Lecture 8:
Architectural Design of Distributed Applications

Reference: H. Gomaa, Chapters 12, 13, 15 - *Software Modeling and Design*, Cambridge University Press, 2011

Steps in Using COMET/UML

1 Develop Software Requirements Model

2 Develop Software Analysis Model

3 **Develop Software Design Model**

- Design Overall Software Architecture (Chapter 12, 13)
- **Design Distributed Applications (Chapter 12,13,15)**
- Structure Subsystems into Concurrent Tasks (Chapter 18)
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design (Chapter 14)

Architectural Design of Distributed Applications

- Distributed processing environment
  - Multiple computers communicating over network
- Typical distributed applications
  - Distributed real-time applications
    - Data collection/control
  - Client / server applications
  - Service-oriented applications

3

Characteristics of Distributed Applications

- Structure of component-based distributed application
  - Consists of one or more subsystems
  - Each subsystem designed as a distributed component
  - Execute on multiple nodes in distributed configuration
- Structure of component-based subsystem
  - Consists of one or more objects
  - Objects all execute on same node
- Communication between component-based subsystems
  - Message communication

4

Steps in Designing Distributed Applications

- Designing distributed applications
  - System decomposition, subsystem decomposition, system configuration
- System Decomposition
  - Decompose system into distributed subsystems
    - Subsystem Structuring Criteria
    - Distributed subsystems (components)
      - Concurrent and self-contained object with a well-defined interface
      - Logical unit of distribution and configuration (deployment)
  - Define message communication interfaces

5

Steps in Designing Distributed Applications

- Subsystem Decomposition
  - Structure subsystem into active objects (tasks) and passive objects
- System Configuration
  - Define component-based subsystem's instances of target system
  - Map to hardware configuration

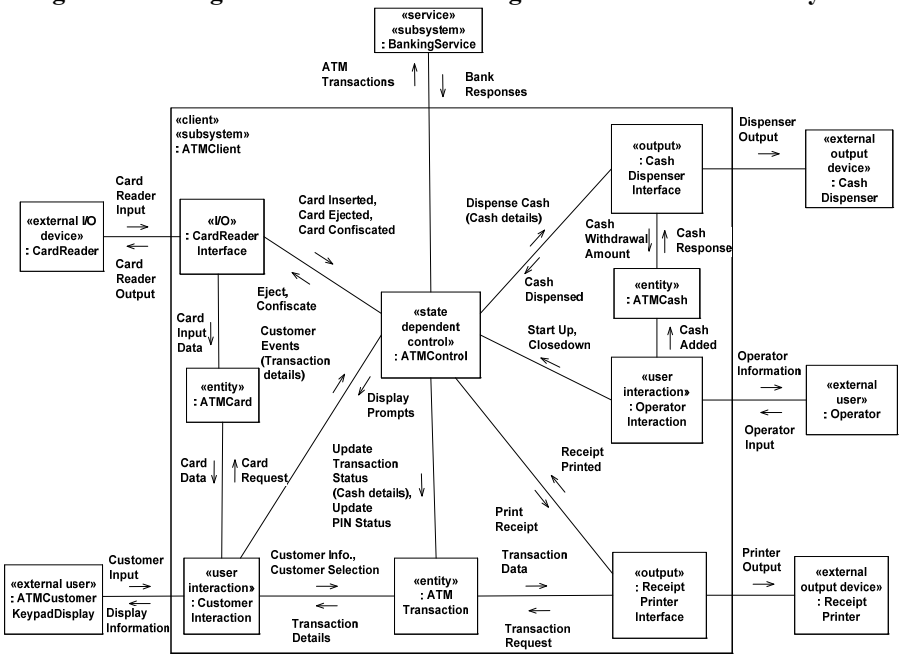
6

Subsystem Structuring Criteria

- Client
  - Requester of one or more services (e.g., Fig. 13.7)
- User Interaction
  - Collection of objects supporting needs of user (e.g., Fig. 13.6, 13.10)
- Service
  - Provides service for client subsystems (e.g., Fig. 13.5, 13.7)
- Control
  - Subsystem controls given part of system (e.g., Fig. 13.10)
- Coordinator
  - Coordinates several control subsystems (e.g., Fig. 13.10)
- Input / Output
  - Performs I/O operations for other subsystems (e.g., Fig. 13.5)

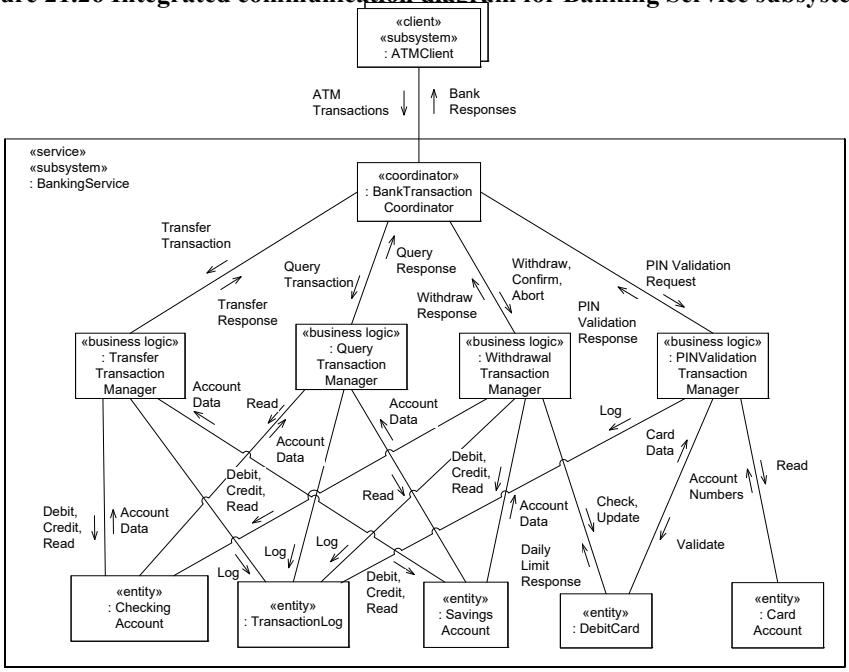
7

Figure 13.2 Integrated communication diagram for ATM Client subsystem



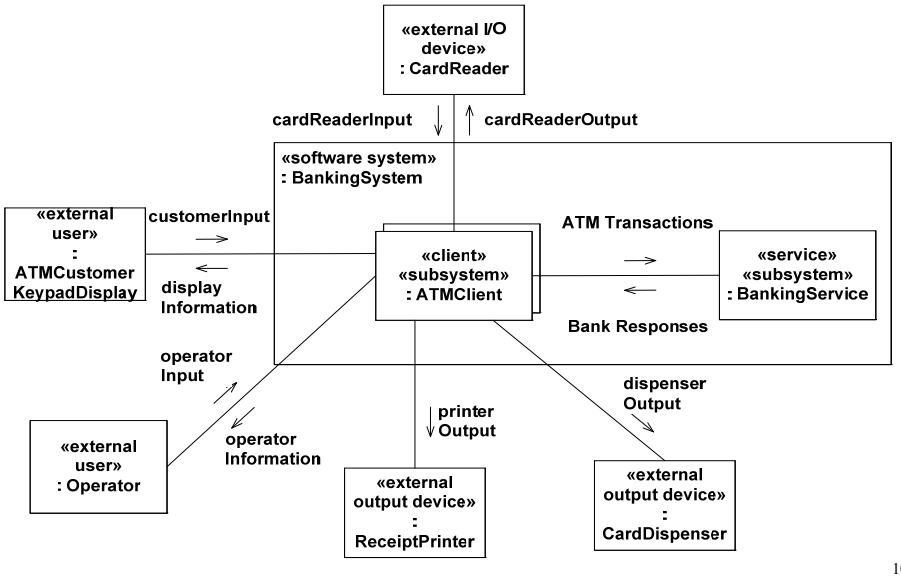
8

Figure 21.26 Integrated communication diagram for Banking Service subsystem



9

Figure 21.27 Subsystem design – high level communication diagram for Banking System



10

Design Distributed Subsystem Interfaces

- Message Communication between distributed subsystems
  - Asynchronous message communication
    - Peer to peer communication
  - Synchronous message communication
    - Client / Service message communication
  - Group Message Communication
    - Broadcast message communication
    - Multicast message communication
  - Brokered Communication
    - Uses Object Broker
- Also referred to as message communication patterns

11

Asynchronous Message Communication (Loosely Coupled)

- Producer sends message and continues
- Consumer receives message
  - Suspended if no message is present
  - Activated when message arrives
- Message queue may build up at Consumer

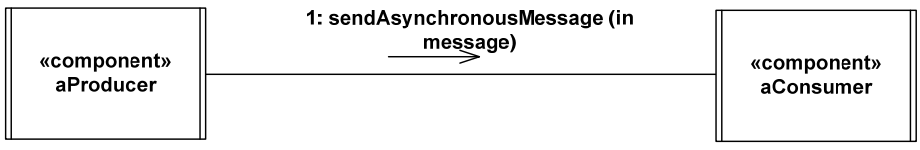
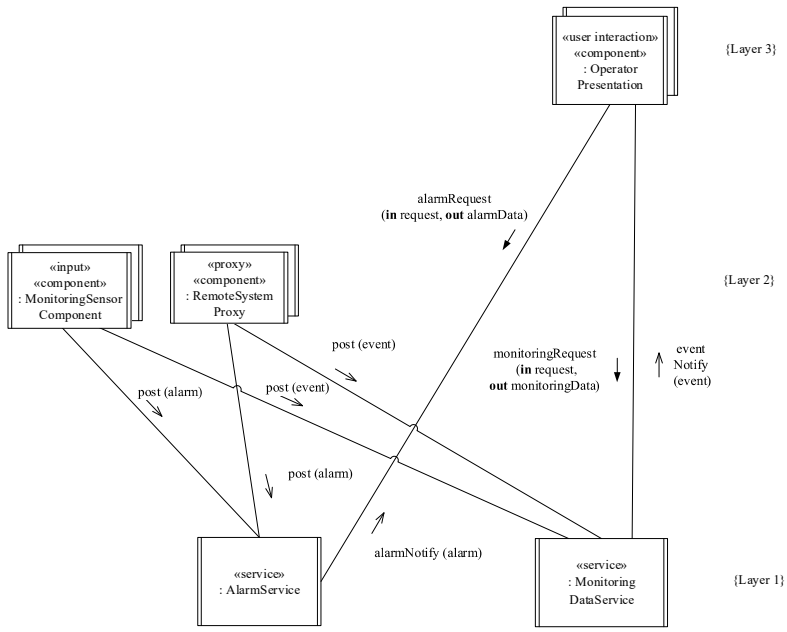


Figure 12.8 Asynchronous message communication

12

Figure 23.10 Concurrent communication diagram for Emergency Monitoring System



Synchronous Message Communication with Reply (Tightly Coupled)

- Client task sends message and waits for reply
- Service receives message
  - Suspended if no message is present
  - Activated when message arrives
  - Generates and sends reply
- Client receives reply and continues

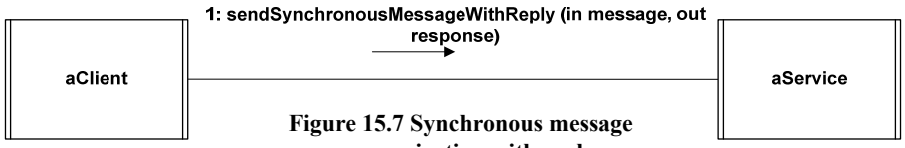


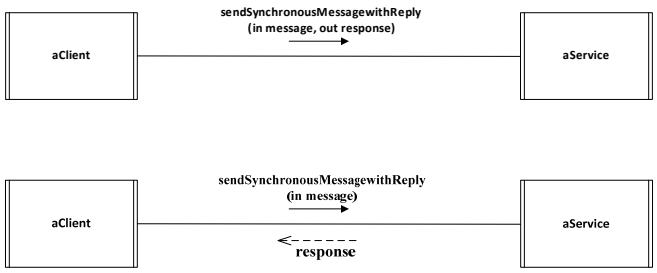
Figure 15.7 Synchronous message communication with reply

14

Synchronous Message Communication with Reply

- Two UML notations – either one can be used

Figure 4.12 Synchronous Message Communication with Reply

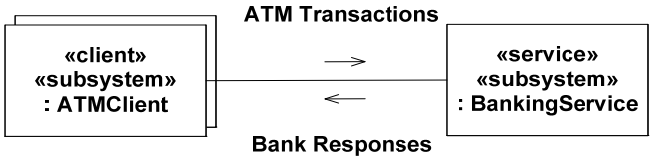


15

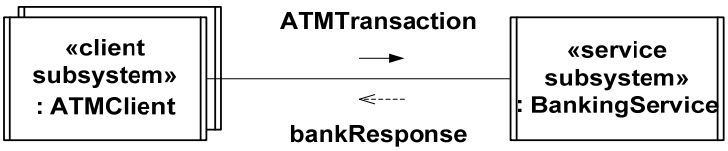
15

Example of Synchronous Message Communication With Reply

Communication diagram – Before subsystem interface design



Communication diagram – After subsystem interface design

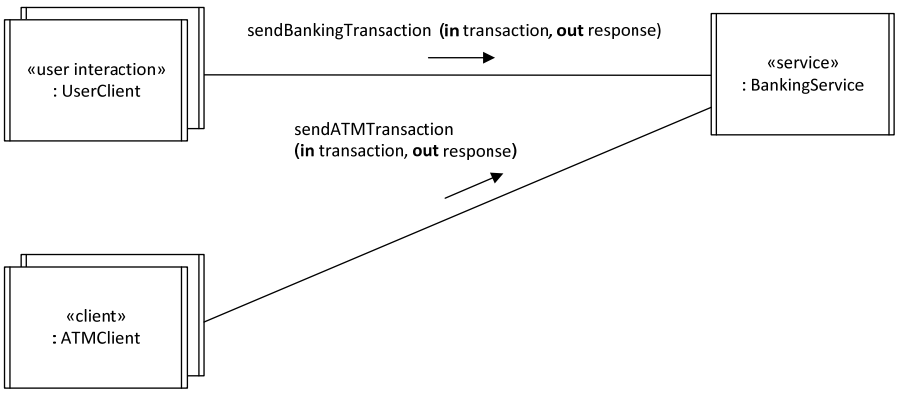


dm-16

16

Example of High-Level Communication Diagram for Client/Server application -
Banking Service with two different clients

- Two client subsystems communicate with one service subsystem (Figure 15.8)



17

17

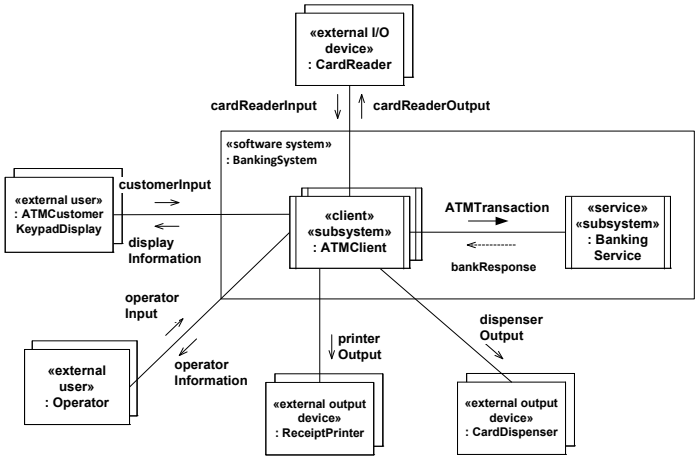
Design of Banking Service Subsystem

- Banking Service
  - Receives messages from ATM Clients in order received
  - First-in-First-Out (FIFO)
- Synchronous communication with reply
  - ATM Client subsystem
    - Sends message
    - Waits for reply
  - Banking Service subsystem
    - Receives message
    - Performs requested transaction
    - Sends response to ATM Client

18

18

Figure 12.2 Design of Subsystem interfaces –high level communication diagram for Banking System

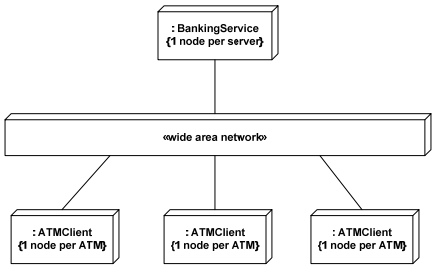


19

19

Figure 21.36 Deployment diagram for Banking System

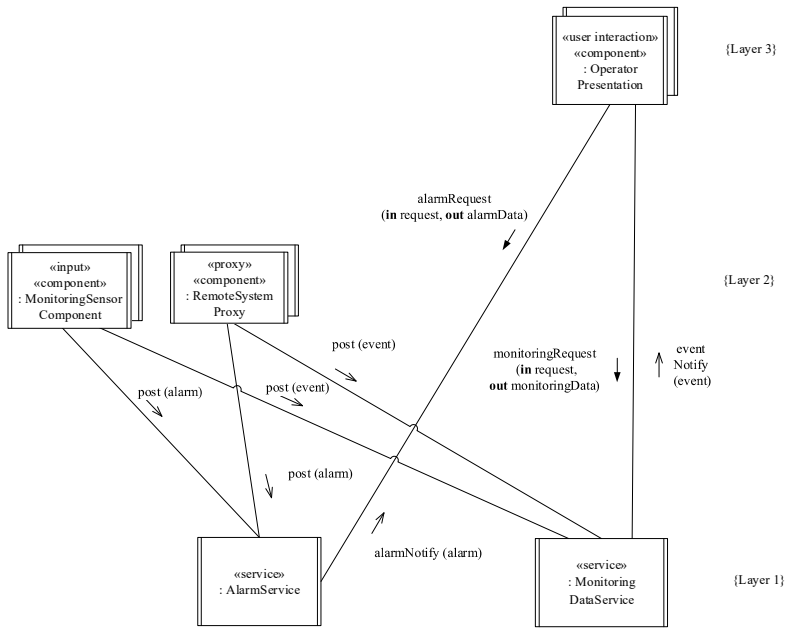
- Depicts instances of software components mapped to hardware configuration



dm-20

20

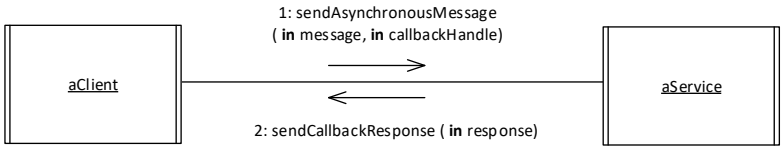
Figure 23.10 Concurrent communication diagram for Emergency Monitoring System



Asynchronous Message Communication With Callback

- Client sends a request with a client operation handle (reference) to the server and can continue
  - Without waiting for the server response
  - However, the client does need the service response later
  - A callback is an asynchronous response to a client request
- This pattern allows the client to execute asynchronously
  - Used between a client and a server
  - But still follow the client/server paradigm
  - A client sends only one message at a time to the server
    - Receives a response from the server

Figure 15.9: Asynchronous message communication pattern with callback pattern



dm-23

23

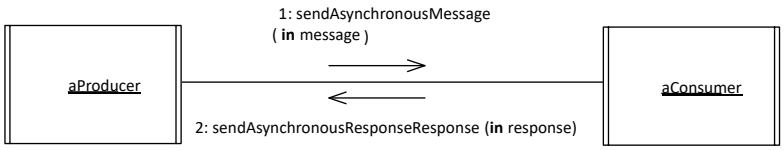
Bidirectional Asynchronous Message Communication

- Producer sends a request with a client operation handle (reference) to the server and can continue without waiting the response from the Consumer
  - But, Producer does need the service response later
  - Producer can send several requests before receiving first reply
  - Use two message queues between P and C
  - Possible for the queues to overflow due to processing speed

dm-24

24

Bidirectional Asynchronous message communication pattern



dm-25