MACIASZEK, L.A. (2007):
***Requirements Analysis and System Design*, 3rd ed.**
Addison Wesley, Harlow England
ISBN 978-0-321-44036-5

Chapter 3
Requirements Specification

Topics

- Architectural prerogatives
- State specifications
- Behavior specifications
- State change specifications

1. Architectural prerogatives

- software architecture:

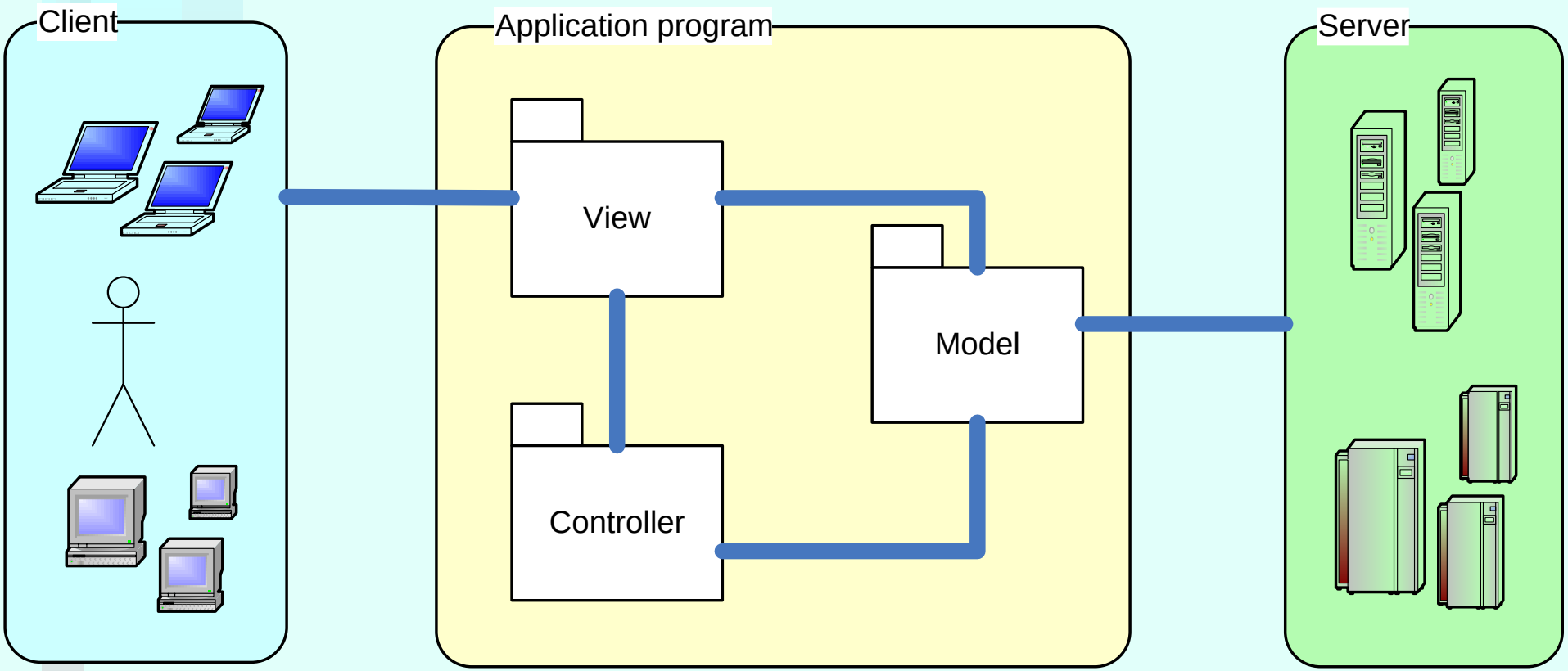
• addresses nonfunctional requirements (software qualities)

• “It’s the key to achieving intellectual control over a sophisticated system’s enormous complexity”

Architectural design

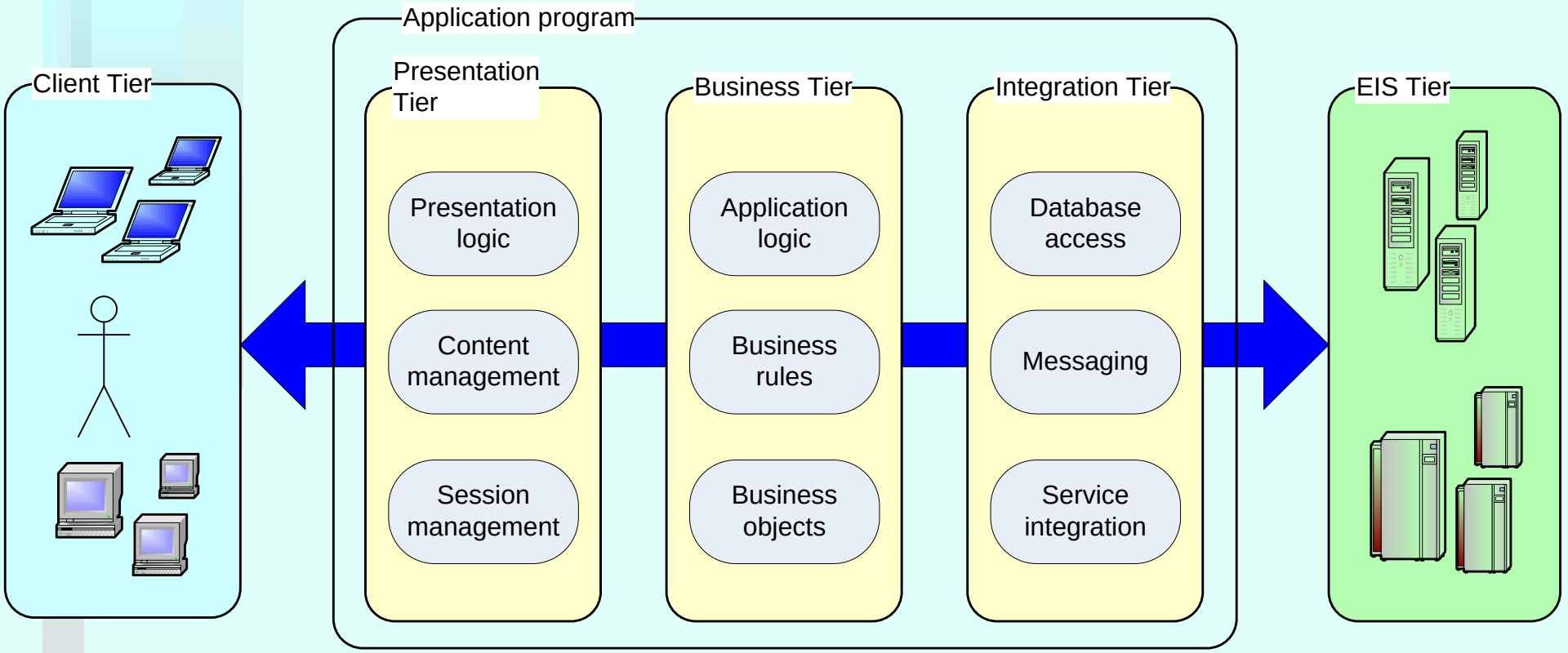
- Design
  - detailed
  - architectural
- Object dependencies $\rightarrow$ complexity and adaptiveness (supportability)
- Architectural model
  - hierarchical layers
  - restrictions on object inter-communications to minimize dependencies

Model-View-Controller (MVC)



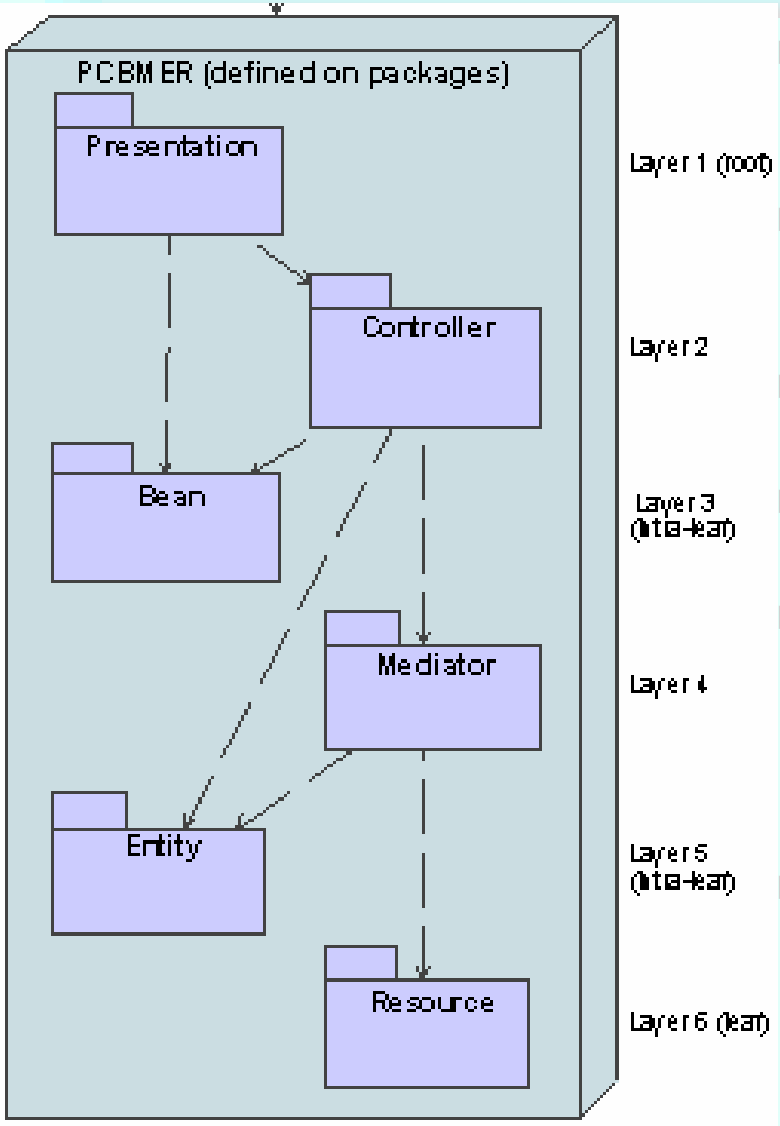
- ***Model objects*** represent data objects
- ***View objects*** represent user interface (UI) objects
- ***Controller objects*** represent mouse and keyboard events

The Core J2EE architecture



- The user communicates with the system from the *Client* tier
- The *EIS* tier (called also the *Resource* tier) is any persistent information delivery system
- The user accesses the application via the *Presentation* tier (known also as the *Web* tier)
- The *Business* tier contains application logic
- The *Integration* tier establishes and maintains connections to data sources

The Core PCBMER framework



- The *Presentation* represents the screen and UI objects on which the beans can be rendered
- The *Bean* represents the data classes and value objects that are destined for rendering on user interface
- The *Controller* represents the application logic
- The *Entity* contains classes representing business objects
- The *Mediator* manages business transactions, enforces business rules, instantiates business objects in the Entity layer, and in general manages the memory cache of the application
- The *Resource* is responsible for all communications with external persistent data sources

Architectural principles

- DDP – downward dependency principle
- UNP – upward notification principle
- NCP – neighbor communication principle
- APP – acquaintance package principle
- EAP – explicit association principle
- CEP – cycle elimination principle
- CNP – class naming principle

DDP, UNP, NCP

- DDP
  - higher PCBMER layers depend on lower layers
  - lower layers should be designed to be more stable
- UNP
  - upward communication that minimizes object dependencies
  - lower layers rely on interfaces and event processing (publisher/subscriber protocols) to communicate with objects in higher layers
- NCP
  - objects can communicate across layers only by using direct neighbors
  - chains of message passing

APP, EAP

- APP
  - separate layer of interfaces to support more complex object communication under strict supportability guidelines
  - subsystem of interfaces only
    - other objects in the system can use these interfaces, and pass them in arguments to method calls, instead of concrete objects → classes in non-neighboring subsystems can communicate without knowing the concrete suppliers of services (and, therefore, without creating dependencies on concrete classes).
- EAP
  - legitimizes run-time object communication in compile-time data structures.

CEP, CNP

- CEP
  - cyclic dependencies, between classes and other structures (methods, packages, subsystems)
  - unavoidable, but can be neutralized
    - extra classes to reduce a network of calls to a hierarchy
    - purposeful use of interfaces
- CNP
  - name of each class and each interface in the system should identify the subsystem/package layer to which it belongs
  - ensuring that each class begins with a single letter identifying the PCBMER layer (i.e. P, C, etc.)
    - EVideo means that the class is in the entity subsystem
    - IMVideo means that the interface is in the mediator subsystem.

Review Quiz 3.1

1. What is the necessary and the most important condition to building into the software the quality of adaptiveness (supportability)?
2. Which MVC objects represent mouse and keyboard events?
3. Which Core J2EE tier is responsible for establishing and maintaining connections to data sources?
4. Which PCBMER layer is responsible for establishing and maintaining connections to data sources?