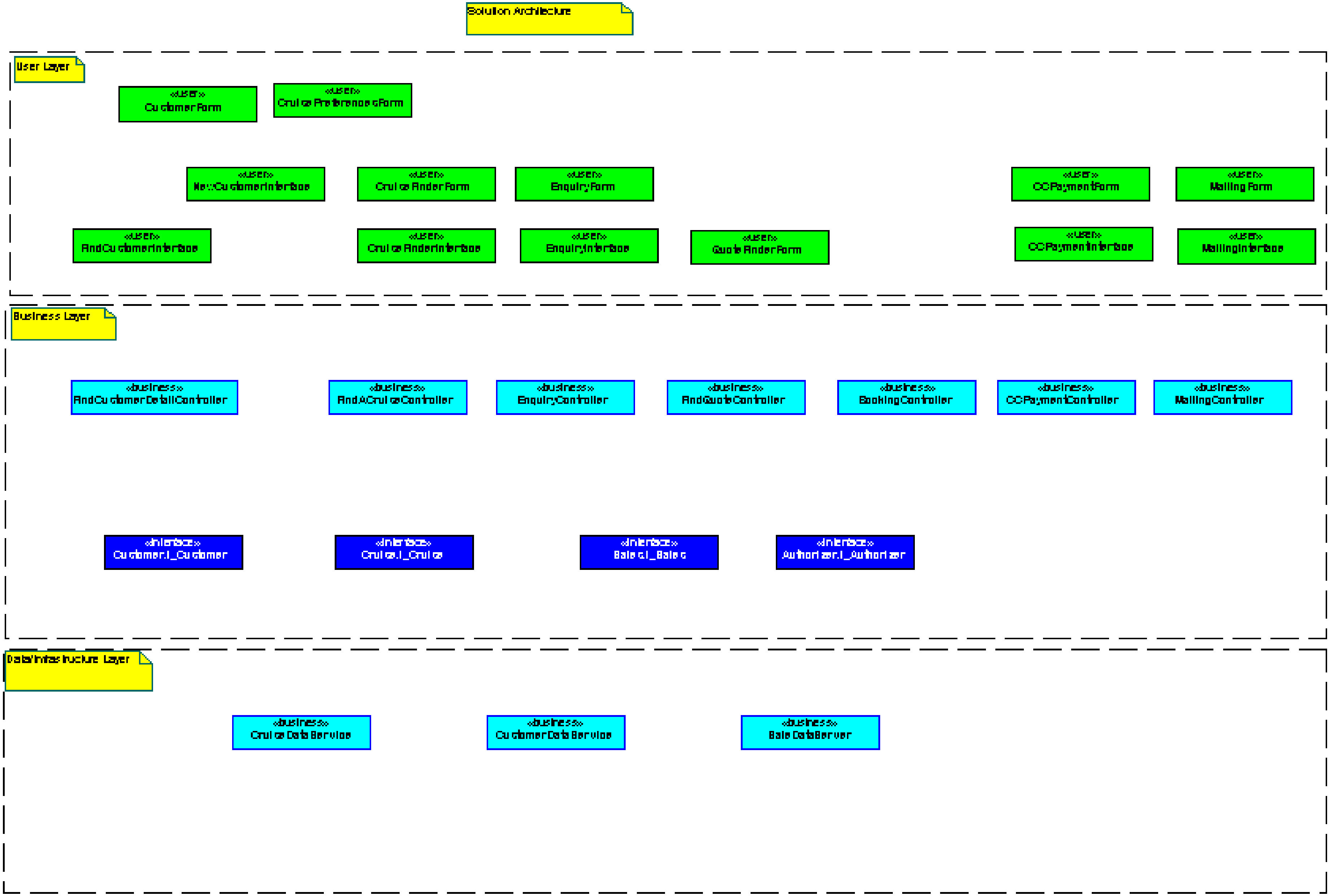

Solution Architecture
User Layer
«user» CustomerForm
«user» CruisePreferencesForm
«user» NewCustomerInterface
«user» CruiseFinderForm
«user» EnquiryForm
«user» CCPaymentForm
«user» MailingForm
«user» FindCustomerInterface
«user» CruiseFinderInterface
«user» EnquiryInterface
«user» QuoteFinderForm
«user» CCPaymentInterface
«user» MailingInterface
Business Layer
«business» FindCustomerDetailController
«business» FindACruiseController
«business» EnquiryController
«business» FindQuoteController
«business» BookingController
«business» CCPaymentController
«business» MailingController
«interface» Customer.I_Customer
«interface» Cruise.I_Cruise
«interface» Sales.I_Sales
«interface» Authorizer.I_Authorizer
Data/Infrastructure Layer
«business» Cruise Data Service
«business» Customer Data Service
«business» Sale DataServer

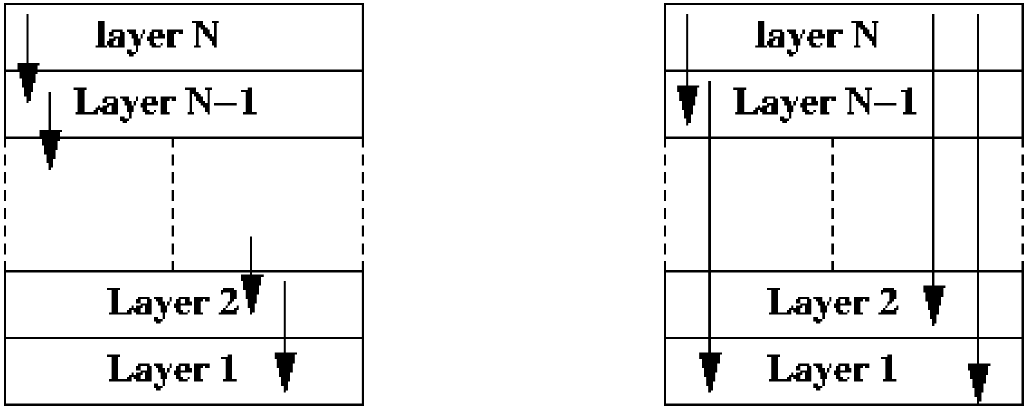
Figure 10. Technical solutions architecture.

System Design: Subsystems

- Advantages of division of system into sub-systems:
  - Produces smaller units of development.
  - Maximises reuse at the component level.
  - Helps developers to cope with complexity.
  - Improves maintainability.
  - Improves portability.
- Each sub-system should have a clearly specified boundary and fully defined interfaces.
- Each sub-system provides services for other sub-systems, and two styles of communication make this possible: client server and peer to peer.
- Client server sub-systems better - less tightly coupled.
- Could also use component and deployment diagrams.

System Design: Layering & Partitioning

- Two approaches to division of software systems into sub-systems.
  - Layering: so called because different sub-systems represent different levels of abstraction.
  - Partitioning: each sub-system focuses on a different aspect of the functionality of the system as a whole.
- Pros and cons ???



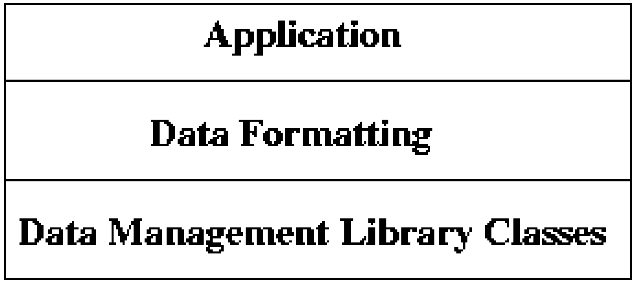
Closed architecture
Messages may be only sent
to the adjacent lower layer

Open architecture
Messages may be sent
to any lower layer

Schematic of a layered architecture

System Design: Layering and Partitioning

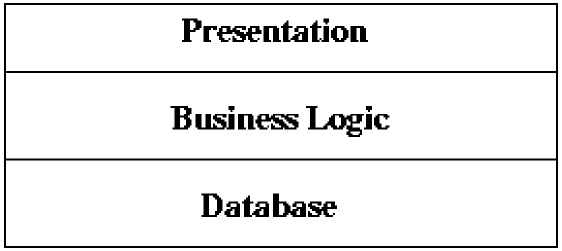
- The OSI 7 layer model is an example of one of the best-known examples of a layered architecture.
- Buschmann et al. (1996) suggest that the following issues need to be addressed when applying a layered architecture:
  - Maintaining the stability of the interfaces of each layer.
  - The construction of other systems using the lower layers.
  - Variations in the appropriate level of granularity for sub-systems.
  - The further sub-division of complex layers.
  - Performance reductions due to a closed layered architecture.



Simple Layered Architecture

System Design: Layered Architecture

- Three layered architecture commonly used for business-oriented information systems.
- Eliminates tight coupling between user interfaces and data representation.
- A common four layered architecture separates the business logic into application logic and domain layers.
- Used when several applications may share one domain layer, or because complexity of business objects forces a separation into two layers.

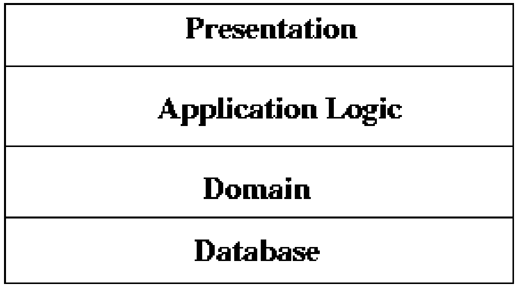


Presentation

Business Logic

Database

Three Layered Architecture



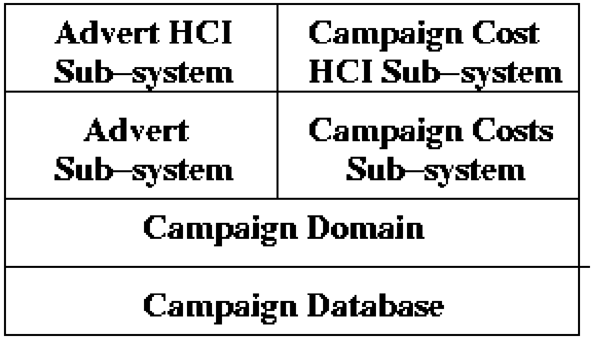
Presentation

Application Logic

Domain

Database

Four Layered Architecture



Advert HCI Sub–system

Campaign Cost HCI Sub–system

Advert Sub–system

Campaign Costs Sub–system

Campaign Domain

Campaign Database

Four layered architecture applied to part of the Agate management system

System Design: Layered Architecture

Buschmann et al. (1996):

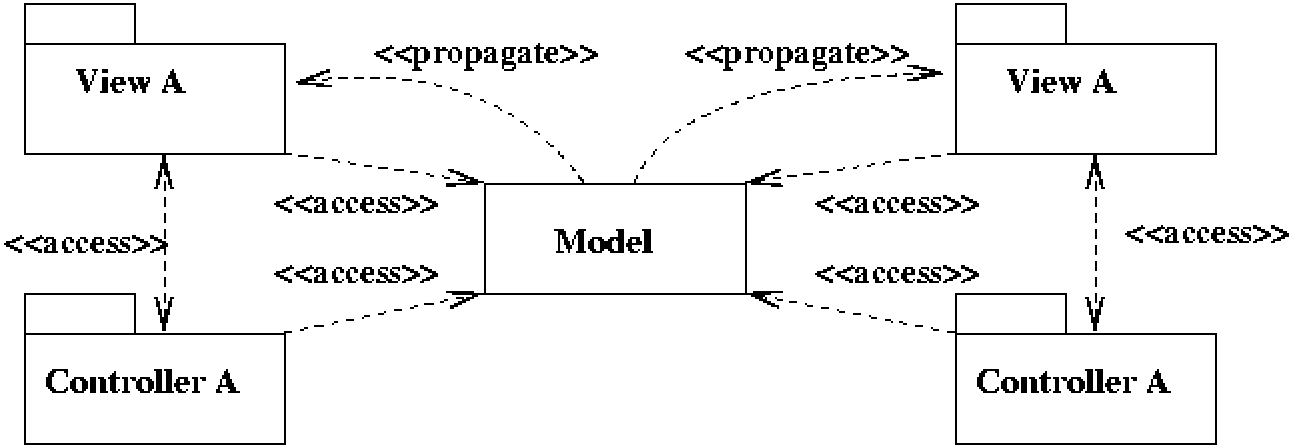
1. Define criteria by which application will be grouped into layers i.e. level of abstraction from hardware.
   - Low levels primitives in lower layers, application concepts in higher layers.
2. Determine the number of layers.
3. Name the layers and assign functionality to them.
   - Top layers should focus on provision of services required by users.
   - Layer below that should focus on provision of services and infrastructure that support services in the top layer.
4. Specify the services for each layer.
   - Lower level layers should provide a limited set of primitive services that are used by a larger number of services in the higher layers.
5. Refine the layering by iterating through steps 1 to 4.
6. Specify interfaces for each layer.
7. Specify structure of each layer. May involve partitioning.
8. Specify communication between layers.
9. Reduce coupling between adjacent layers in a closed architecture – layers should be strongly encapsulated. Layers should only have knowledge of layer immediately below it.

System Design: MVC

- Many interactive systems use the Model-View-Controller (MVC) architectural pattern.
- Separates an application into three major types of components:
  - Models that comprise the main functionality.
  - Views that present the user interface.
  - Controllers that manage the update to views.
- Facilitates maintenance and portability.
- Common for the view to differ for each user.
- Implies that data and functionality available to each user should be tailored to needs. i.e. Agate case study, different perspectives of campaign manager and creative artist.
- An alternative architectural pattern for interactive systems is the Presentation-Abstraction-Control (PAC) pattern.

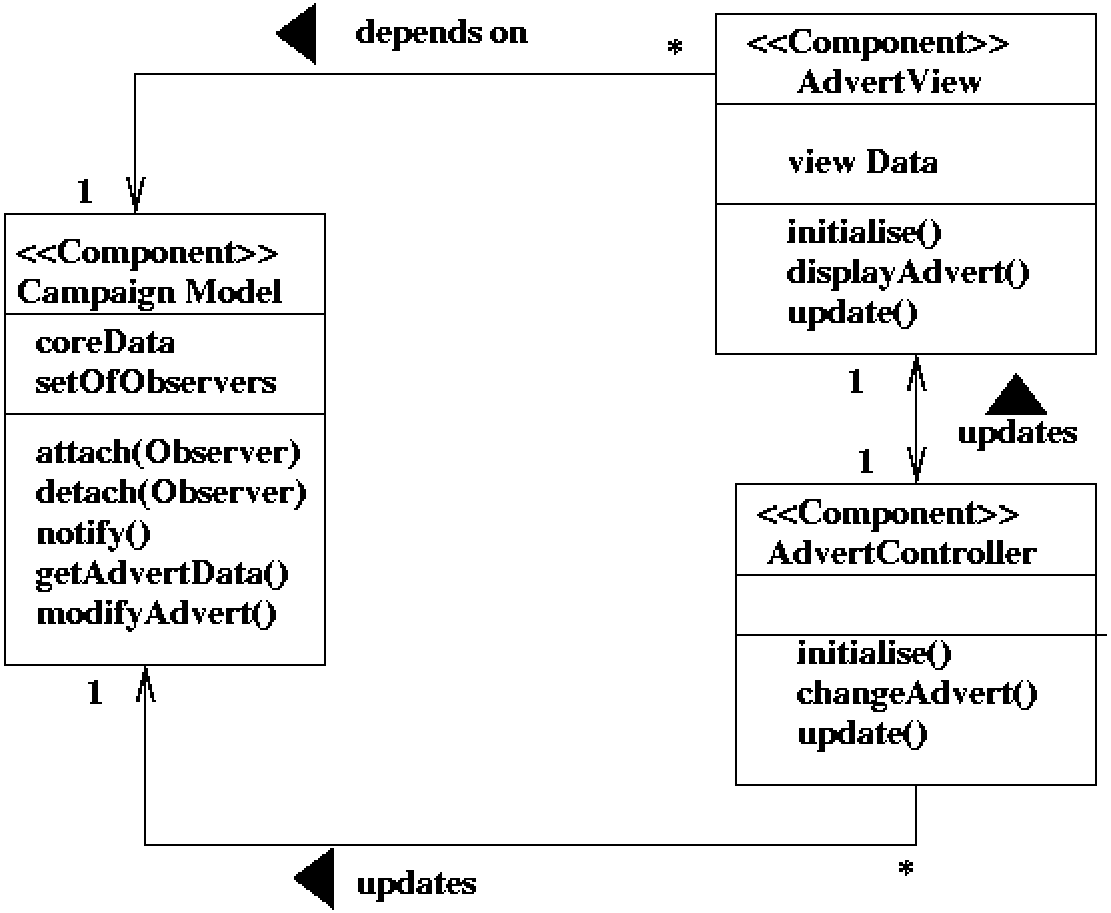
System Design: MVC

- Issues that should be addressed:
  - The same information should be capable of presentation in different formats in different windows.
  - Changes made within one view should be reflected immediately in all other views.
  - Changes in user interfaces should be easy to make.
  - Core functionality should be independent of the interface to enable multiple interface styles to co-exist.
- The MVC architecture solves the problems of updating by the separation of core functionality (model) from the interface through the use of a mechanism for propagating updates to other views.
- The interface is split into two elements: the output presentation (view) and the input controller.



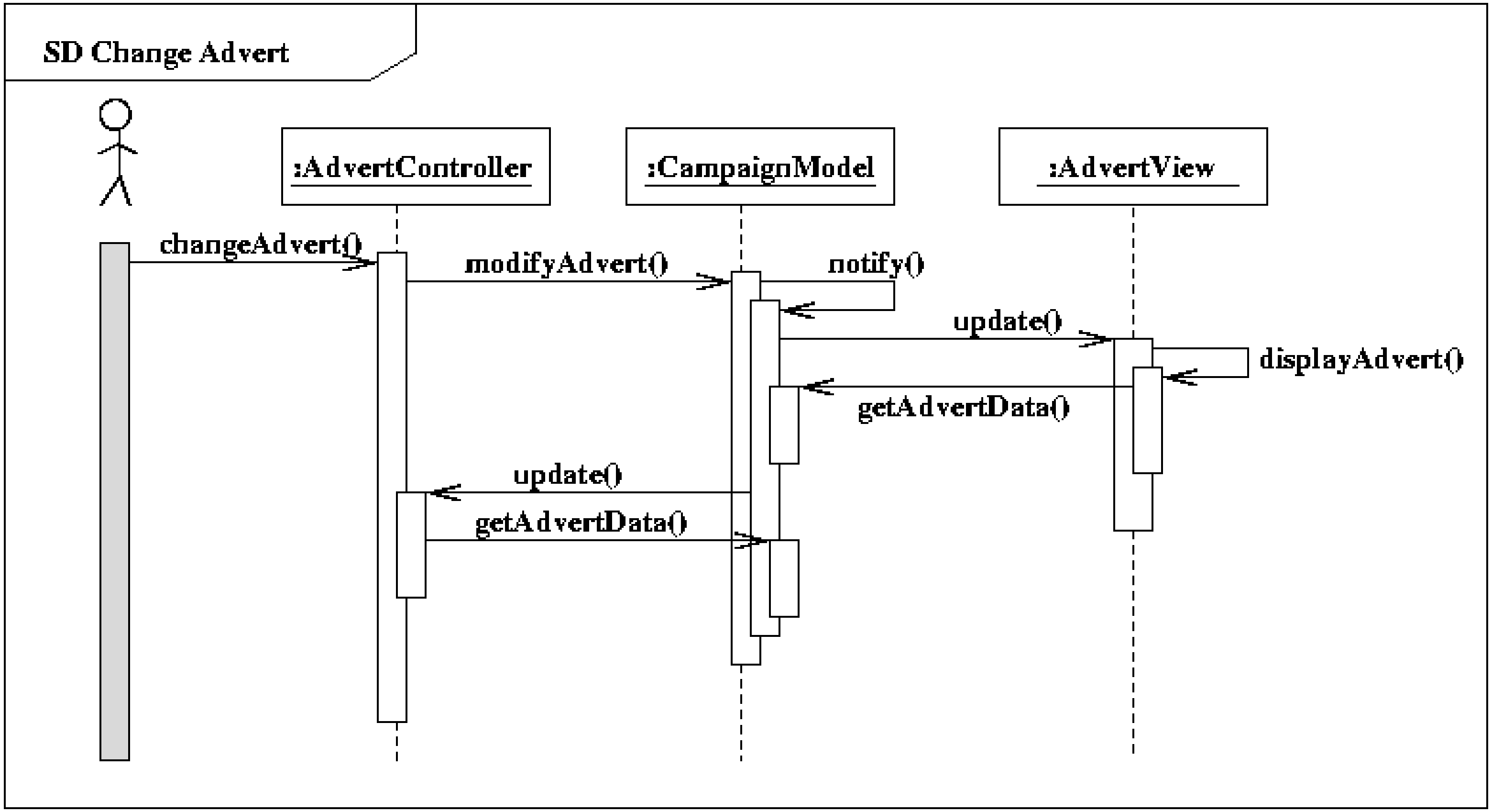
Model–View–Controller (adapted from Hopkins and Horan, 1995

System Design: MVC Class Diagram



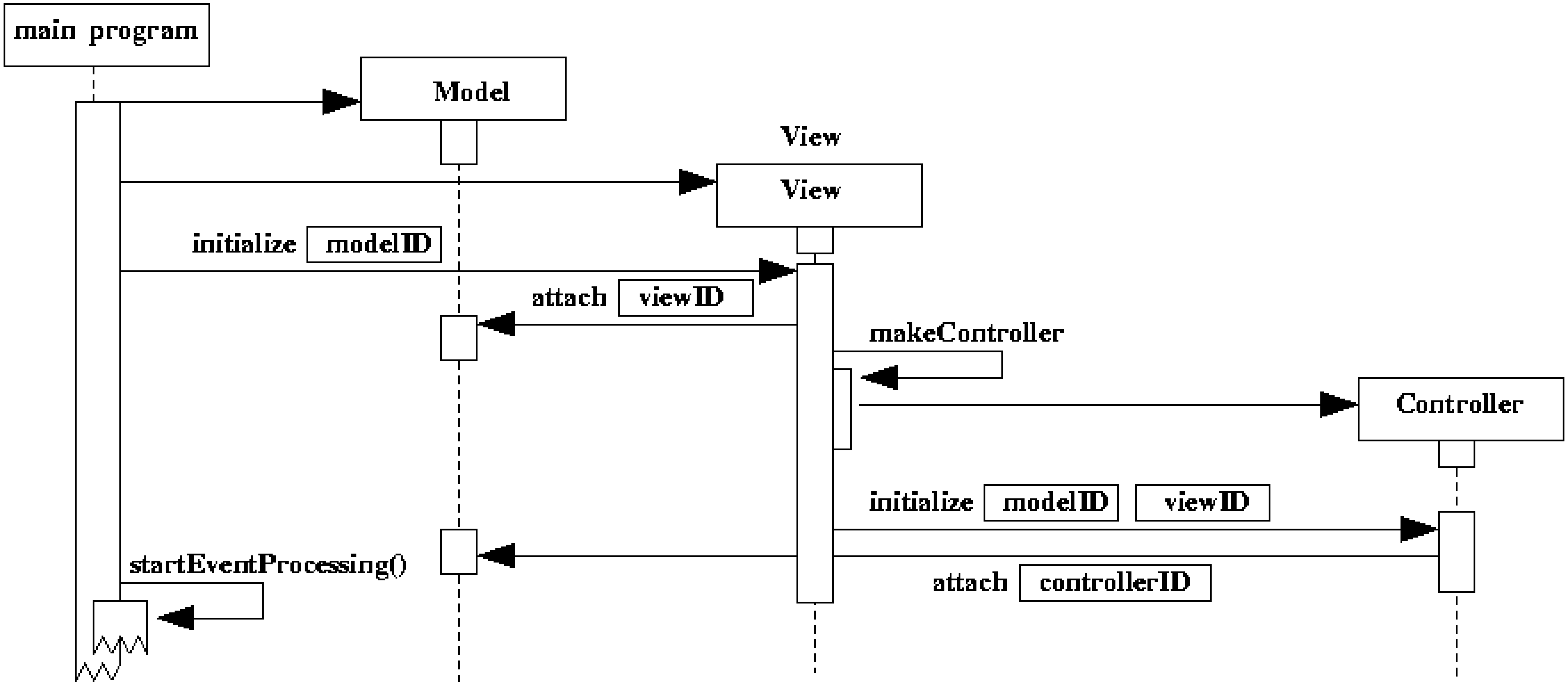
Responsibilities of MVC components, as applied to Agate

System Design: MVC - Behaviour

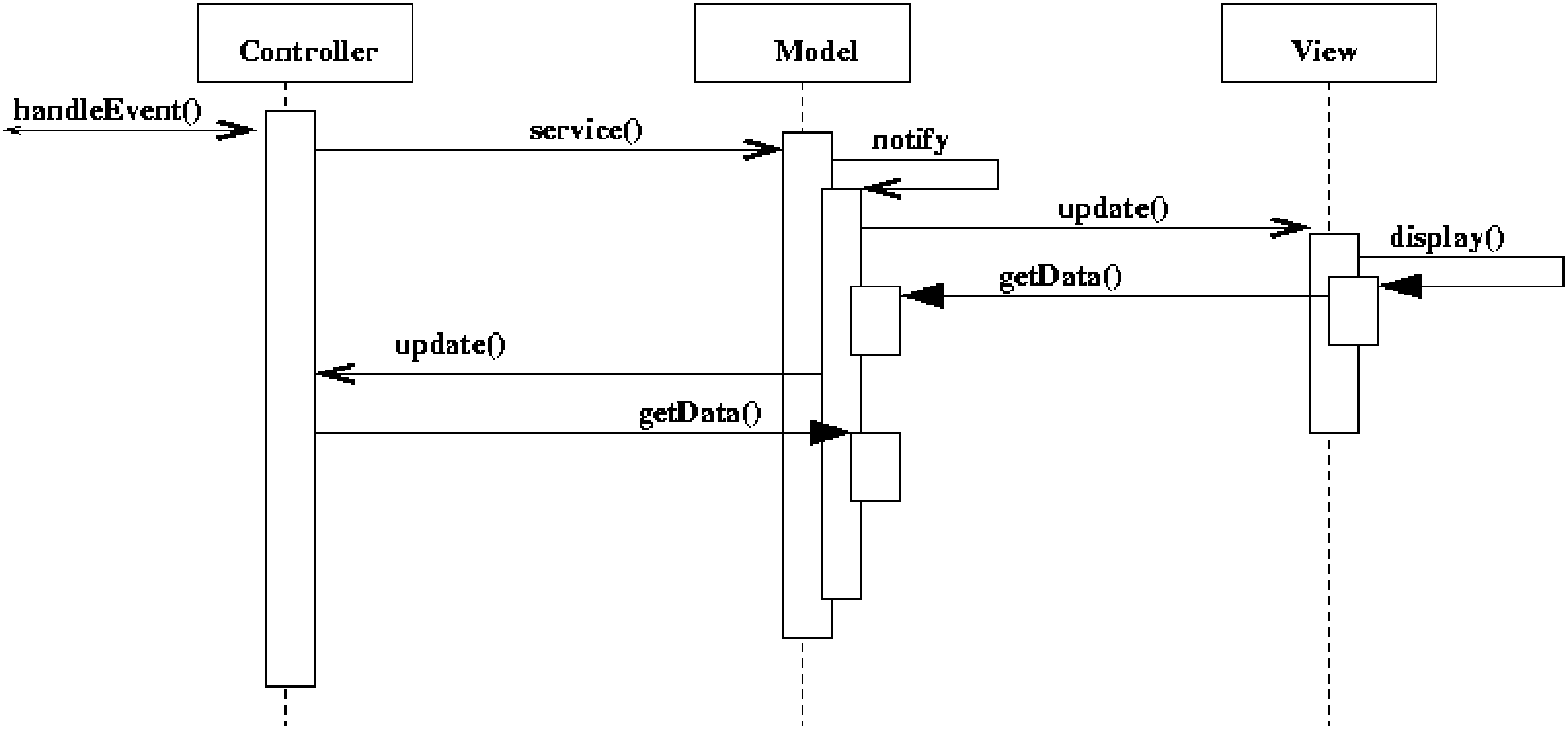


MVC component interaction (adapted from Buschmann et al., 1996)

System Design: MVC - Initialisation

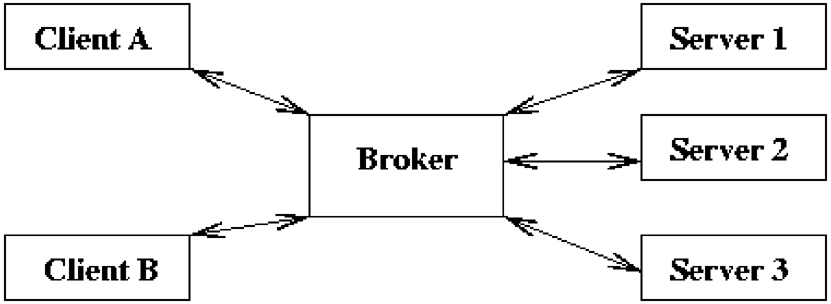


System Design: MVC - Behaviour

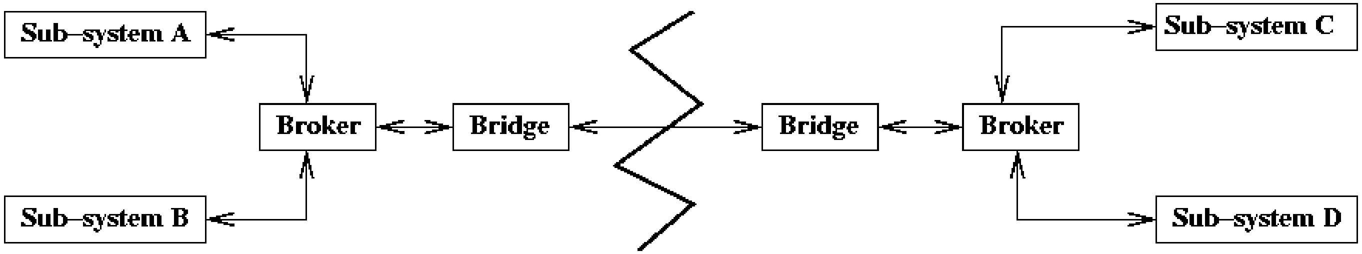


System Design: Distributed Systems

- Distributed architectures usually supported by Distributed Database Management Systems (DBMS) and by middleware such as CORBA compliant object request brokers (ORBs).
- Figure 12 depicts a simplified version of the broker architecture for a distributed system (Buschmann et al., 1996).
- Broker decouples the client and server.

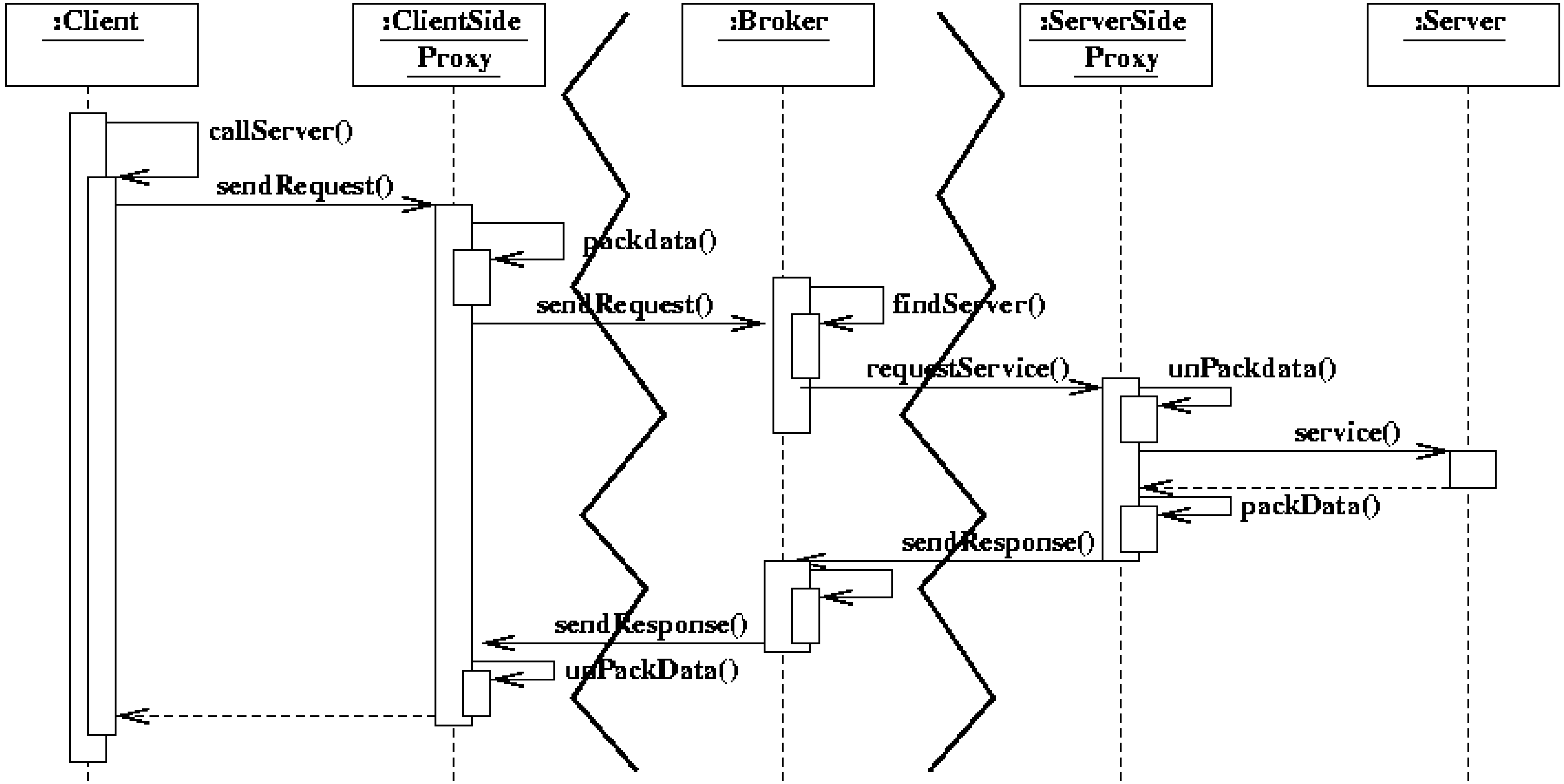


Simplified broker architecture



Broker architecture using Bridge components

System Design:

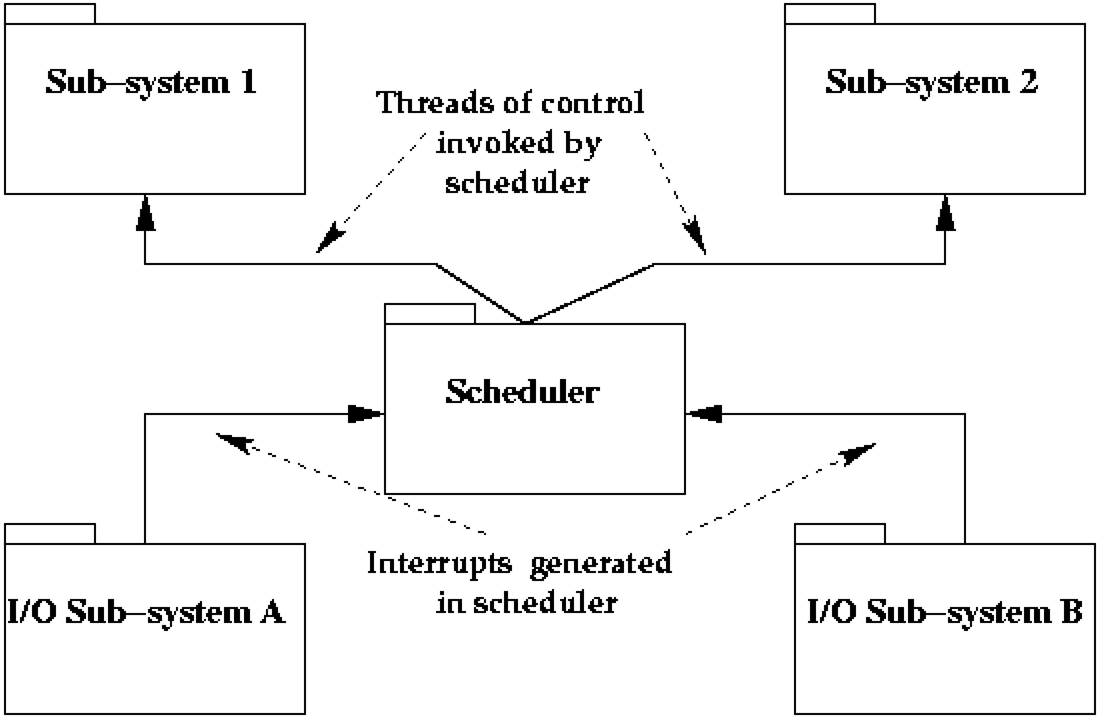


Broker architecture for local server (adapted from Buschmann et al., 1996)

System Design: Concurrency

- Different objects that are not actively executing at the same time may be implemented on same logical processor.
- Objects that operate concurrently must execute on different logical processors.
- Logical processors ultimately mapped to physical processors on deployment diagram.
- Examine use case descriptions.
- Examine state charts.
- Two types of concurrency:
  - Many to one: - multitasking.
  - Many to many: multi-processor environment - distributed system.
- On sequence diagrams, specify focus of control, and active objects.
- On a uni-processor hardware architecture, may use a scheduler sub-system to ensure that each thread of control operates within the constraints on its response time.

System Design: Concurrency



Scheduler handling concurrency

System Design: Processor Allocation

- The application should be divided into sub-systems.
- Processing requirements for each sub-system should be estimated.
- Access factors and location requirements should be specified.
- Concurrency requirements for the sub-systems should be identified.
- Each sub-system should be allocated to an appropriate hardware platform - PC, workstation, server, embedded micro-controller.
- Communications requirements should be identified.
- Communications infrastructure should be specified.

3. Reading

□ Chapters 13 in Bennett et al. (Fourth Edition).

Have a look at:

□ Sections 3 and 4 in Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. Pattern-Oriented Software Architecture: A System of Patterns. Wiley. 1996