# Software Design
# The Dynamic Model
# Design Sequence Diagrams and Communication Diagrams

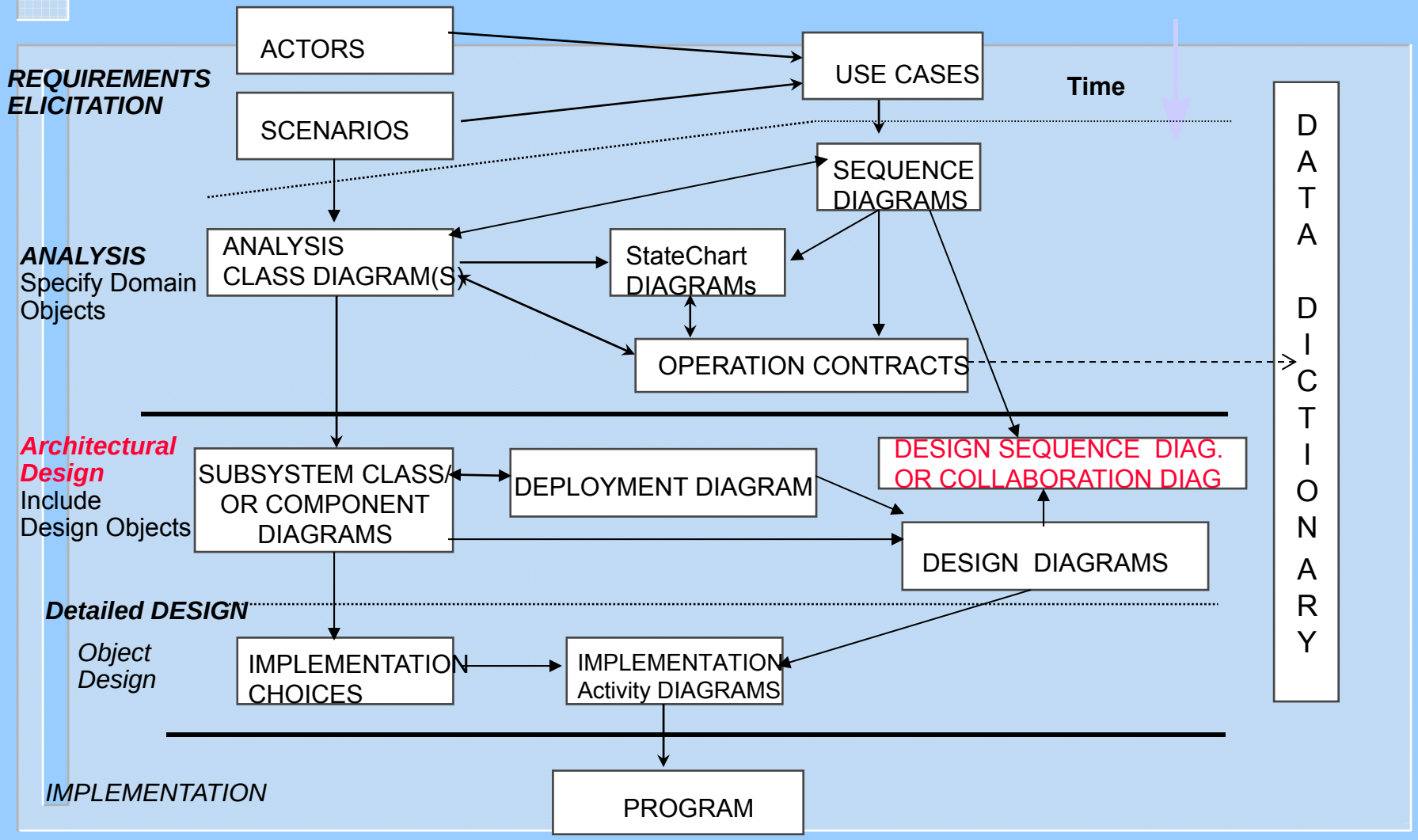
Instructor: Dr. Hany H. Ammar

Dept. of Computer Science and Electrical Engineering, WVU

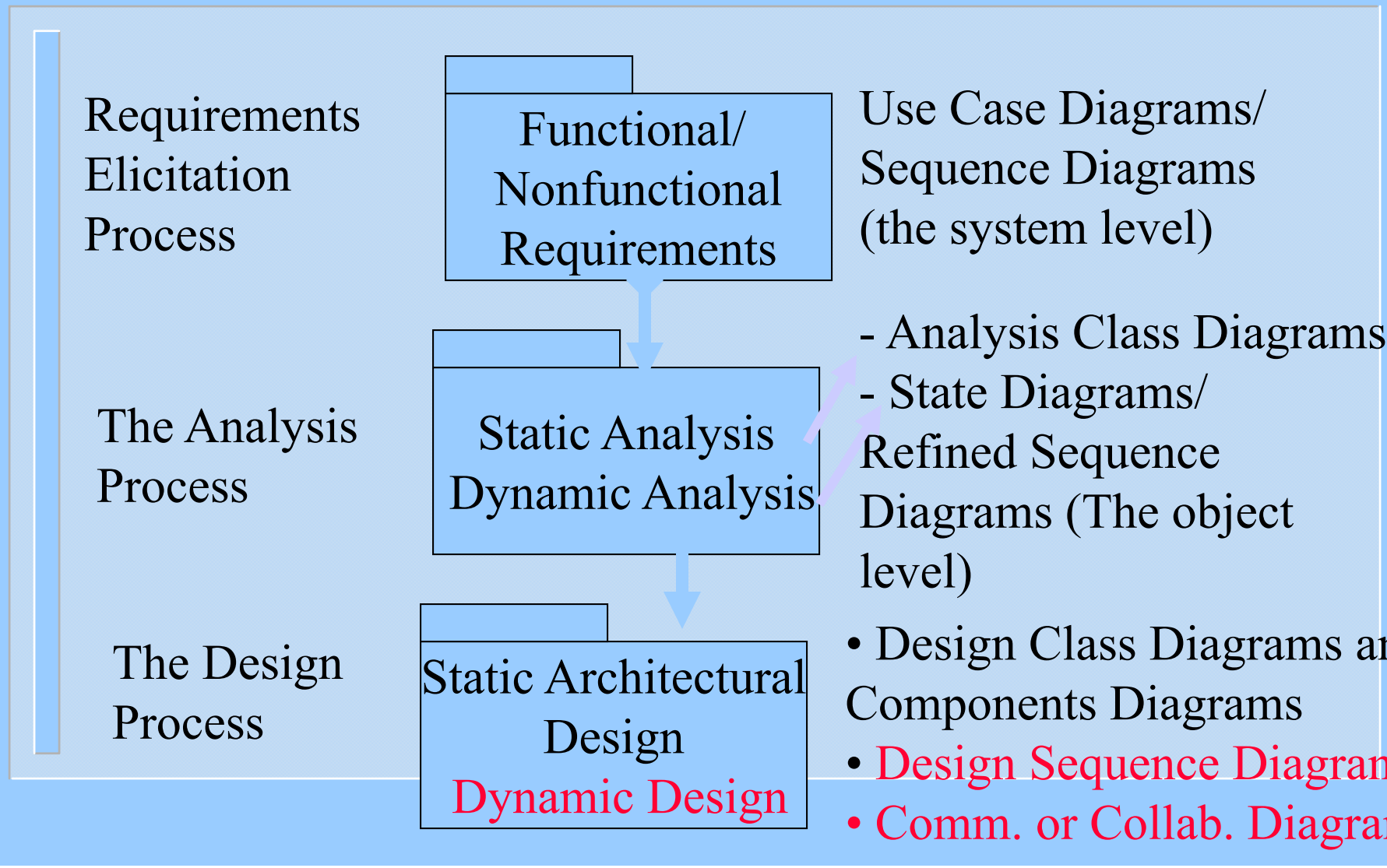
# Ouline

- UML Development – Overview
- The Requirements, Analysis, and Design Models
- Static and Dynamic Design
- Examples

# UML Development - Overview



# The Requirements, Analysis, and Design Models



# Static and Dynamic Design

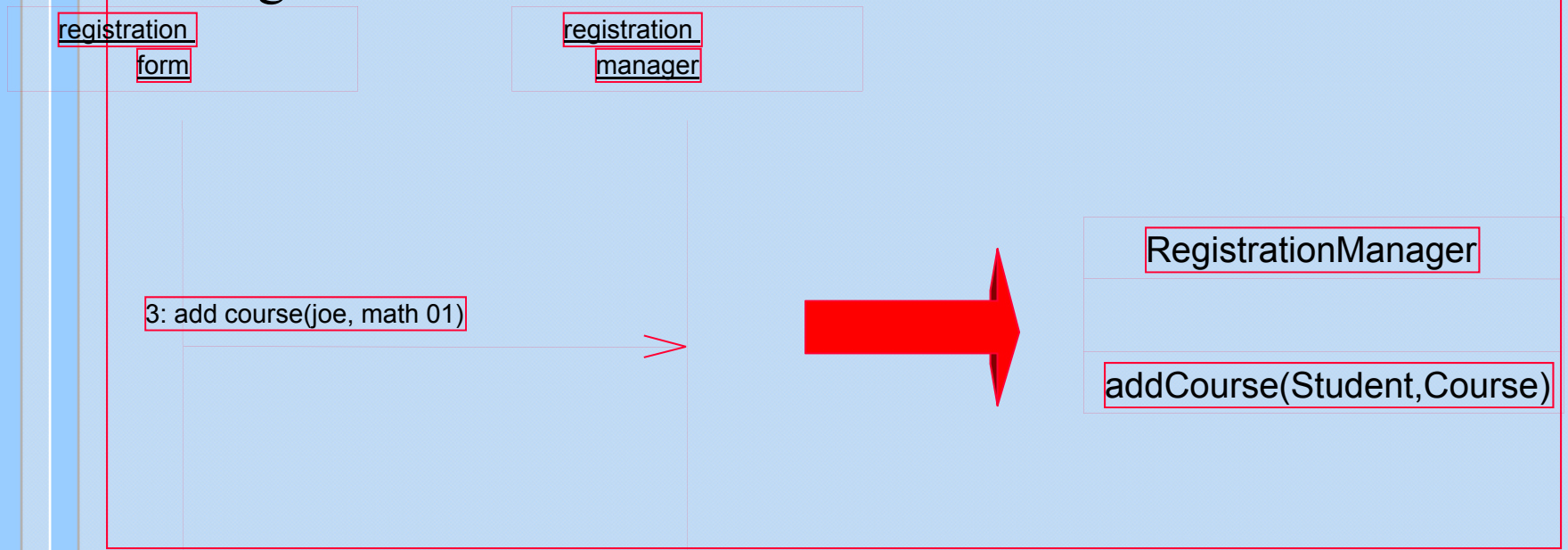
- The Static design class diagram model is developed iteratively using the dynamic model represented in design sequence diagrams or collaboration (also called communication) diagrams
- Design sequence diagrams show detailed interactions between objects of classes in different subsystems
- They are defined based on analysis (system) sequence diagrams developed for a given Use-Case scenario defined in the analysis (or the requirements) model

# Static and Dynamic Design

- The development of Design Class Diagrams is completed by defining operations and classes to support the interactions represented in the dynamic model.
- Operations of classes in the design class diagram are defined using the dynamic interactions in the dynamic model sequence diagrams
- New classes might be needed in the design class diagram to support the interactions of objects in the sequence diagrams (e.g., Interface classes, proxy classes, scheduler classes etc.)

# Class Operations

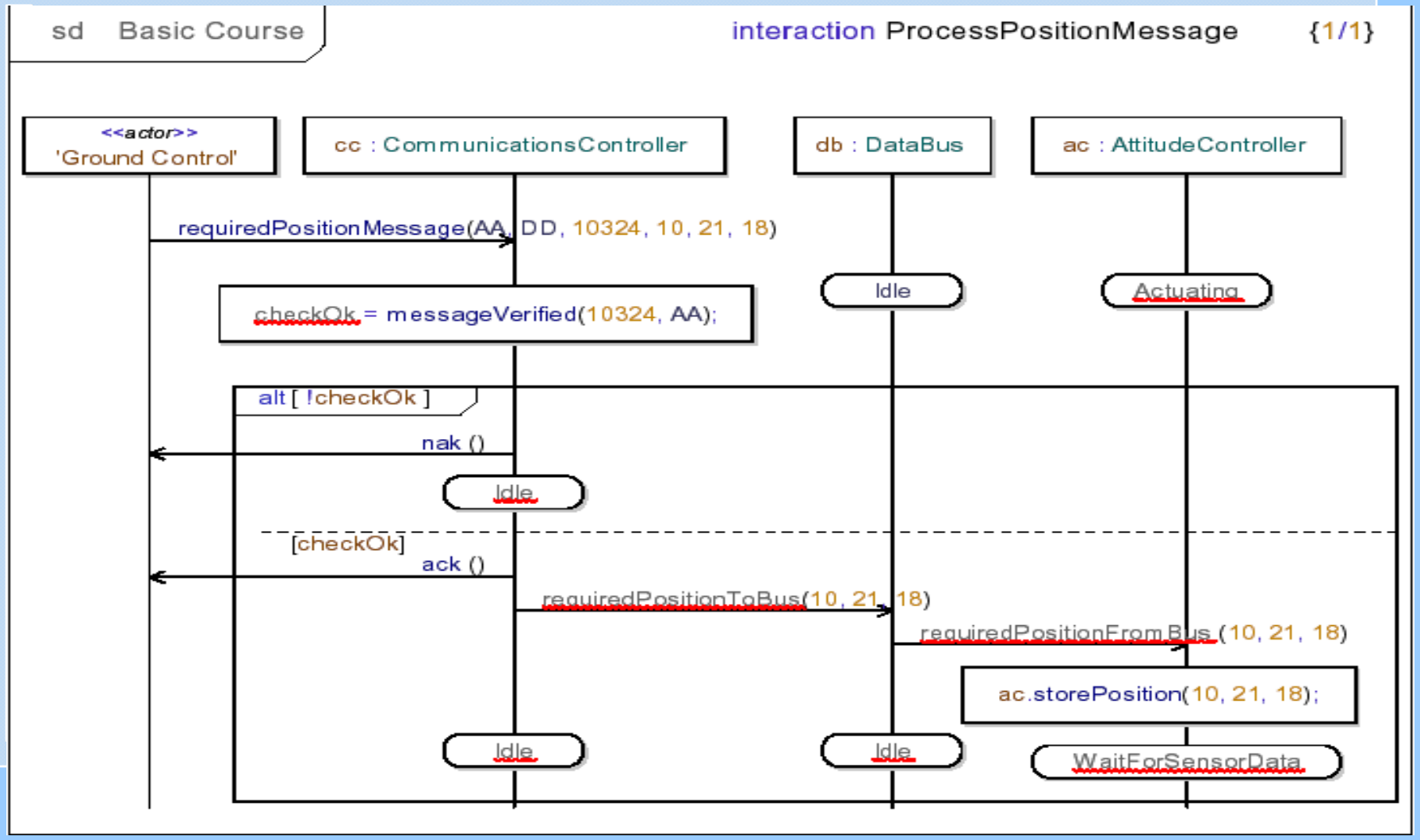
- The behavior of a class is represented by its operations
- Operations may be found by examining interaction diagrams



# Design Sequence Diagrams (UML2)

Specify operations and states on the timelines

Detailed Parameters list can be specified during detailed design



# Digital Sound Recorder: A Complete Example

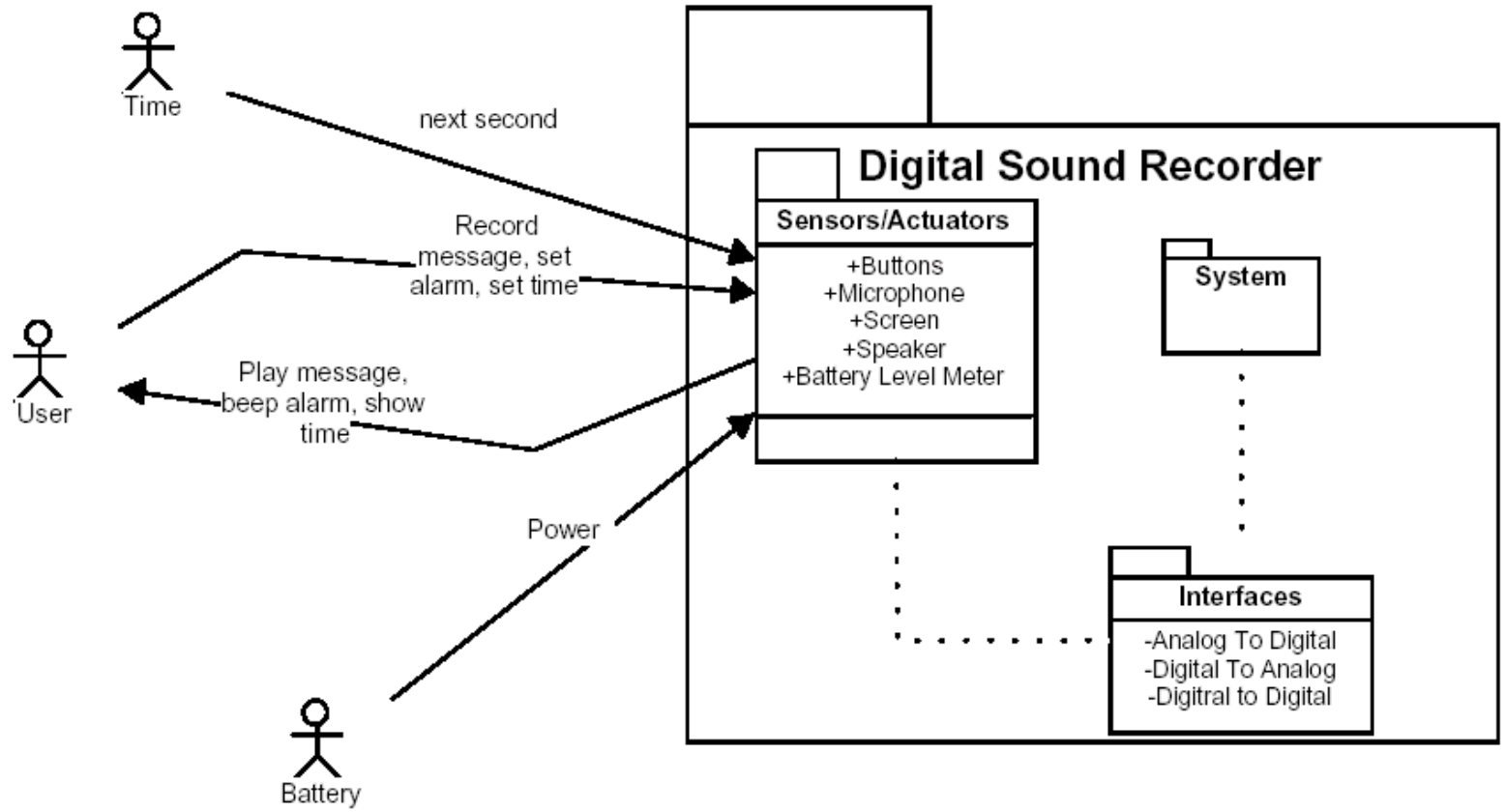



**Figure 2.2: Context-Level diagram**

# Digital Sound Recorder:
# A Complete Example, Use Case Diag.

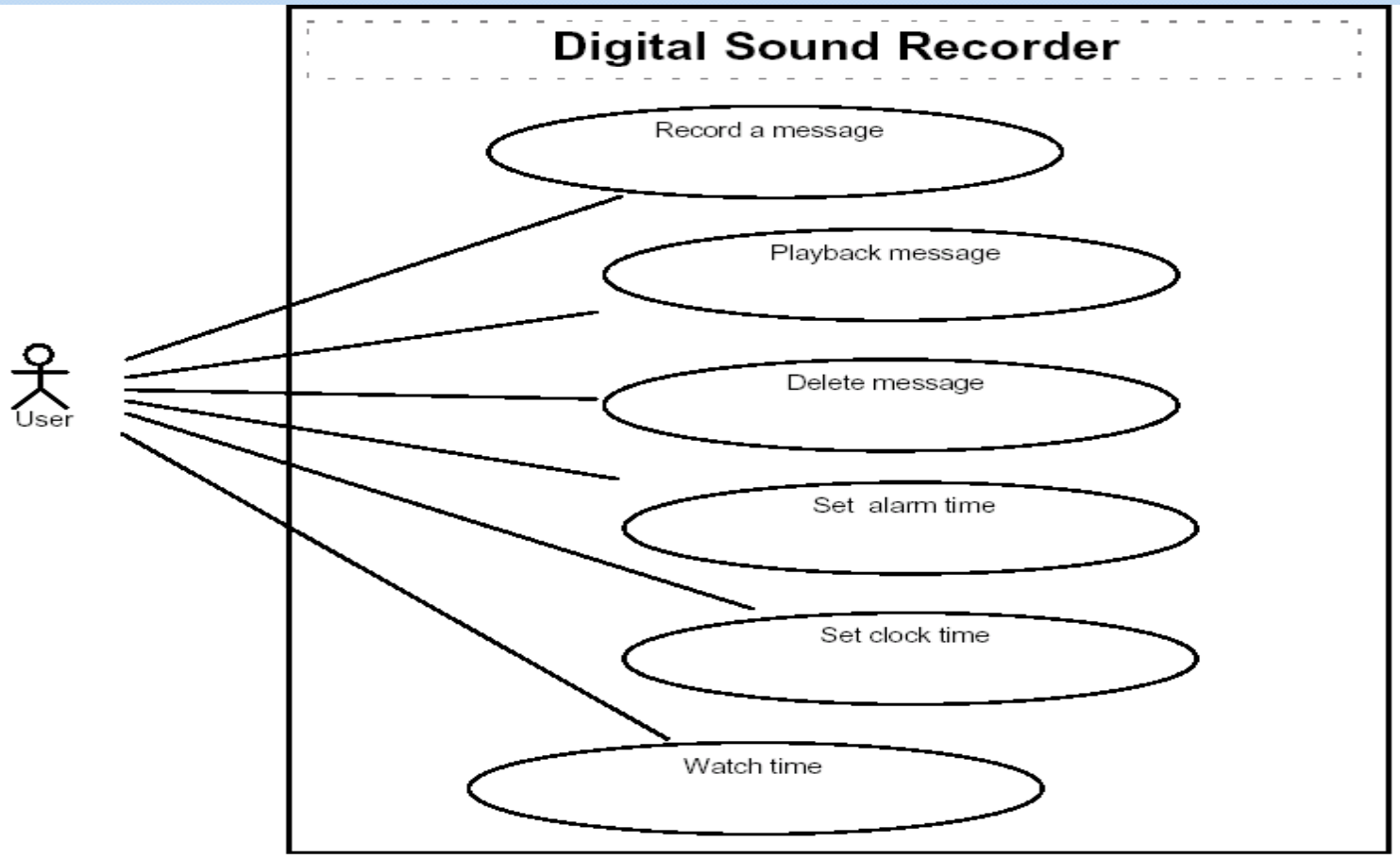



**Figure 2.3: Use Case diagram**

# The Sound Recorder
# Analysis Level Class Diagram

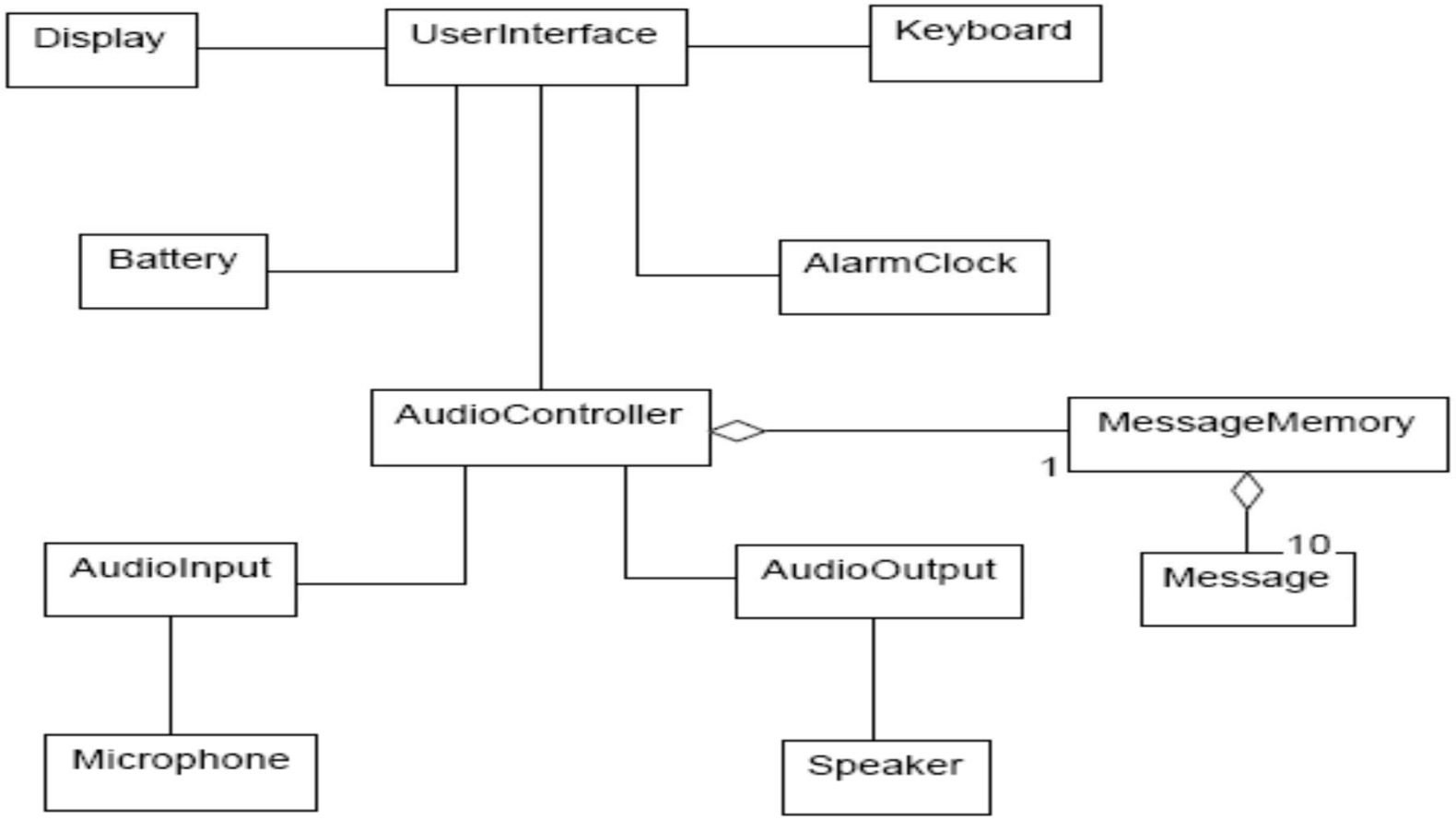



**Figure 3.2: Sound Recorder class diagram**

# Digital Sound Recorder: A Complete Example: Architecture

The Static model, Design Class Diagram: Designing The Subsystems,

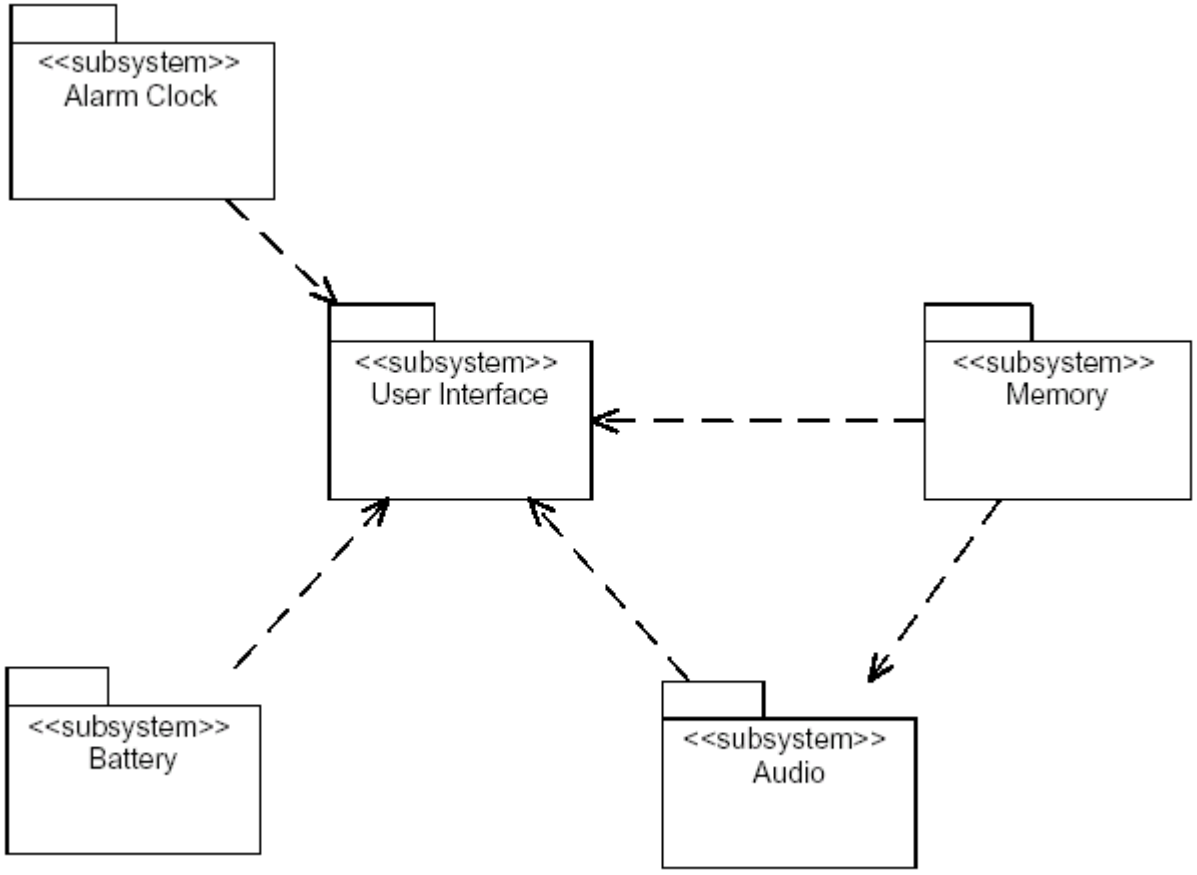



**Figure 3.3: Subsystems in the sound recorder**

# Digital Sound Recorder: A Complete Example:

Design Class Diag:
Class Operations
Are defined
Using
Design
Sequence diagrams

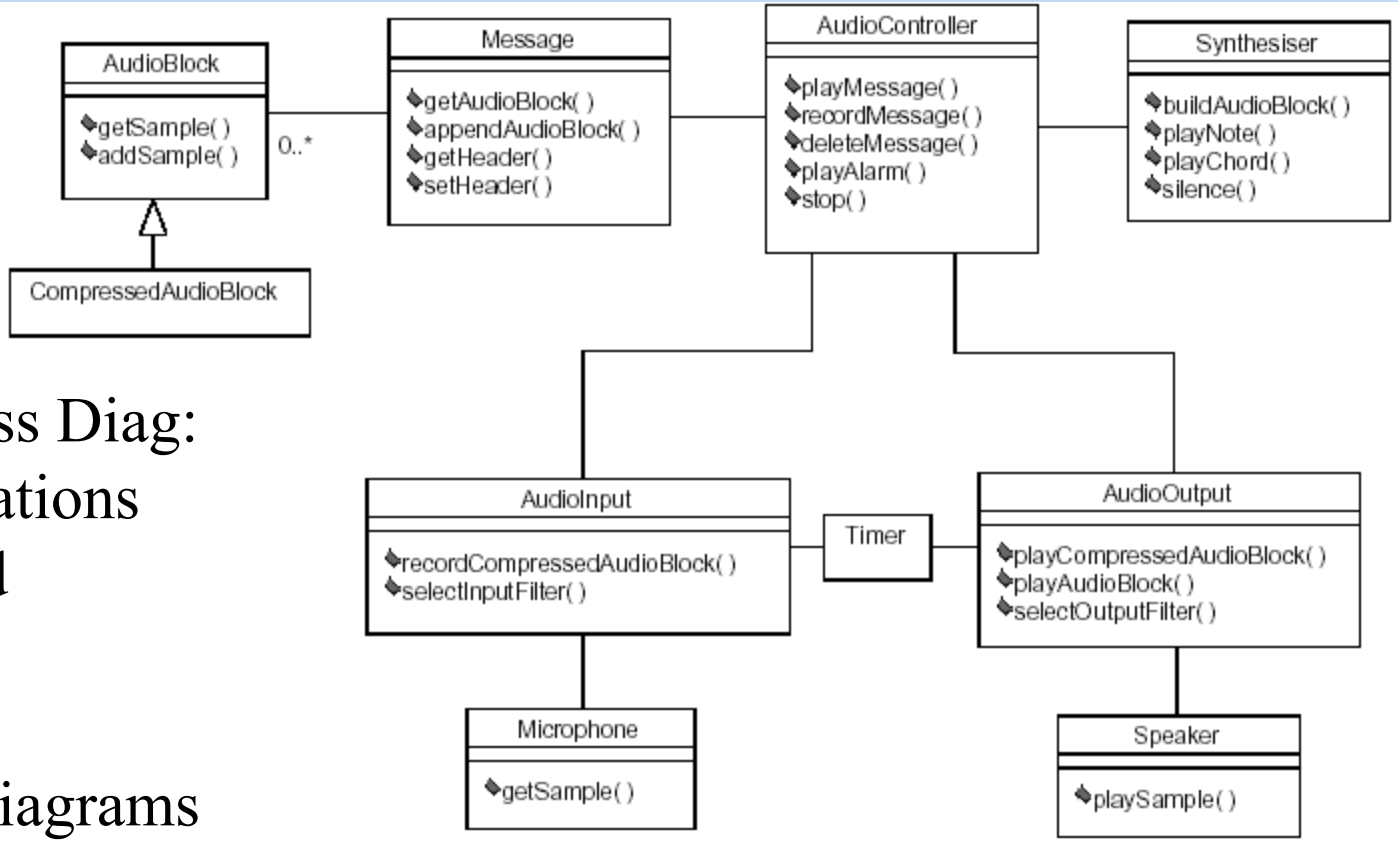



**Figure 3.4: Audio subsystem class diagram**

# Digital Sound Recorder: A Complete Example

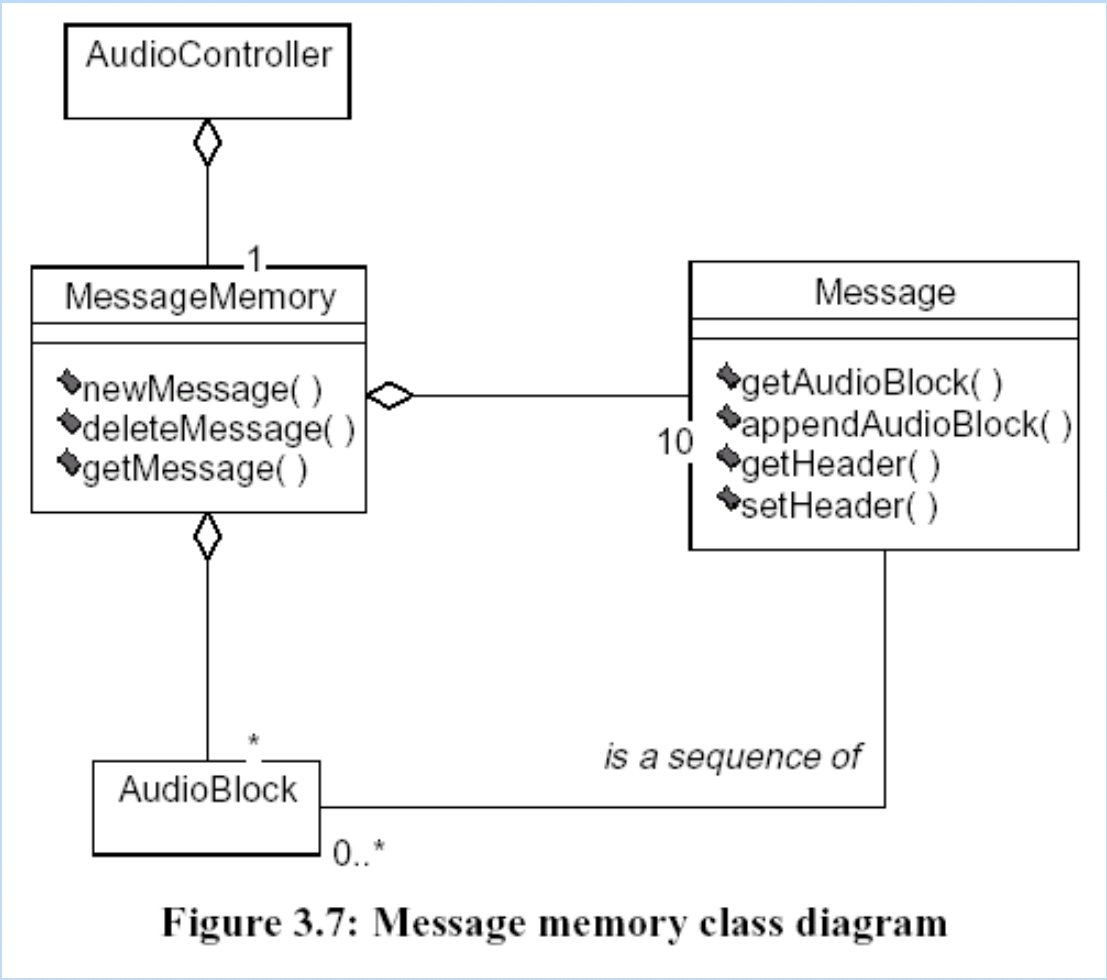



**Figure 3.7: Message memory class diagram**

# Digital Sound Recorder:

## A Complete Example: The Dynamic model

DSD shows the interactions between objects in different subsysts

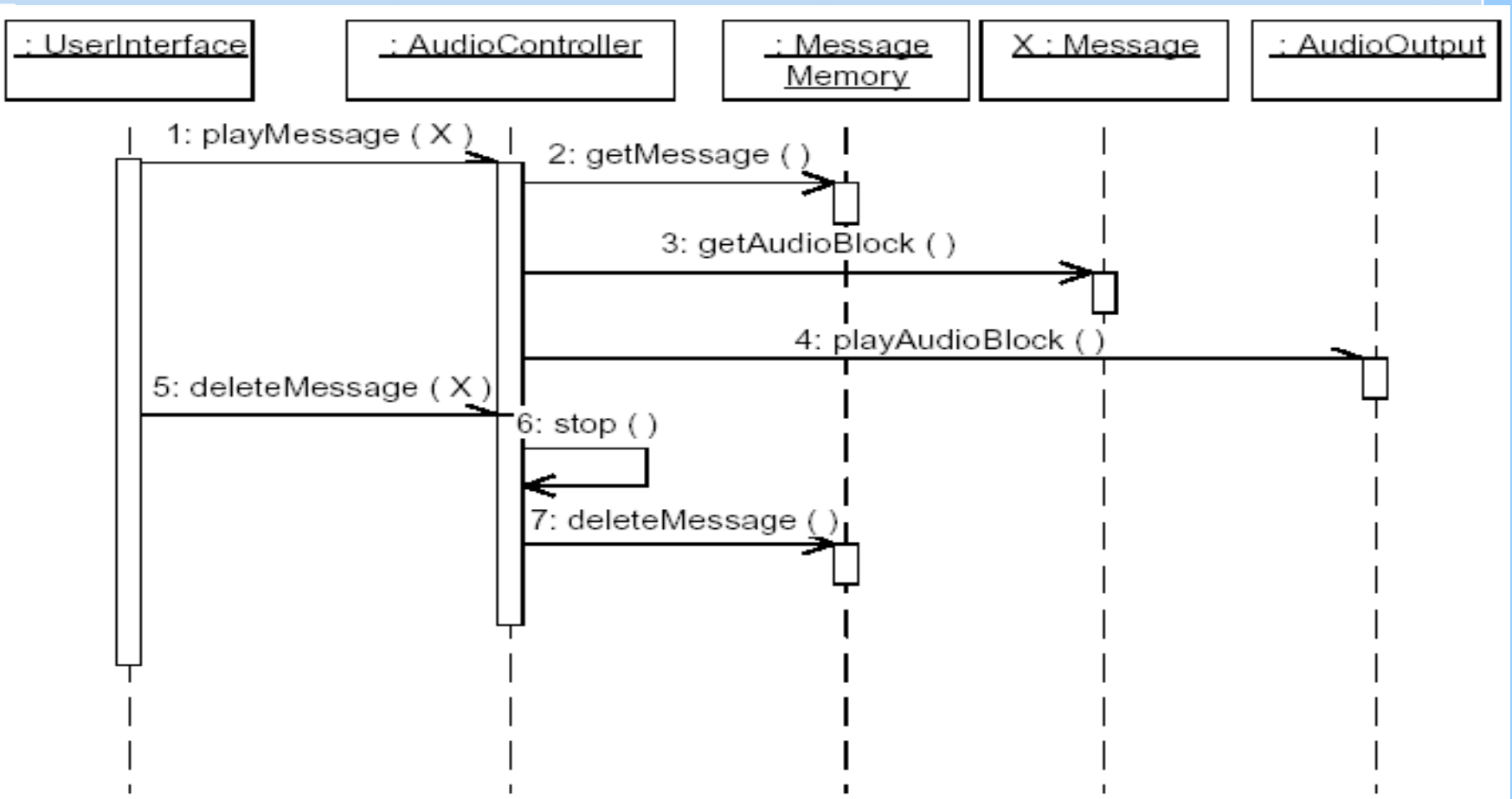



**Figure 3.8: Deleting a message while playing it**

Design Sequence Diagram

# Digital Sound Recorder: A Complete Example

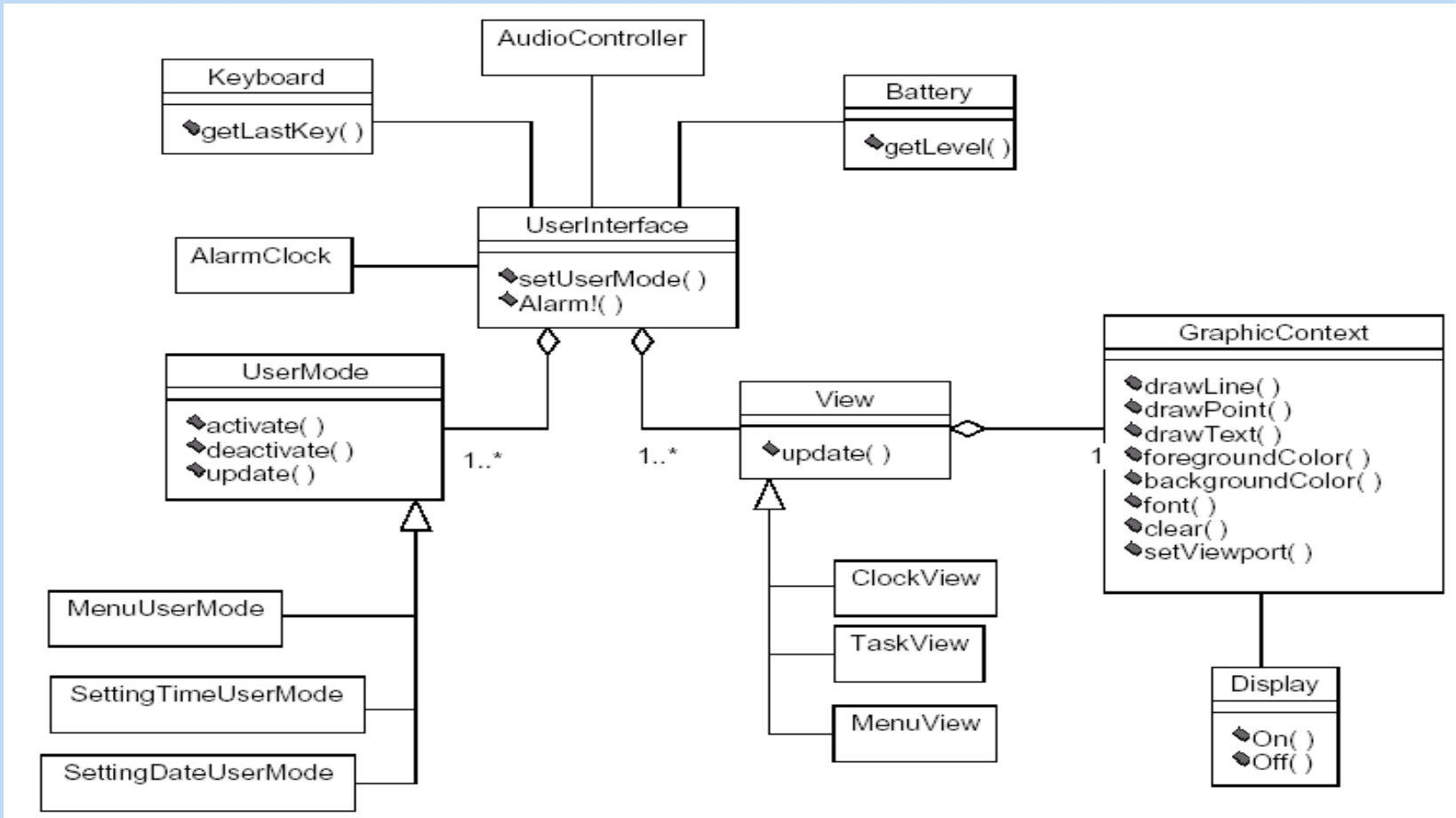



**Figure 3.11: User interface subsystem class diagram**

# Digital Sound Recorder: A Complete Example

- A Scheduler subsystem is added to provide interrupt Handling for timer interrupts to alert observers for synchronous tasks
- Uses the observer design pattern (to be discussed later)

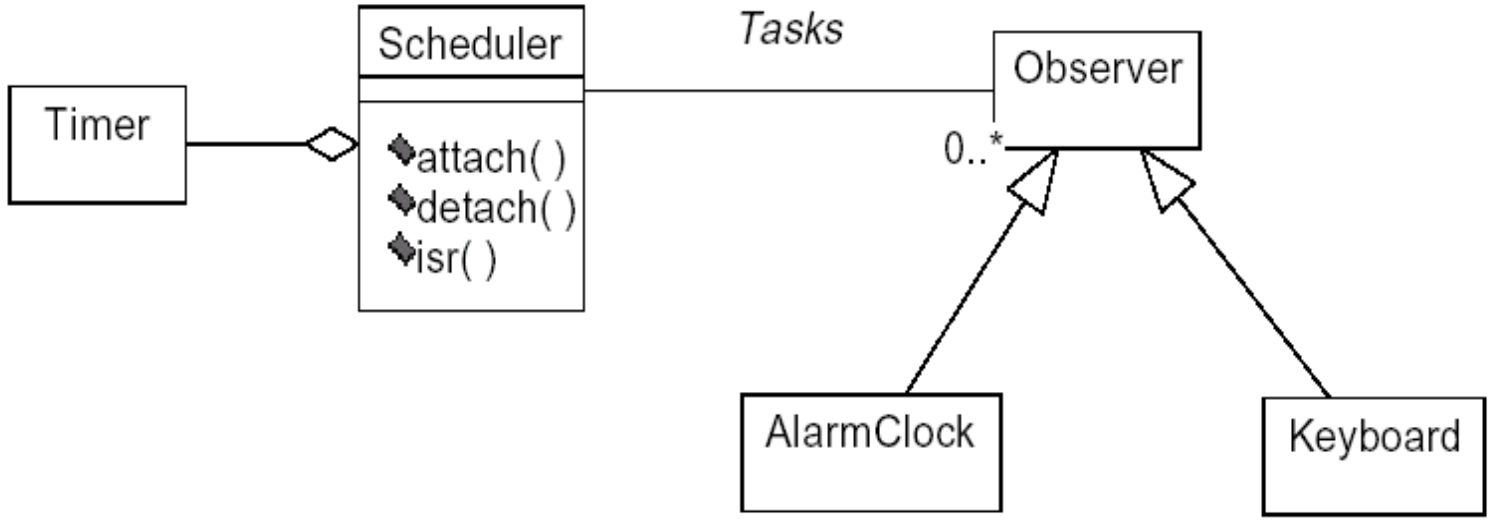
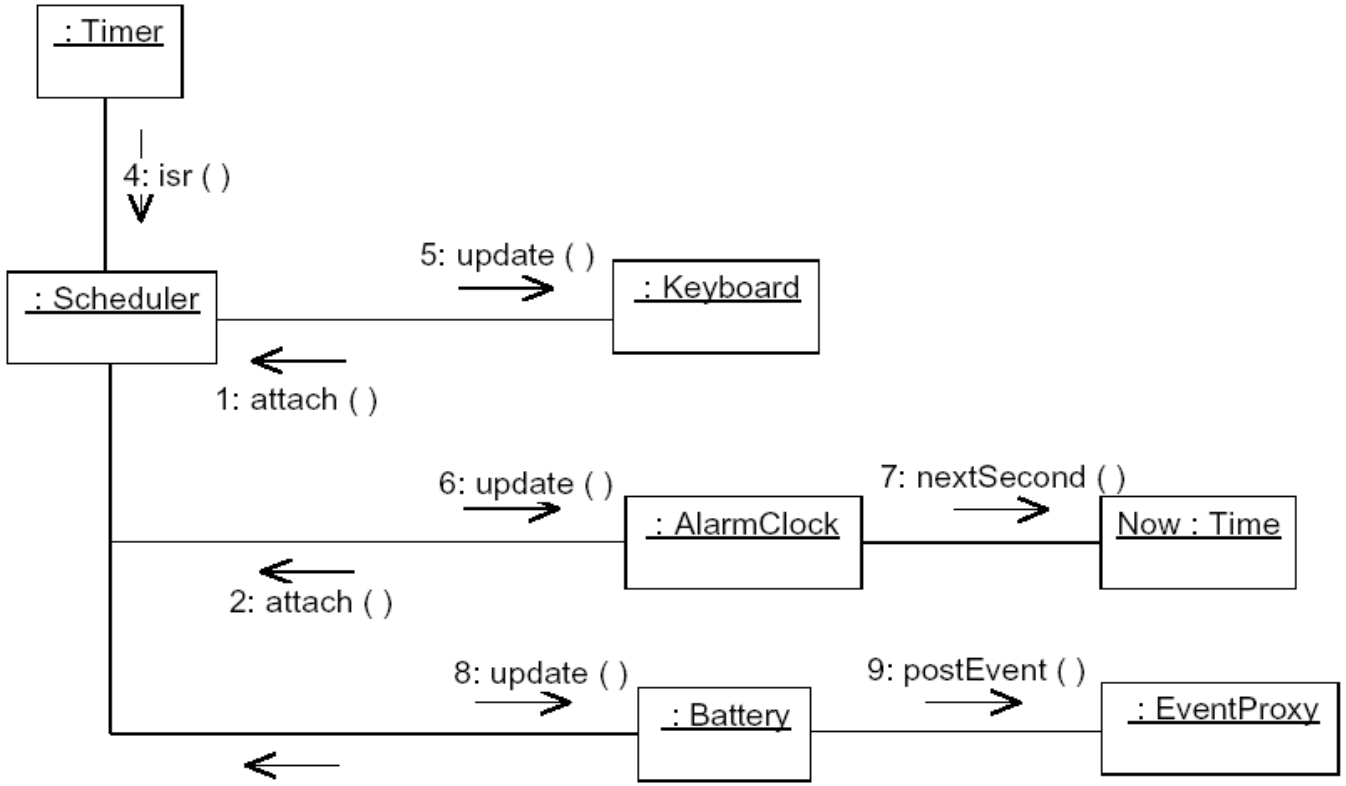



**Figure 6.1: Scheduler class diagram**

# Digital Sound Recorder: A Complete Example: The Dynamic model

- Interactions are shown using a UML collaboration diagram. Timer interrupt update scenario

Notice an EventProxy Class is added For posting Events.
Uses the Proxy Desgin pattern



# Digital Sound Recorder: A Complete Example: The Dynamic model

- Setting the alarm clock scenario: the controller object of type SettingTimeUserMode is setting the AlarmClock object which is shown by the ClockView object

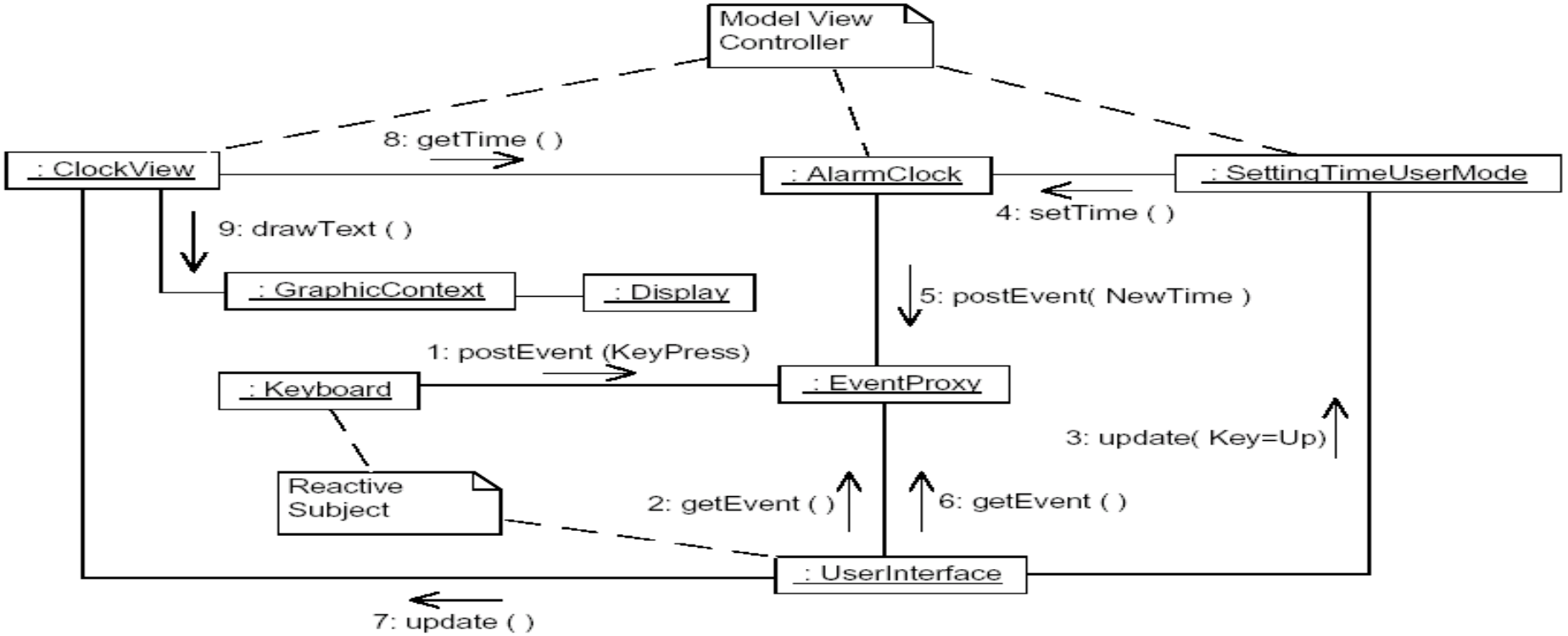



**Figure 6.3: A Model-View-Controller collaboration**

# Model-View-Controller Architecture Style

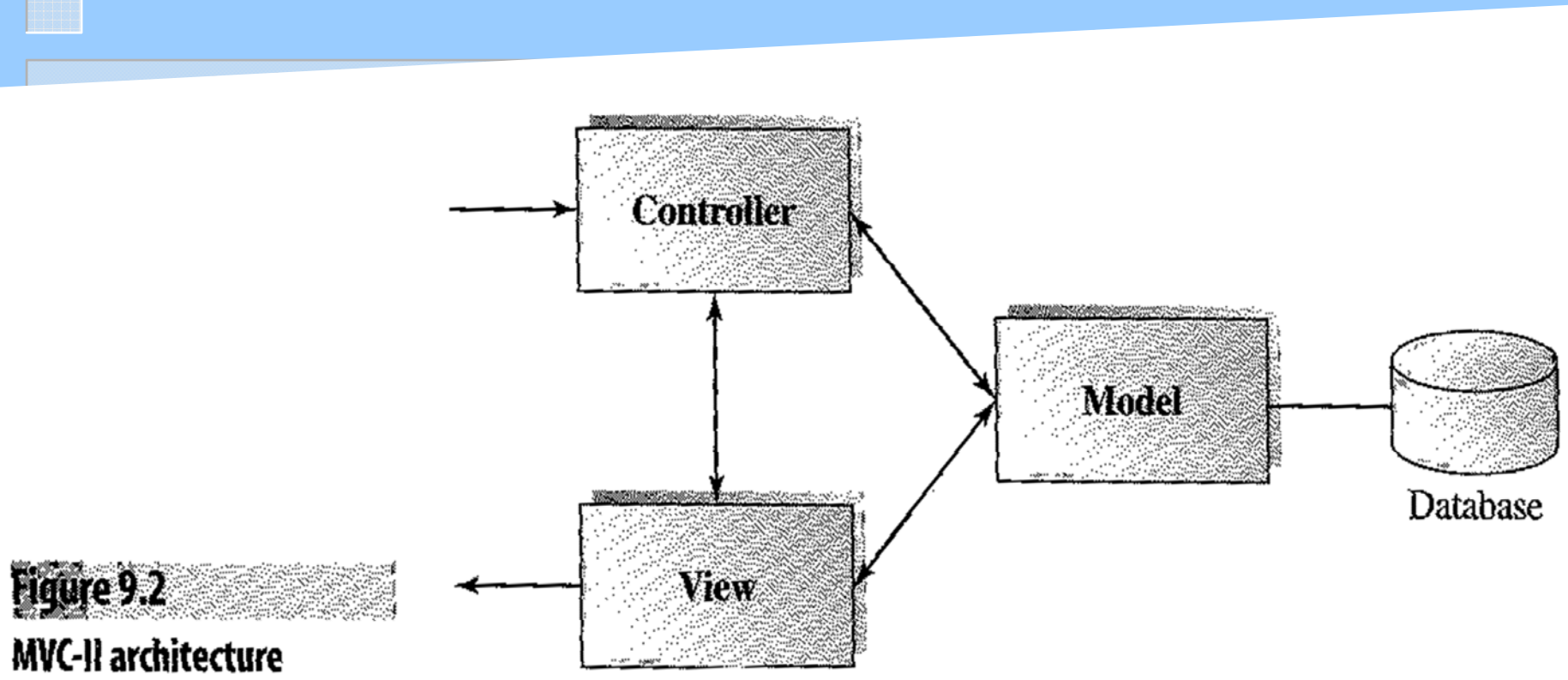



Figure 9.2
MVC-II architecture

- The Controller manipulates the data Model
- The View retrieves data from the model and displays needed information

# Model-View-Controller Architecture Style
## Dynamic Interactions

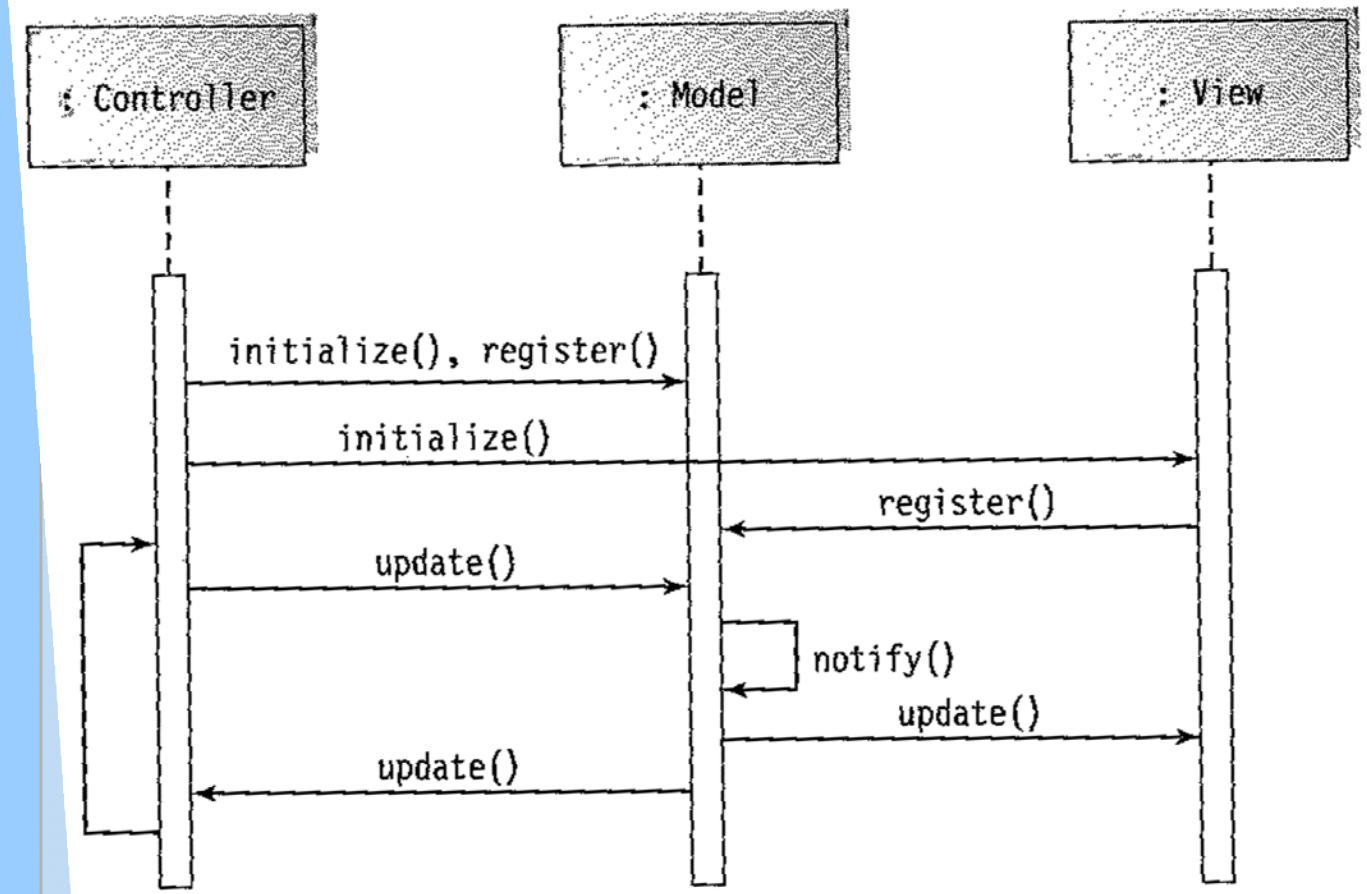



**Figure 9.4**
**Sequence diagram for MVC architecture**

# Digital Sound Recorder: A Complete Example: The Dynamic model

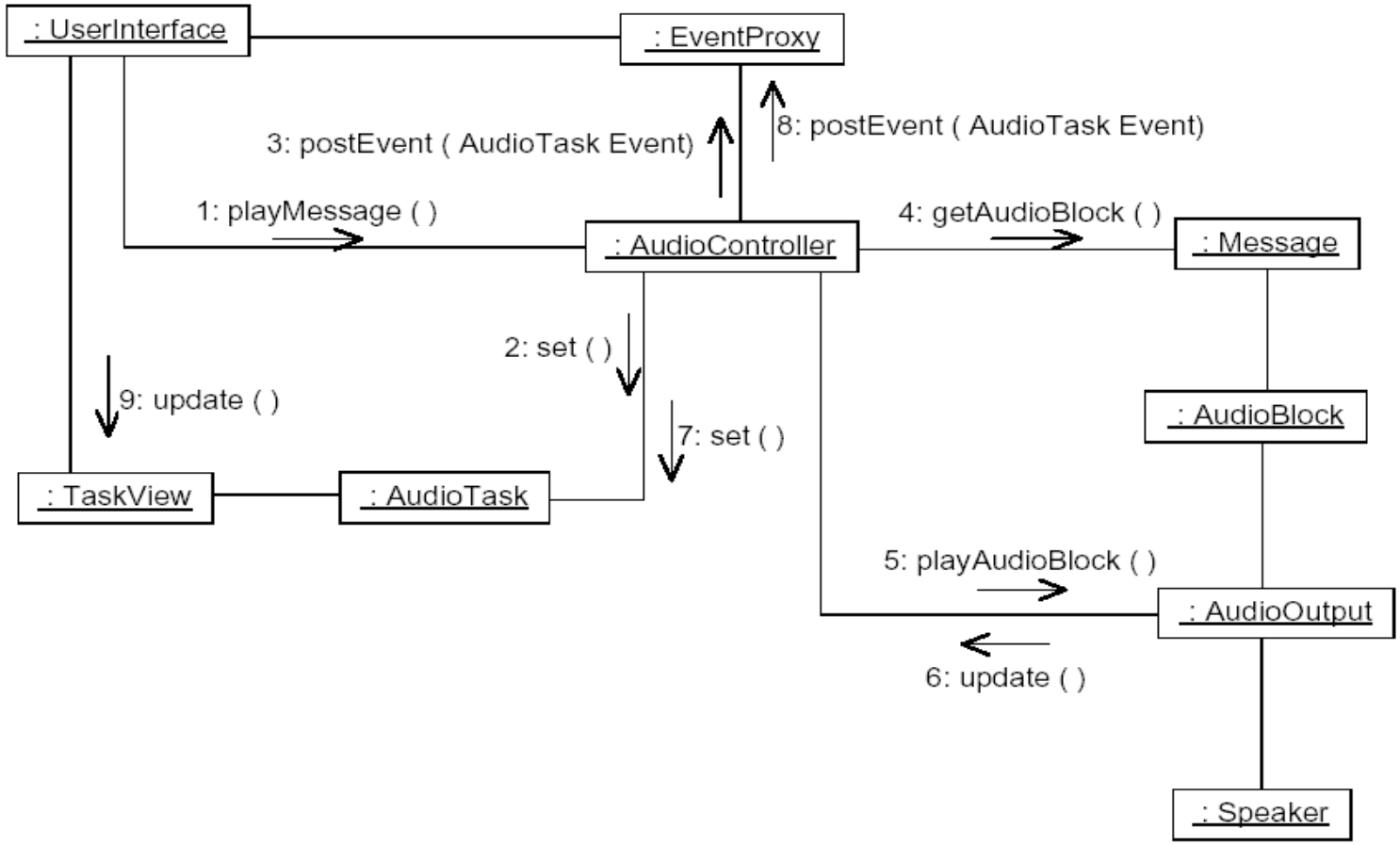



**Figure 6.4: Collaboration between the User Interface and the Audio Controller**

# **The Banking System Example:** Consolidated Collaboration (Communication) Diagrams combines static and dynamic information

Obtained by combining Multiple scenarios of Interactions

(Context Subsystem diag.)

Labeled interactions between subsystems

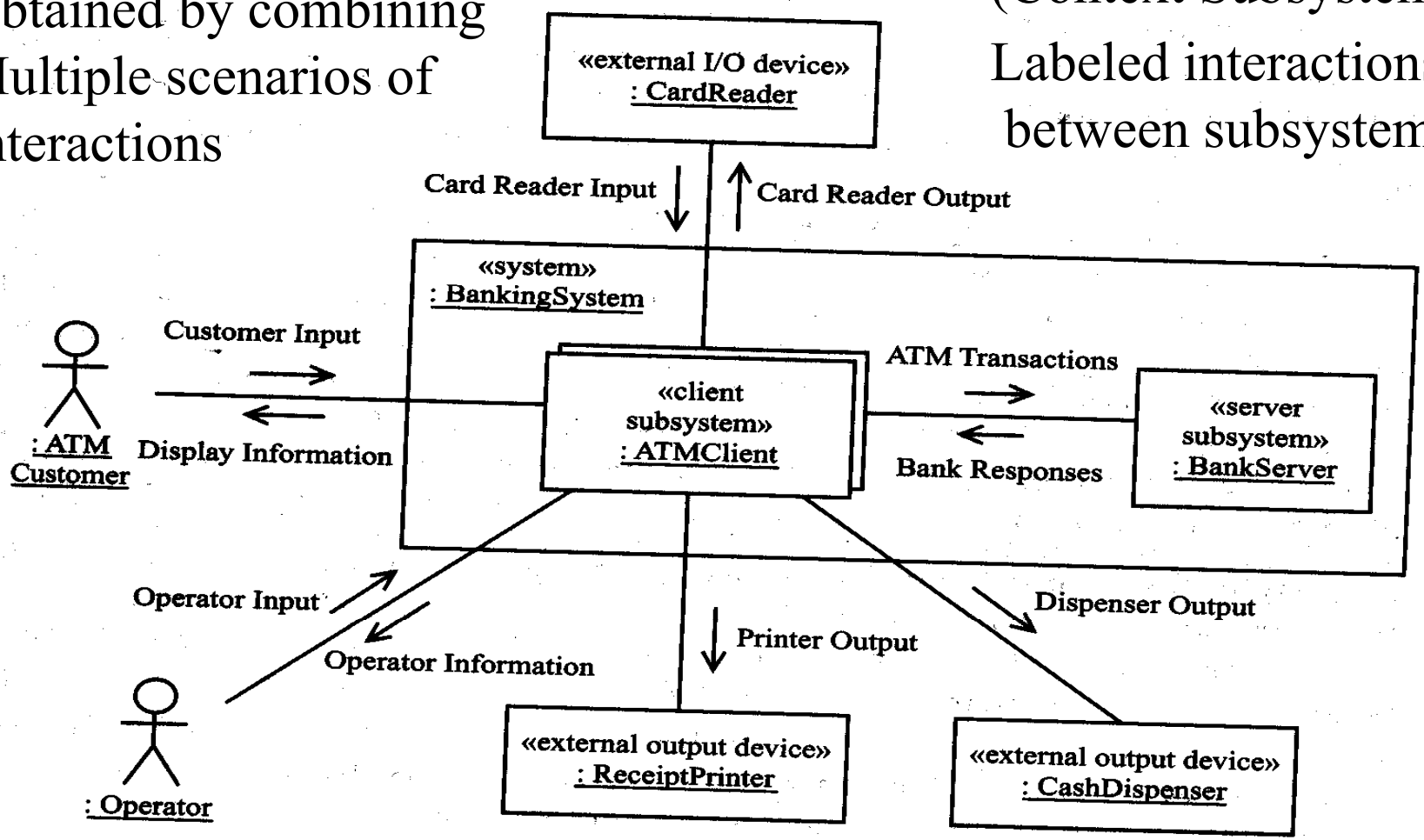



**Figure 12.6** *Example of subsystem design: high-level collaboration diagram for Banking System*

# Labels Interactions between objects in a subsystem

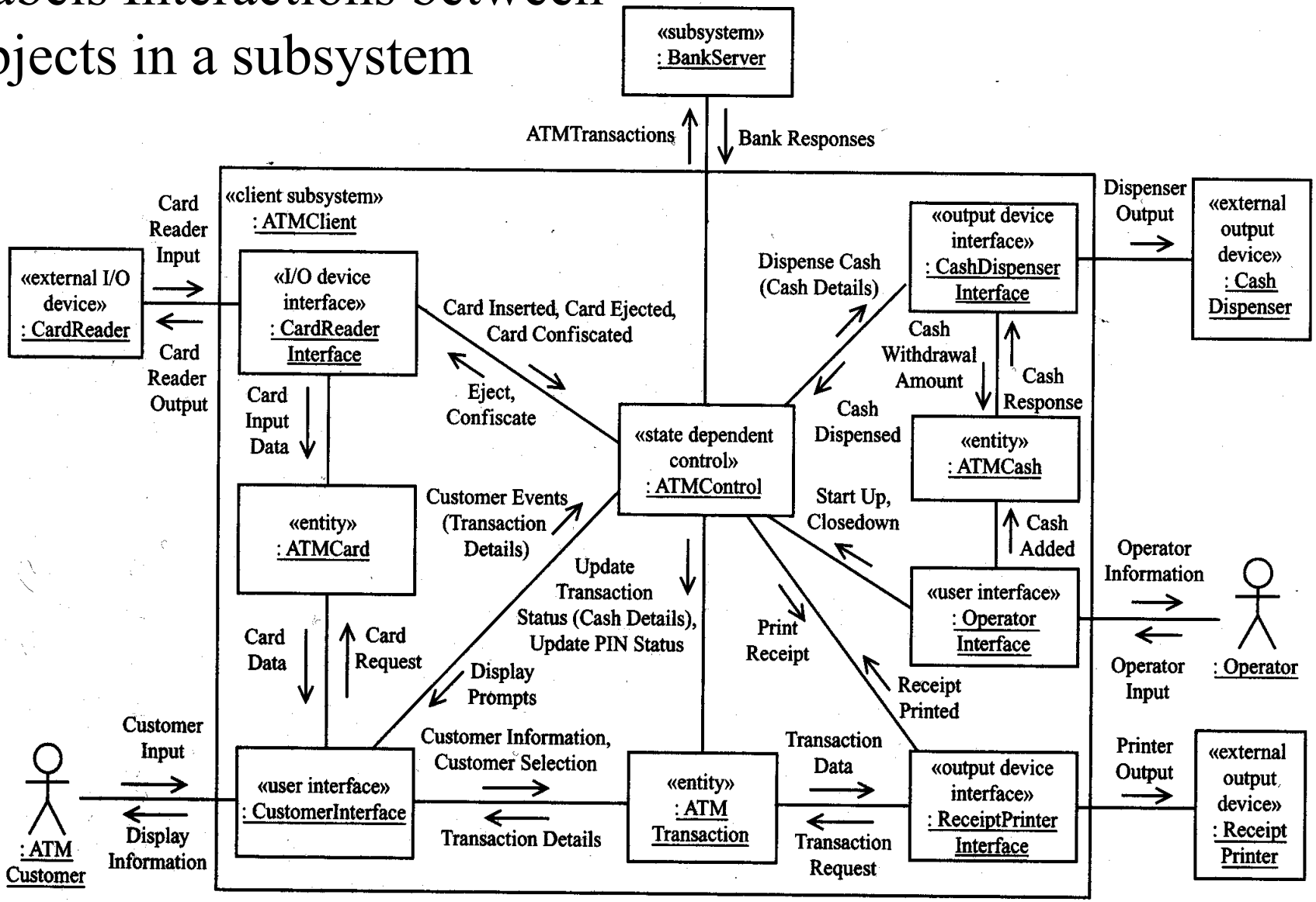



**Figure 12.5** *Example of consolidated collaboration diagram: ATM Client subsystem*

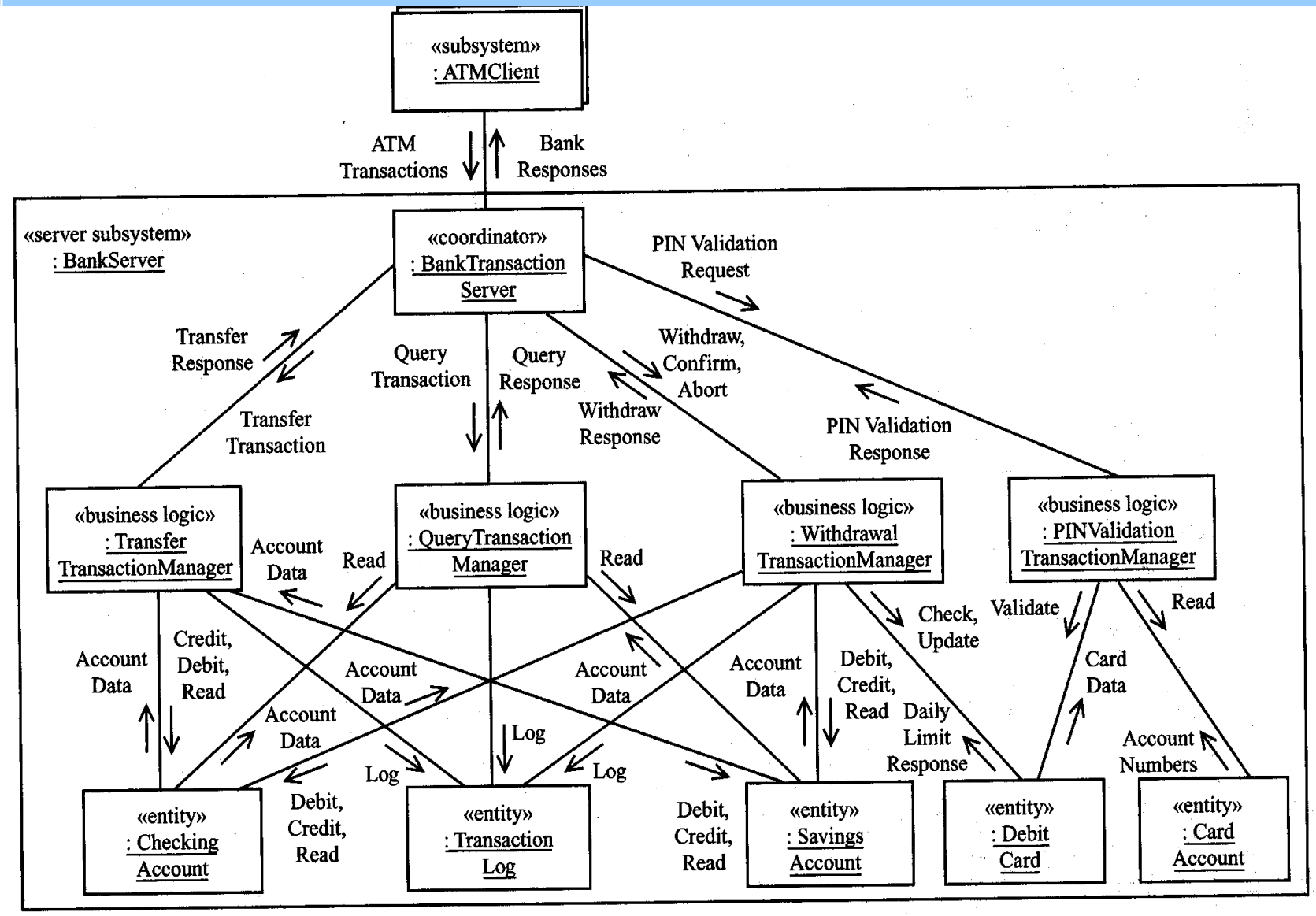



**Figure 12.13** *Consolidated collaboration diagram for Bank Server subsystem*

# Example: Consolidated Collaboration Diagram of the Elevator Control System

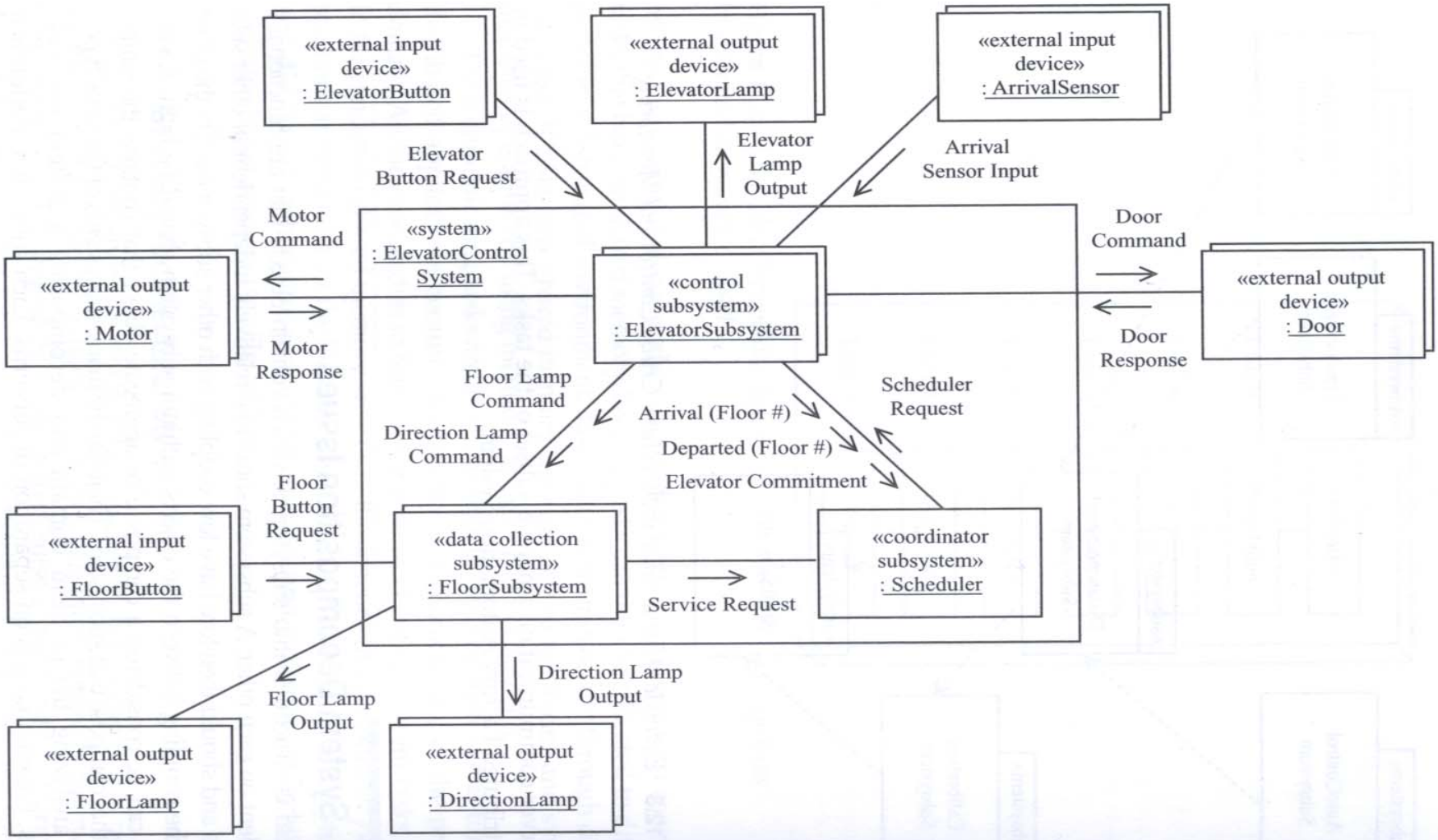



**Figure 12.4** *Example of distributed software architecture: Elevator Control System*

# Peer-to-Peer Architecture Style

### Hybrid Client-Server/Peer-to-Peer: Napster

P2P systems became part of the popular technical parlance due in large measure to the popularity of the original Napster system that appeared in 1999. Napster was designed to facilitate the sharing of digital recordings in the form of MP3 files. Napster was not, however, a true P2P system. Its design choices, however, are instructive.

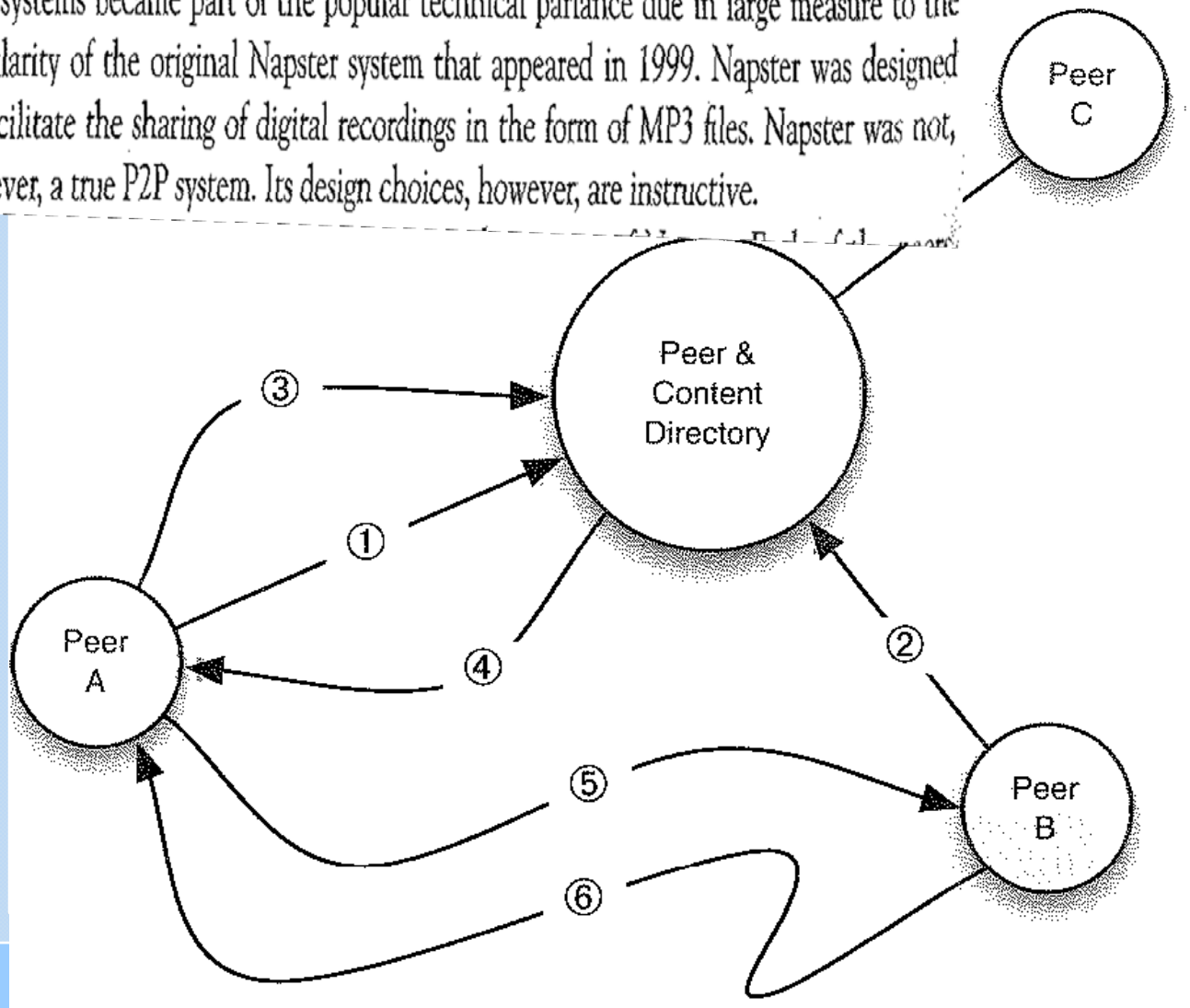



Figure 11-4. *Notional view of the operation of Napster. In steps 1 and 2, Peers A and B log in with the server. In step 3, Peer A queries the server where it can find Rondo Veneziano's "Masquerade." The location of Peer B is returned to A (step 4). In step 5, A asks B for the song, which is then transferred to A (step 6).*

# Peer-to-Peer Architecture Style
# The Gnutella Example

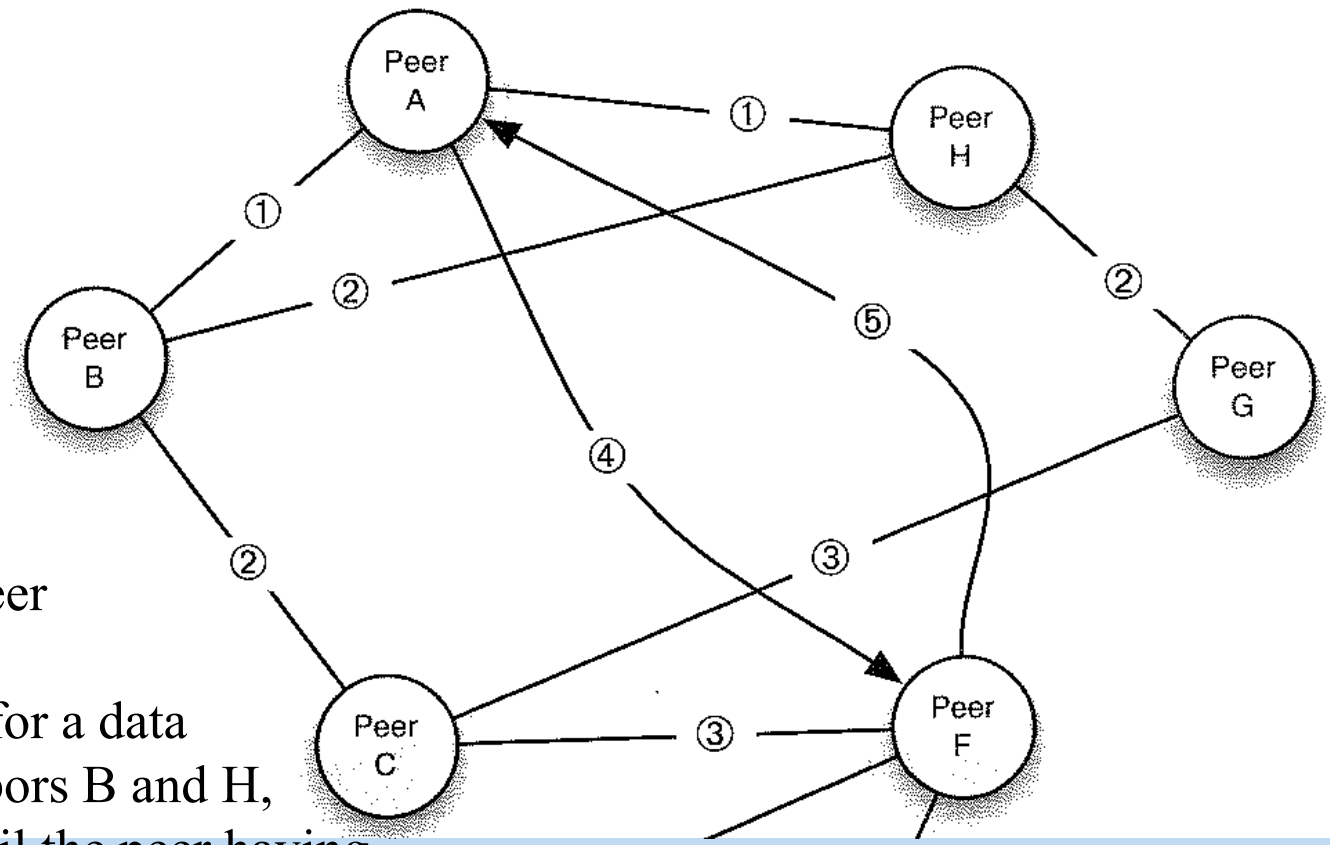
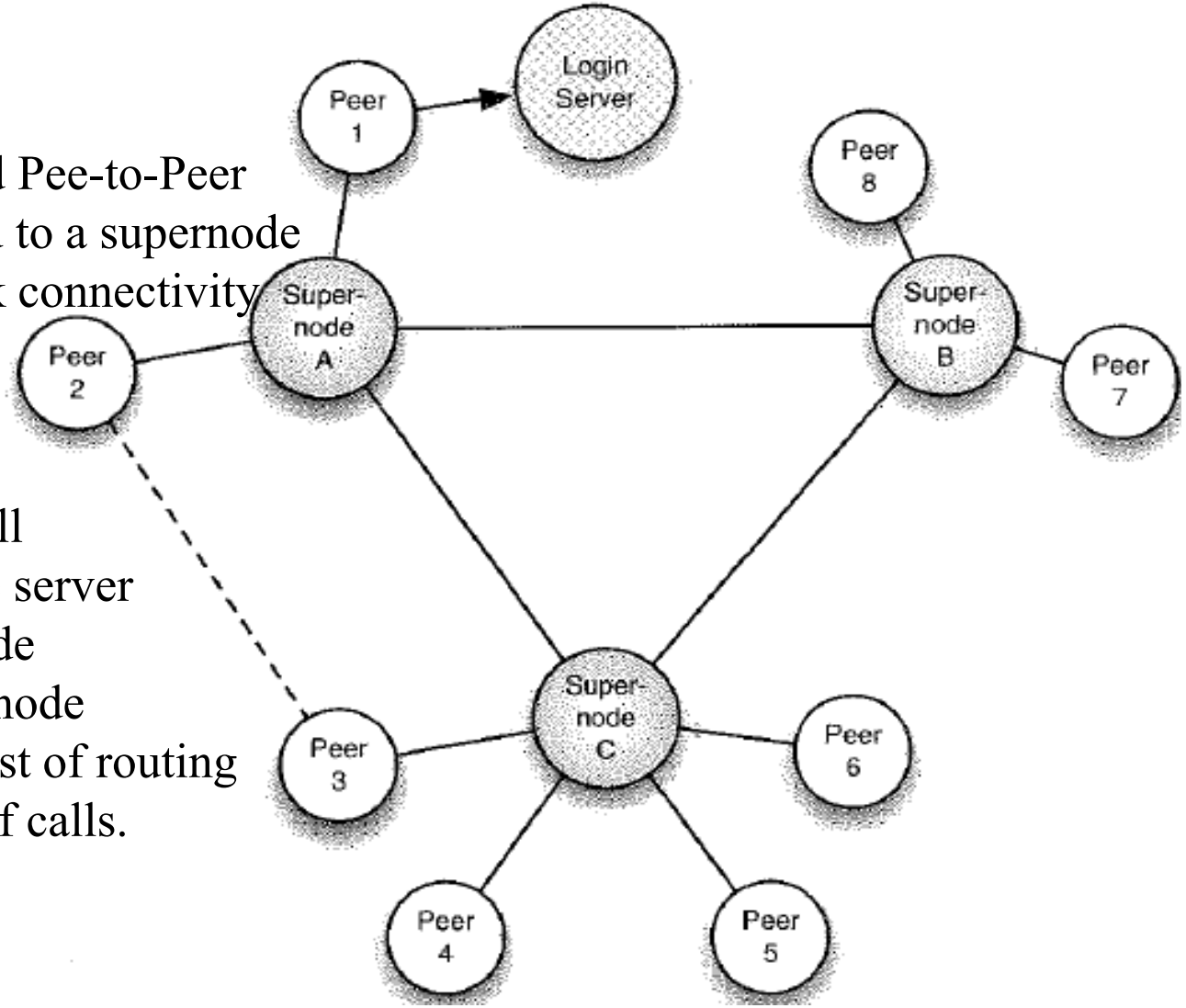



Figure 11-5. *Notional interactions between peers using the original Gnutella protocol.*

- Pure Peer-to-Peer Architecture
- A sends query for a data resource to neighbors B and H, they pass it on until the peer having the resource is found or until a certain threshold of hops is reached

# Peer-to-Peer Architecture Style
# The Skype Example

Figure 11-6.
*Notional instance of the Skype architecture.*




- A mixed client-Server and Pee-to-Peer
- Skype Peers get promoted to a supernode status based on their network connectivity And machine performance
- Supernodes perform the Communication and routing of massages to establish a call
- When a user logs in to the server he is connected to a supernode
- If a peer becomes a supernode he unknowingly bears the cost of routing a potentially large number of calls.