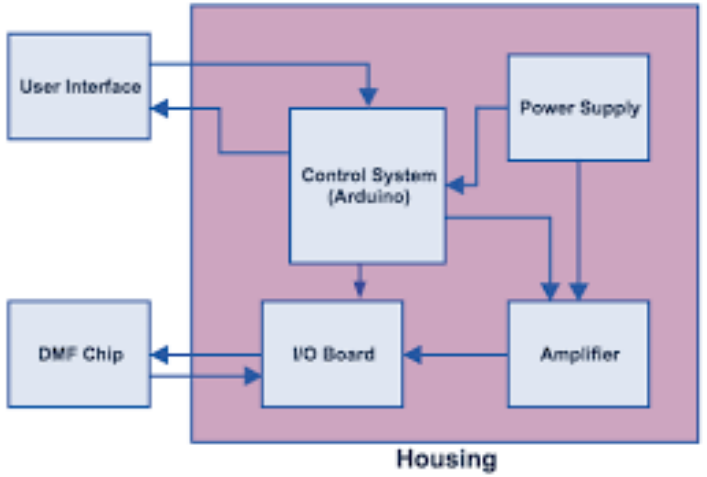
Software Subsystem Architectural Design

Sadegh Sulaimany

info@Bioinfotmation.ir



Initial assessment

1. What is difference and similarities between aggregation and composition?

2. What is the integrated communication diagram?

Session titles

- Idea
- General Process
- Issues in Subsystem architectural design
- Start point of subsystem determination
- Integrated communication diagrams
- Separation of concepts in subsystem design
- Subsystem structuring criteria
- Decision about message communication

Idea

- Successful management of inherent complexity of a large-scale software system
  - an approach for decomposing the system into subsystems
    - Then developing the overall software architecture of the system
  - After performing this decomposition, each subsystem can then be designed independently

General process

- To design the software architecture
  - it is necessary to start with the analysis model
  - Integrate the use case–based interaction models into an initial software architecture
  - Determine the subsystems using separation of concerns and subsystem structuring criteria
  - Determine the precise type of message communication among the subsystems

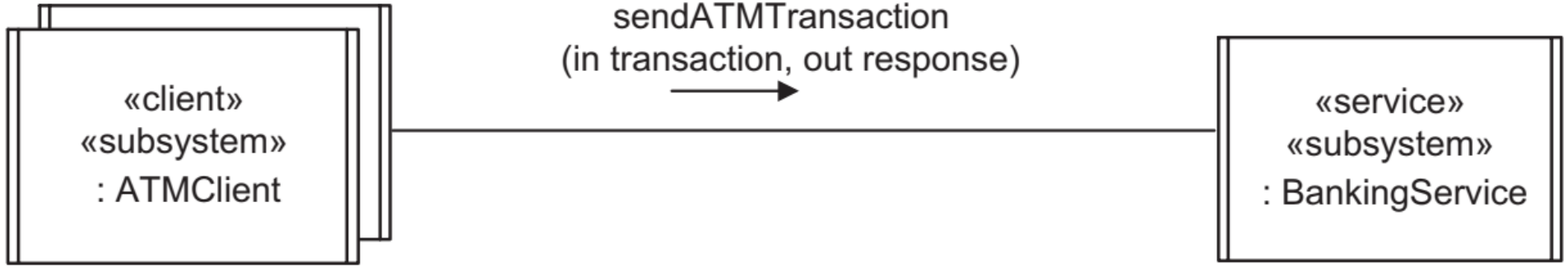
Issues in Subsystem Architecture design

- Decomposing system into subsystems
  - emphasis is on functional decomposition
    - such that each subsystem addresses a distinctly separate part of the system
    - The goal
      - for each subsystem to perform a major function that is relatively independent of the functionality provided by other subsystems
      - A subsystem can be structured further into smaller subsystems
    - After that, interface between subsystems should be defined

Start point of subsystem determination

- Some subsystems can be determined relatively easily
  - because of geographical distribution or server responsibility

  - most common forms of geographical distribution
    - clients and services,
    - allocated to different subsystems:
    - **a client subsystem** and **a service subsystem**
    - Example:



High-level software architecture: Banking System

Start point of subsystem determination

- In many applications, it is not so obvious how to structure the system into subsystems

  - a good place to start is with the use cases
    - one of the goals of subsystem structuring
      - have objects that are functionally related and highly coupled in the same subsystem
    - Objects that participate in the same use case
      - candidates to be grouped into the same subsystem
      - Their dynamic interactions are almost related
      - High coupling that subsystem and low coherency with other subsystems
      - **Note:**
        - Logically, each object should be assigned to just one subsystem

Integrated communication diagrams

- Initialize architecture from analysis
  - by integrating the use case–based interaction diagrams
  - Depicting with communication (collaboration) diagrams

- Integrated communication diagrams
  - synthesis of all the communication diagrams developed to support the use cases
    - Merging communication diagrams
    - The order of the synthesis of the communication diagrams should correspond to the order in which the use cases are executed

Integrated communication diagrams

- How to create?
  - Start with the communication diagram for the **first** use case

    and

    superimpose the communication diagram for the second use case on top of the first to form a integrated diagram

    Next, superimpose the third diagram on top of the integrated diagram of the first two,

    and so on

  - In each case,
    - add new objects and new message interactions from each subsequent diagram onto the integrated diagram
    - gradually gets bigger as more objects and message interactions are added
    - Duplicate objects are only shown once

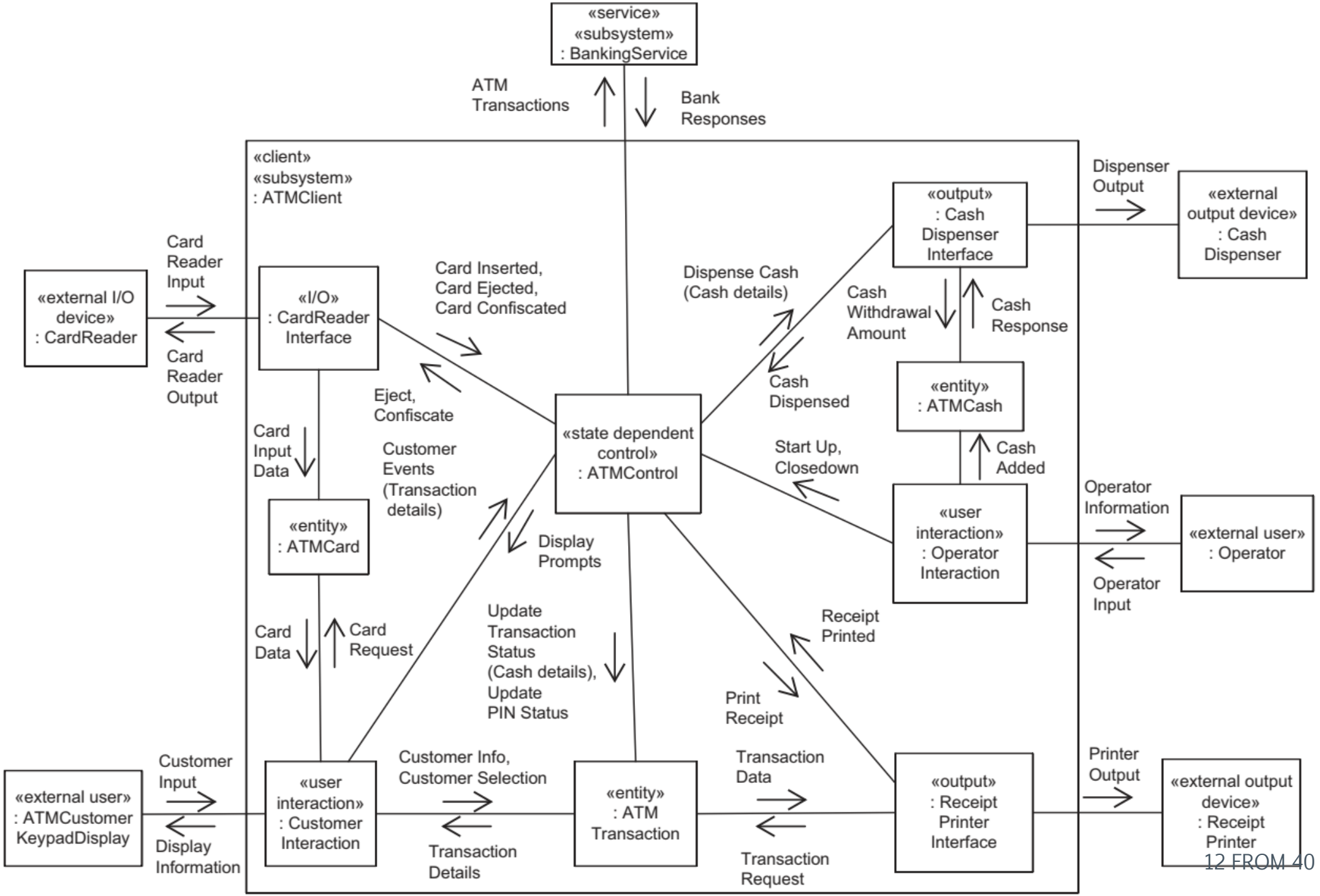
Integrated communication diagrams

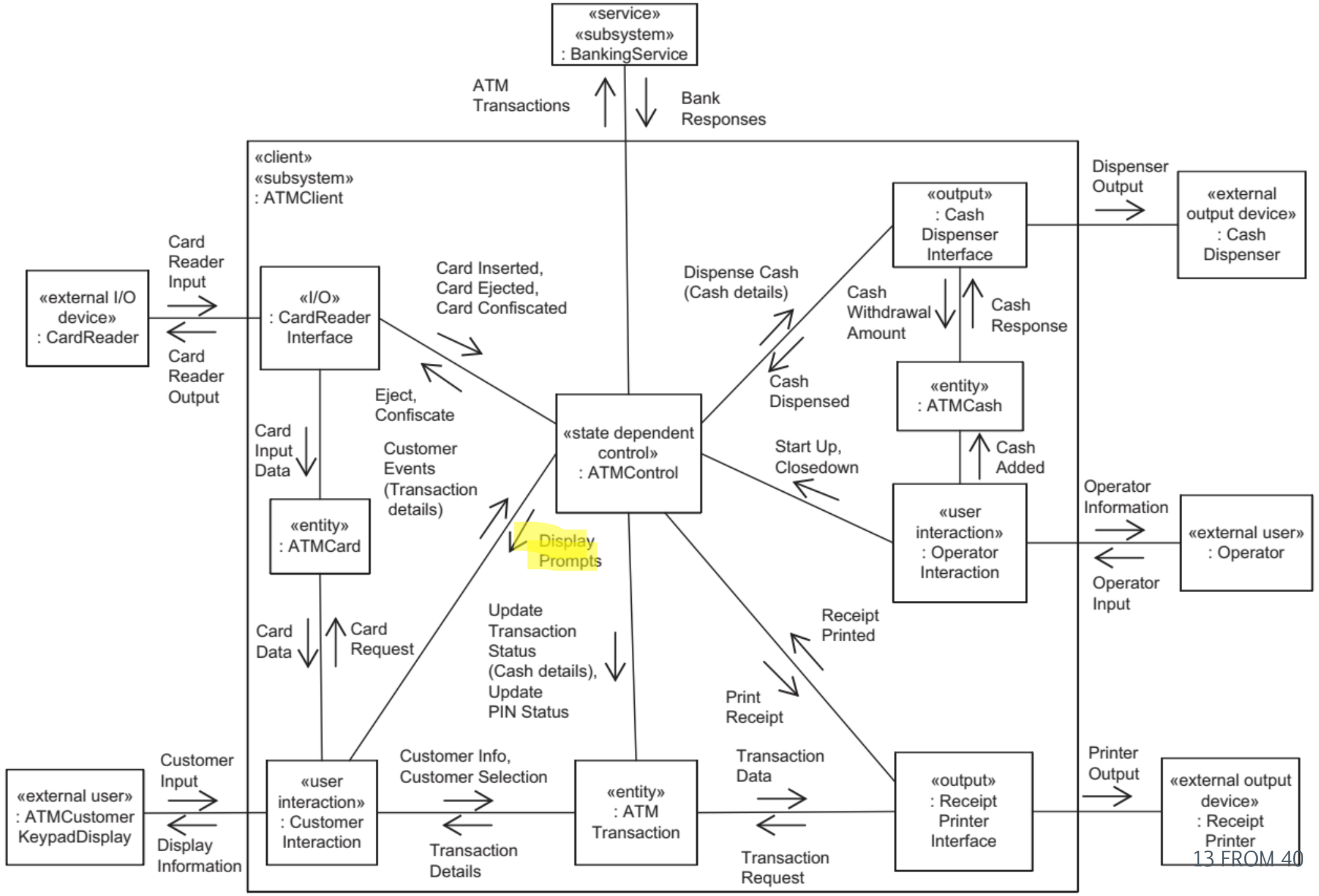
- In the integrated communication diagram,
  - it is necessary to show
  - all message communication derived from the integrated use cases

  - The integrated communication diagram
    - is a synthesis of all relevant use case–based communication diagrams showing all objects and their interactions
    - message sequence numbering does not need to be shown because this would only add clutter

    - represented as a generic UML communication diagram

    - Example: ATM Client subsystem of the Banking System realizing the seven usecases


«service»
«subsystem»
: BankingService
ATM Transactions
Bank Responses
«client»
«subsystem»
: ATMClient
«external I/O device»
: CardReader
Card Reader Input
Card Reader Output
«I/O»
: CardReader Interface
Card Inserted,
Card Ejected,
Card Confiscated
Eject,
Confiscate
Card Input Data
«entity»
: ATMCard
Card Data
Card Request
Customer Events (Transaction details)
«state dependent control»
: ATMControl
Display Prompts
Update Transaction Status (Cash details), Update PIN Status
Dispense Cash (Cash details)
Cash Dispensed
«output»
: Cash Dispenser Interface
Cash Withdrawal Amount
Cash Response
«entity»
: ATMCash
Cash Added
Start Up, Closedown
«user interaction»
: Operator Interaction
Receipt Printed
Print Receipt
Dispenser Output
«external output device»
: Cash Dispenser
Operator Information
Operator Input
«external user»
: Operator
«external user»
: ATMCustomer KeypadDisplay
Customer Input
Display Information
«user interaction»
: Customer Interaction
Customer Info, Customer Selection
Transaction Details
«entity»
: ATM Transaction
Transaction Data
Transaction Request
«output»
: Receipt Printer Interface
Printer Output
«external output device»
: Receipt Printer


«service»
«subsystem»
: BankingService
ATM Transactions
Bank Responses
«client»
«subsystem»
: ATMClient
«external I/O device»
: CardReader
Card Reader Input
Card Reader Output
«I/O»
: CardReader Interface
Card Inserted, Card Ejected, Card Confiscated
Eject, Confiscate
Card Input Data
«entity»
: ATMCard
Card Data
Card Request
Customer Events (Transaction details)
Display Prompts
«state dependent control»
: ATMControl
Update Transaction Status (Cash details), Update PIN Status
Dispense Cash (Cash details)
Cash Dispensed
«output»
: Cash Dispenser Interface
Cash Withdrawal Amount
Cash Response
«entity»
: ATMCash
Cash Added
Start Up, Closedown
«user interaction»
: Operator Interaction
Receipt Printed
Print Receipt
Dispenser Output
«external output device»
: Cash Dispenser
Operator Information
Operator Input
«external user»
: Operator
«external user»
: ATMCustomer KeypadDisplay
Customer Input
Display Information
«user interaction»
: Customer Interaction
Customer Info, Customer Selection
Transaction Details
«entity»
: ATM Transaction
Transaction Data
Transaction Request
«output»
: Receipt Printer Interface
Printer Output
«external output device»
: Receipt Printer

Integrated communication diagrams

- Integrated communication diagrams
  - Can get very complicated
    - it is necessary to have ways to reduce the amount of information
      - aggregating the messages between two object
      - **Furthermore,**
        - showing all the objects on one diagram might not be practical
        - It is better to develop a higher-level subsystem communication diagram to show the interaction between the subsystems (integrated communication diagrams)
        - The dynamic interactions between subsystems can be depicted on a **subsystem communication diagram**,

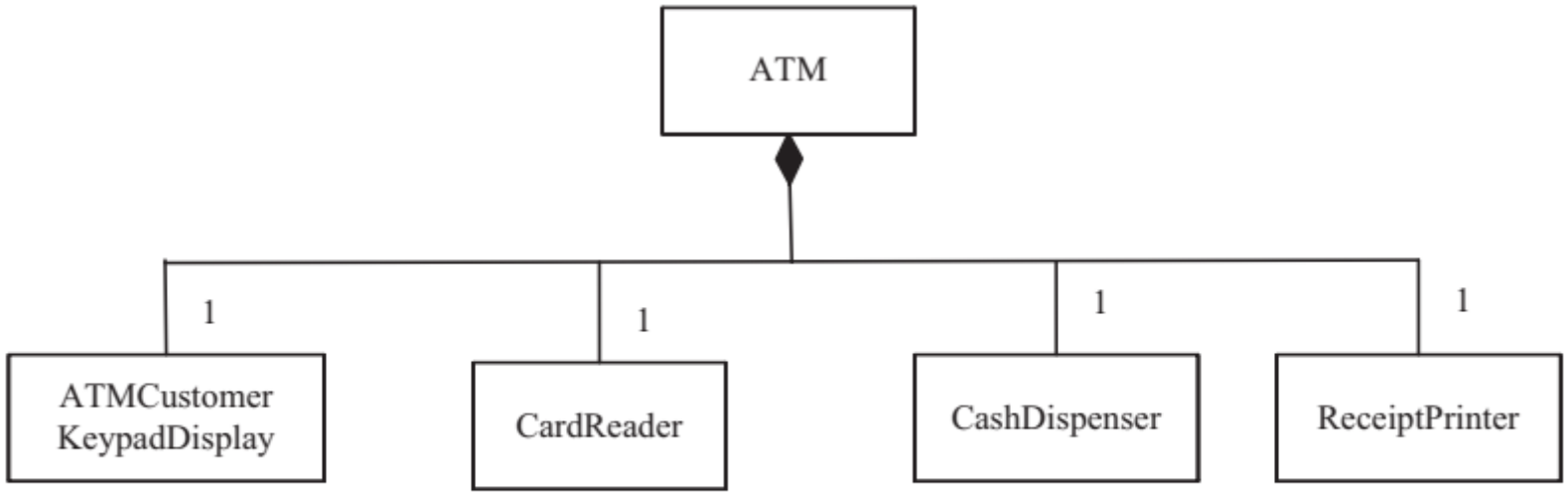
Separation of concepts in subsystem design

- important structuring decisions when designing subsystems
  - The goal is to make subsystems more self-contained
    - so that different concerns are addressed by different subsystems

1. Composite Object
2. Geographical Location
3. Clients and Services
4. User Interaction
5. Interface to External Objects
6. Scope of Control

Separation of concepts in subsystem design

1. Composite Object
   - Objects that are part of the same composite object should be in the same subsystem
   - Composition is stronger than aggregation
     - With composition, the composite object (the whole) and its constituent objects (the parts) are created together, live together, and die together.



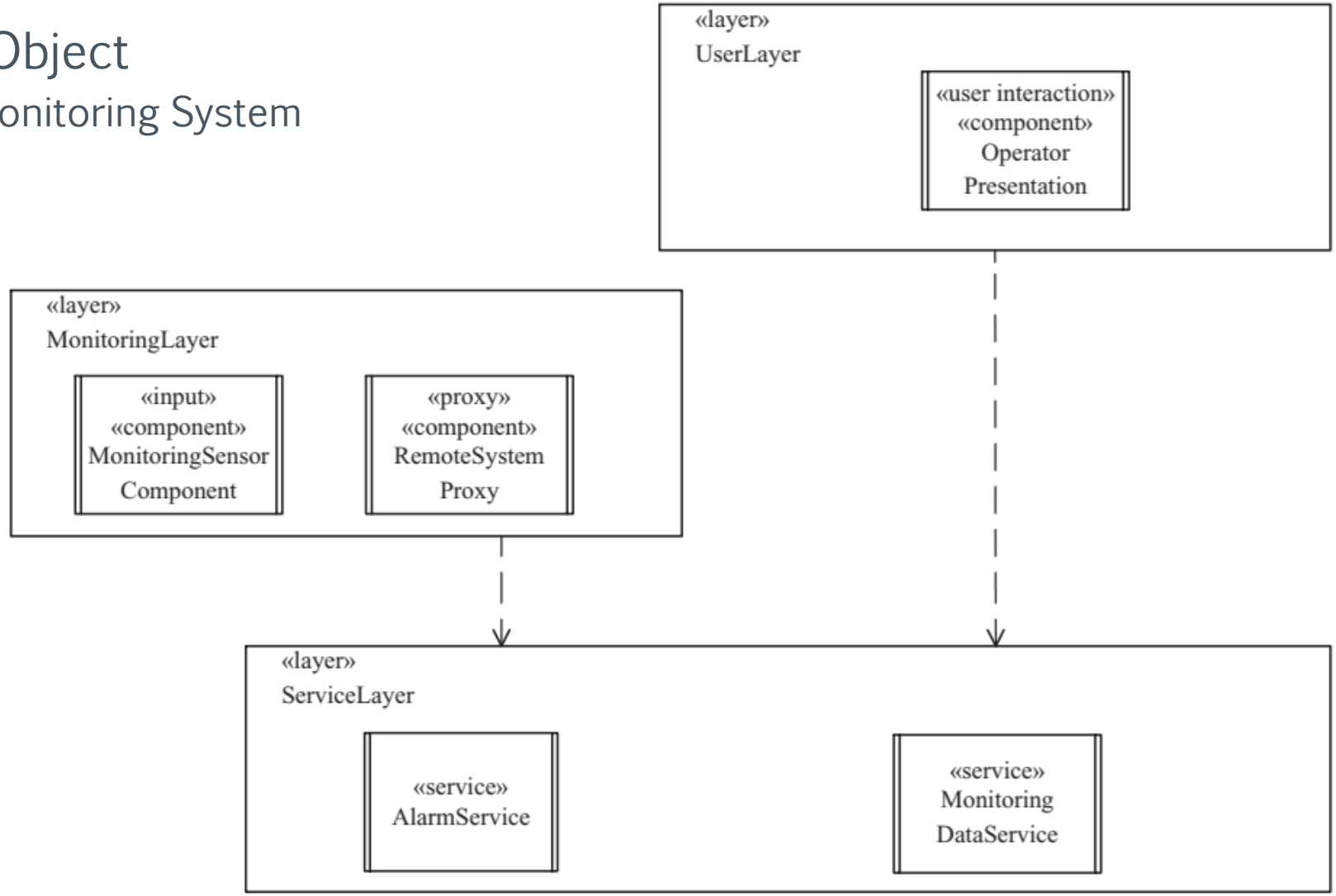
Example of composite class: ATM

Separation of concepts in subsystem design

1. Composite Object
   - A composite object is typically composed of a group of related objects that work together in a coordinated fashion
     - analogous to the assembly structure in manufacturing
   - A subsystem supports information hiding at a higher level of abstraction than an individual object does
   - An **aggregate subsystem**
     - contains objects grouped by functional similarity, which might span geographical boundaries
   - In a software architecture that spans multiple organizations,
     - it can be useful to depict each organization as an aggregate subsystem.

Separation of concepts in subsystem design

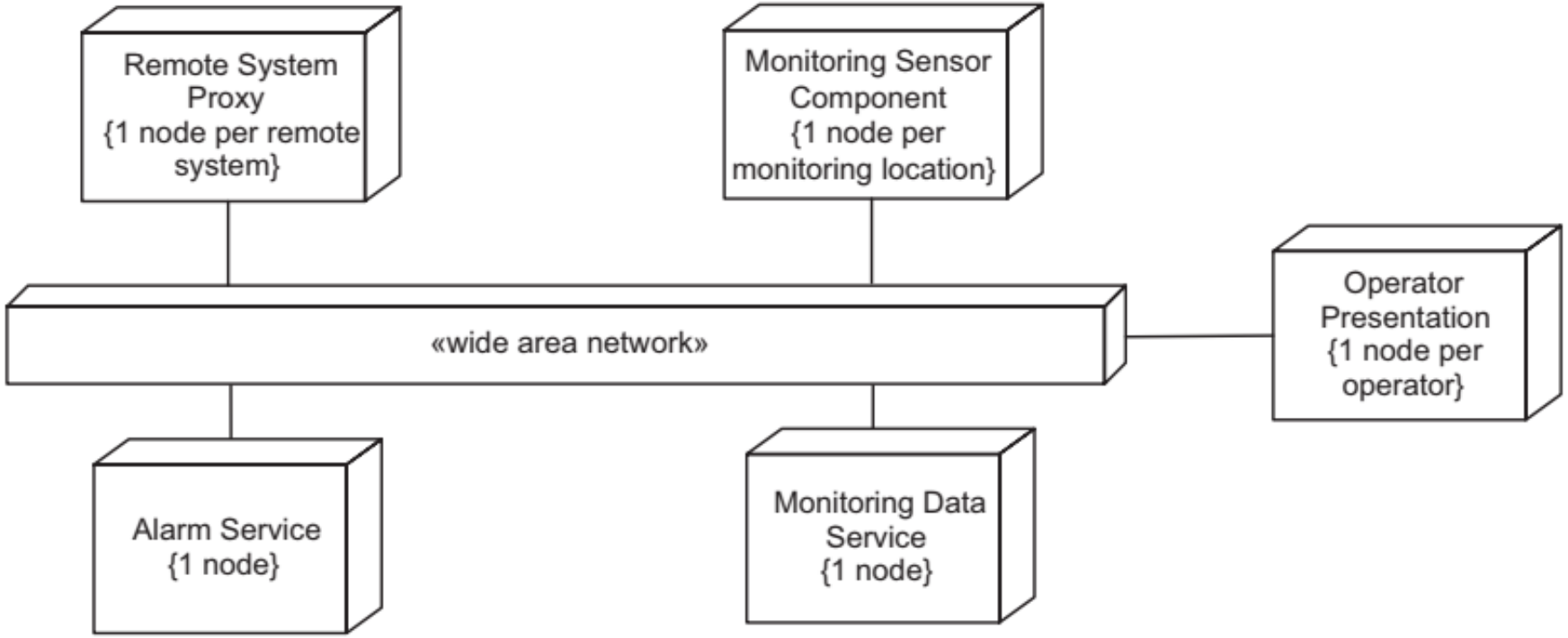
1. Composite Object
   - Emergency Monitoring System



Layered architecture with aggregate and composite subsystems: Monitoring System

Separation of concepts in subsystem design

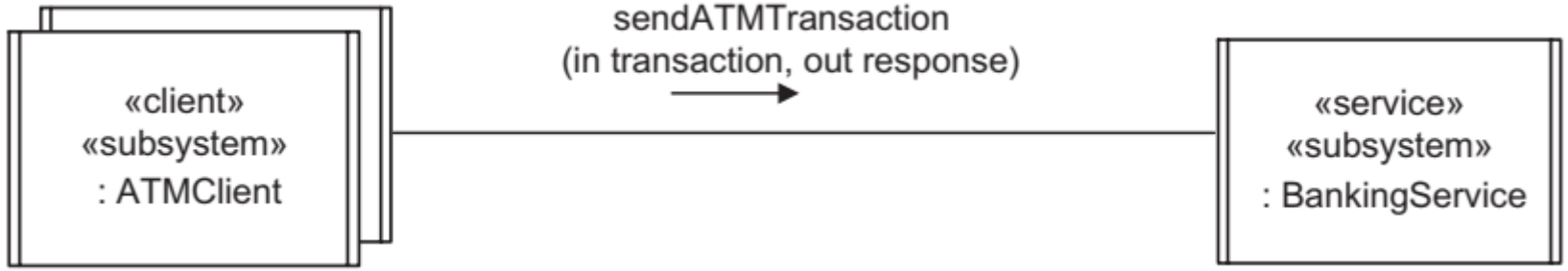
2. Geographical Location



Separation of concepts in subsystem design

3. Clients and Services

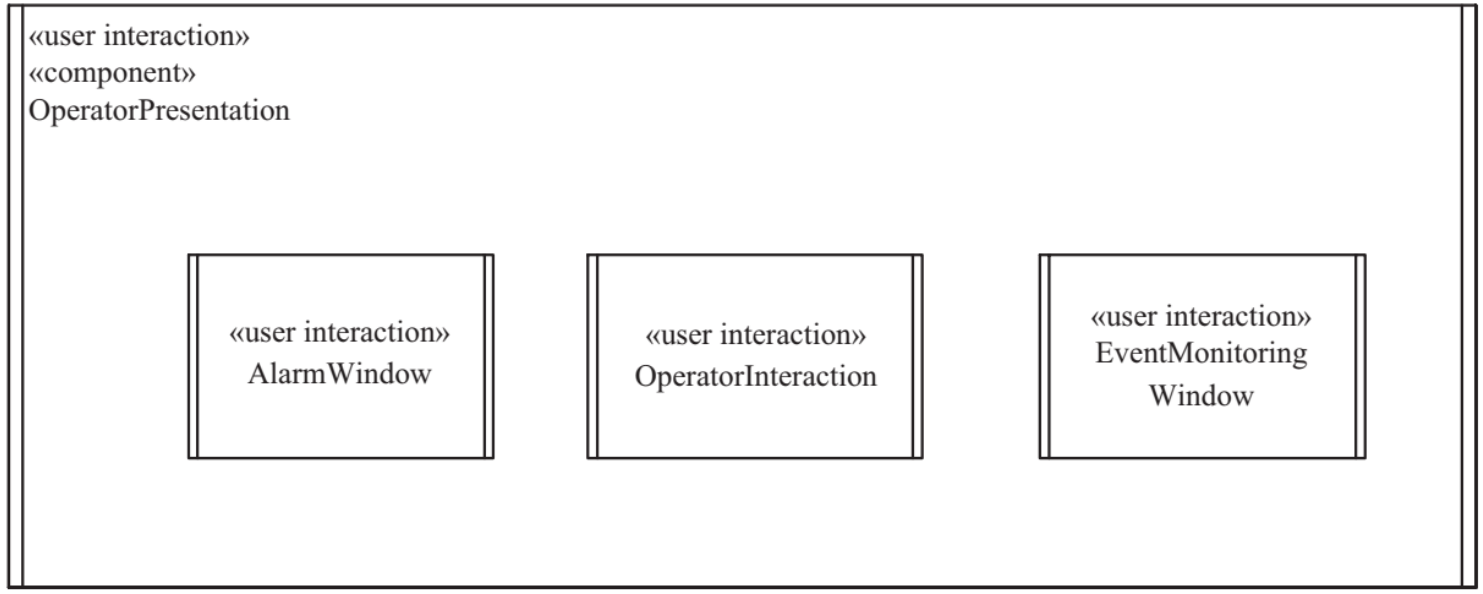
- Clients and services should be in separate subsystems.
- This guideline can be viewed as a special case of the geographical location rule
  - because clients and services are usually at different locations.



Separation of concepts in subsystem design

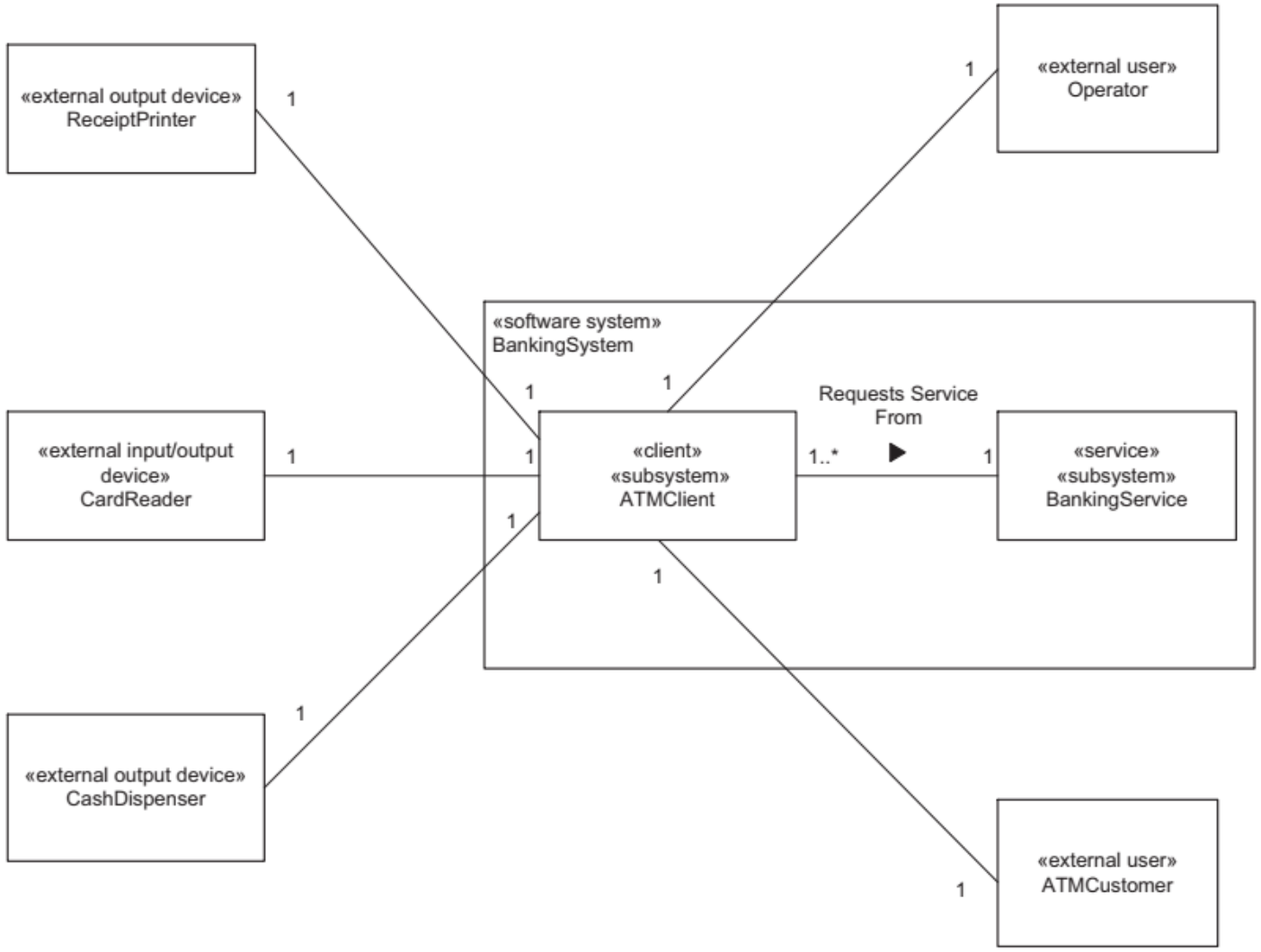
4. User Interaction

   - Users often use their own PCs as part of a larger distributed configuration,
     - so the most flexible option is to keep user interaction objects in separate subsystems
     - Because user interaction objects are usually clients,
       - this guideline can be viewed as a special case of the client/service guideline



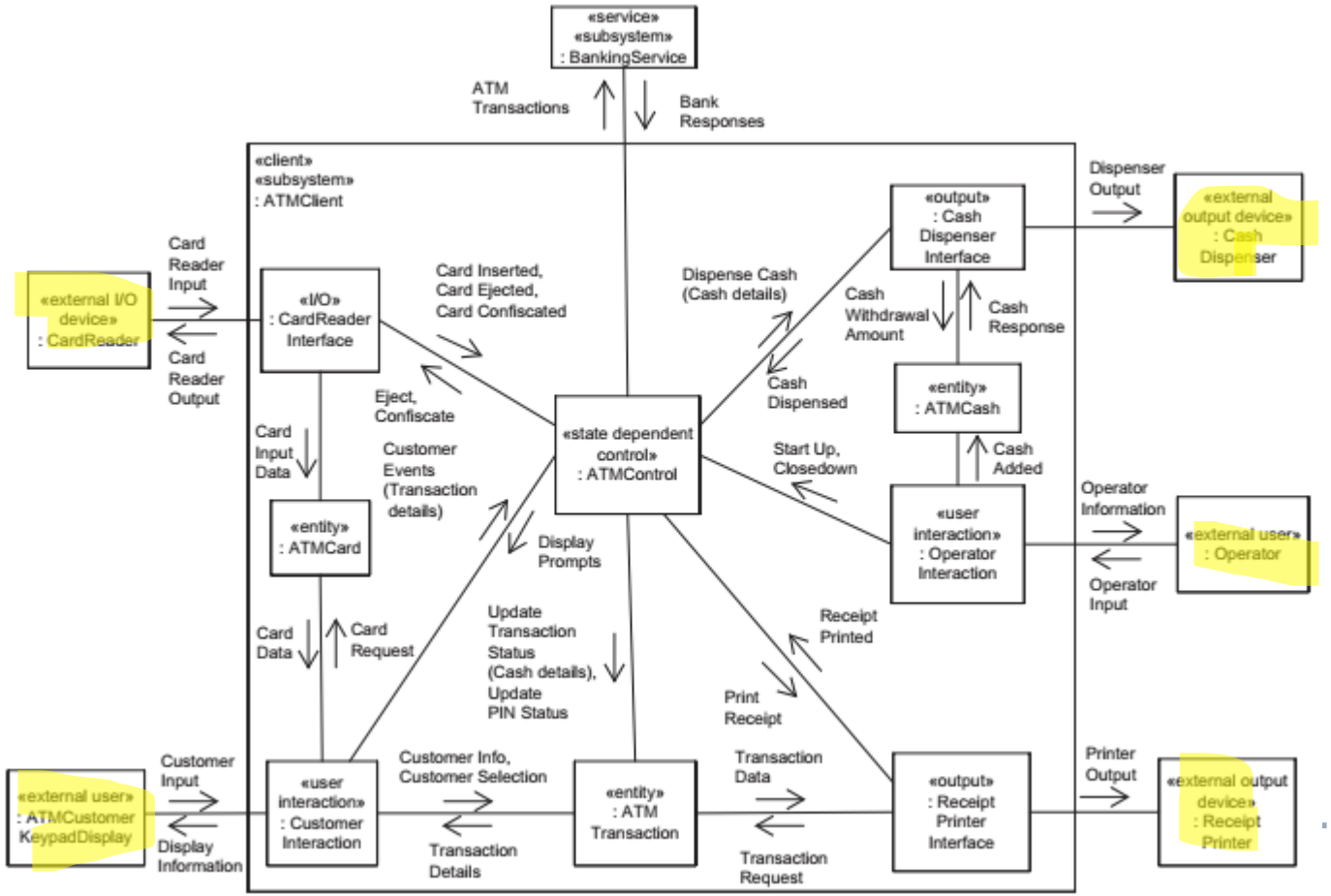
Separation of concepts in subsystem design

5. Interface to External Objects



Separation of concepts in subsystem design

6. Scope of Control
   - A control object and all the entity and I/O objects it directly controls
     - should all be part of one subsystem and not split among subsystems



Subsystem Structuring Criteria

- Design considerations

- help ensure that subsystems are designed effectively
- A subsystem can satisfy more than one of the structuring criteria

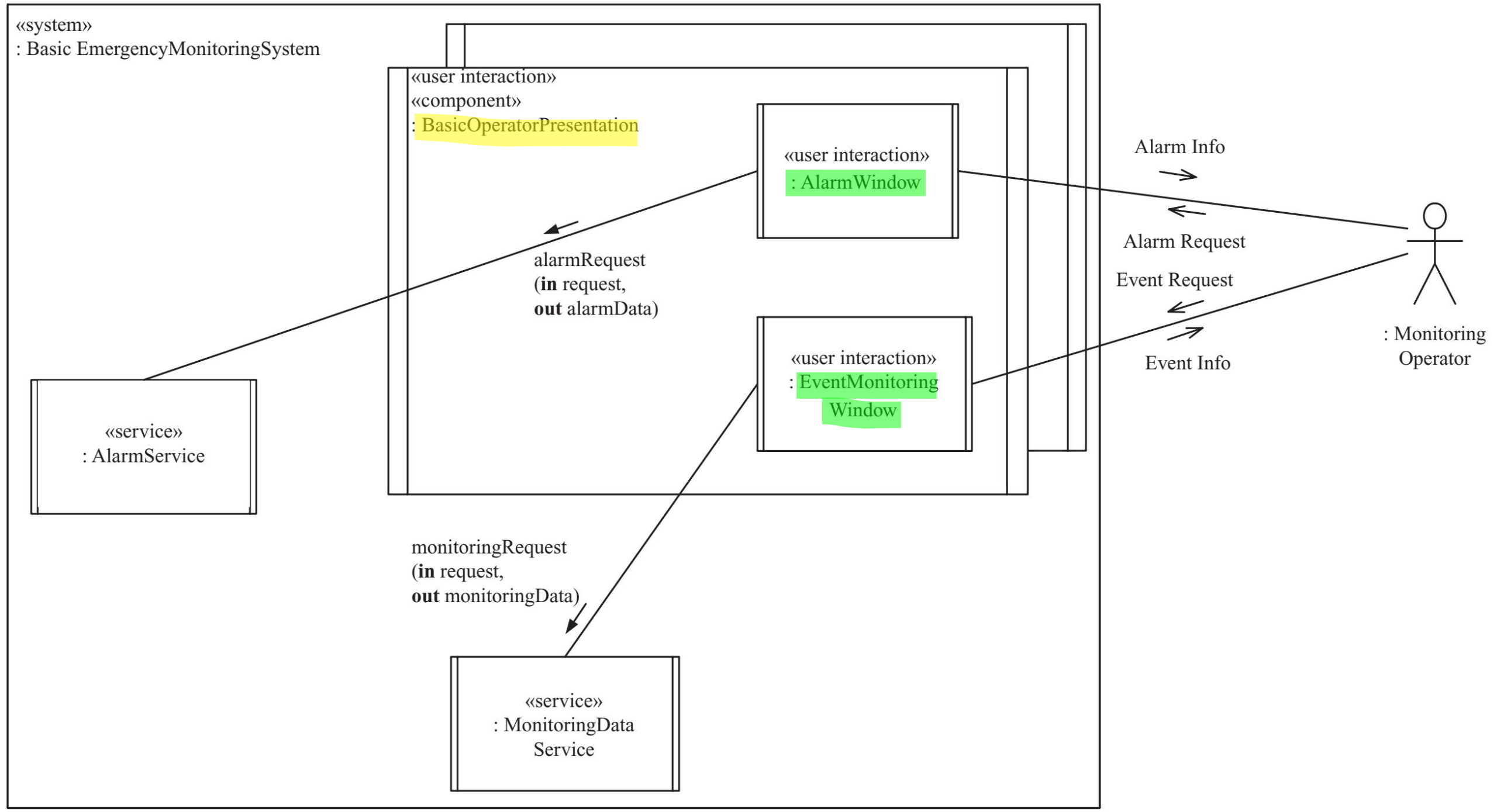
- Depicted Subsystem stereotypes
  - General stereotype: «subsystem»
  - Stereotype «component» is also used for distributed component-based systems
  - Stereotype «service» is also used for service-oriented architectures

Subsystem Structuring Criteria

- Client Subsystem
  - Client: **requester of one or more services**
    - Wholly dependent vs. partially dependent on a given service
  - Client subsystems include:
    1. user interaction subsystems
    2. control subsystems
    3. I/O subsystems

Subsystem Structuring Criteria

- Client Subsystem
  - user interaction subsystems
    - providing user access to services
    - There may be more than one user interaction subsystem
      - one for each type of user
    - is usually a composite object that is composed of several simpler user interaction objects
    - might run on a separate node and interact with subsystems on other nodes
    - could support a simple user interface,
      - consisting of a command line interface or a graphical user interface that contains multiple objects.
    - Or involve multiple windows and multiple threads of control

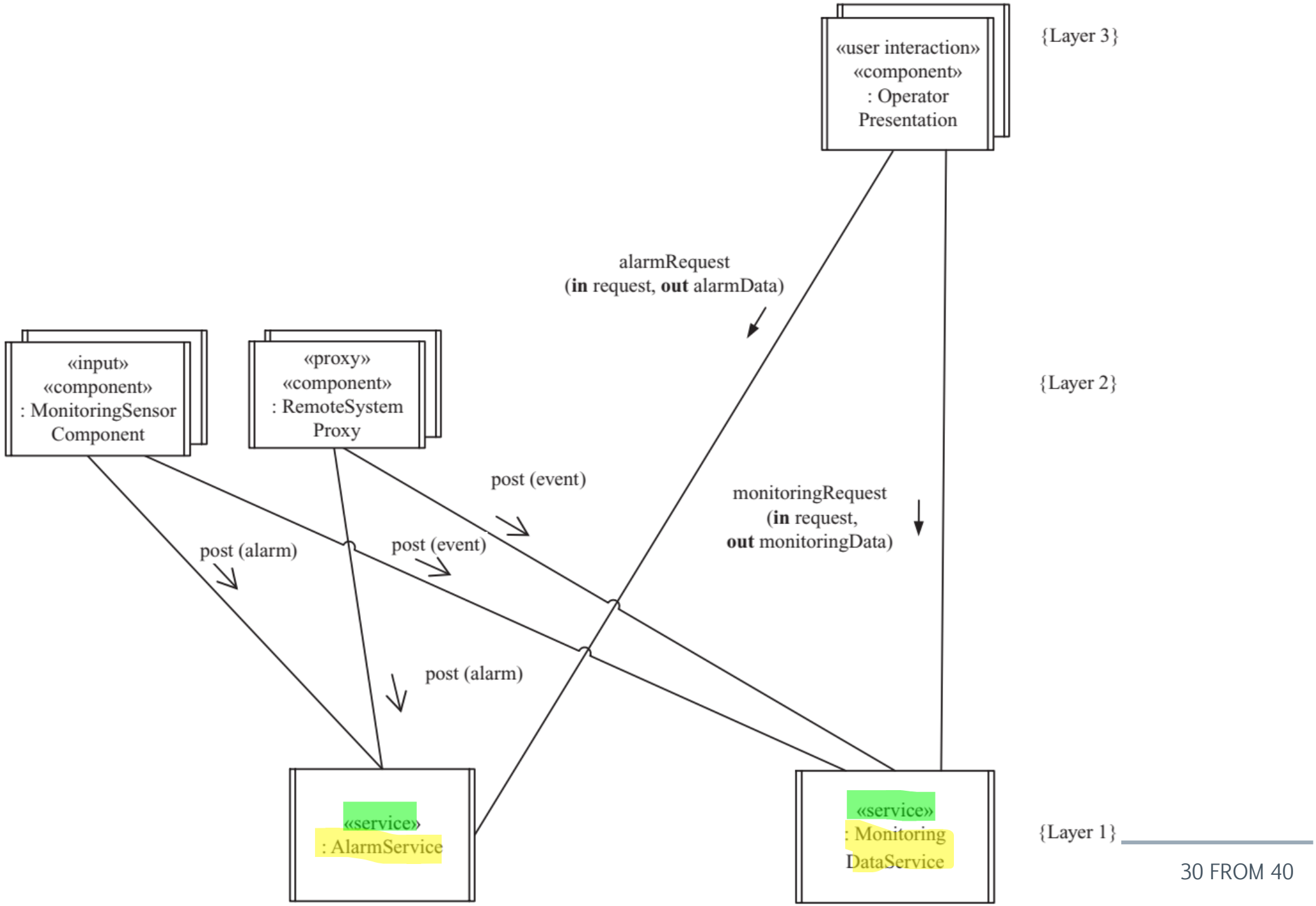

«system»
: Basic EmergencyMonitoringSystem
«user interaction»
«component»
: BasicOperatorPresentation
«user interaction»
: AlarmWindow
«user interaction»
: EventMonitoring
Window
alarmRequest
(in request,
out alarmData)
monitoringRequest
(in request,
out monitoringData)
«service»
: AlarmService
«service»
: MonitoringData
Service
Alarm Info
Alarm Request
Event Request
Event Info
: Monitoring
Operator

Subsystem Structuring Criteria

- Service subsystems
  - provides service
    - servicing client requests
  - does not initiate any requests
  - are usually composite objects
    - that are composed of two or more objects
    - include
      - entity objects
      - coordinator objects that service client requests and determine what object should be assigned to handle them
      - and business logic objects that encapsulate application logic.
    - Frequently
      - is associated with a data repository or a set of related data repositories,
      - or it might provide access to a database

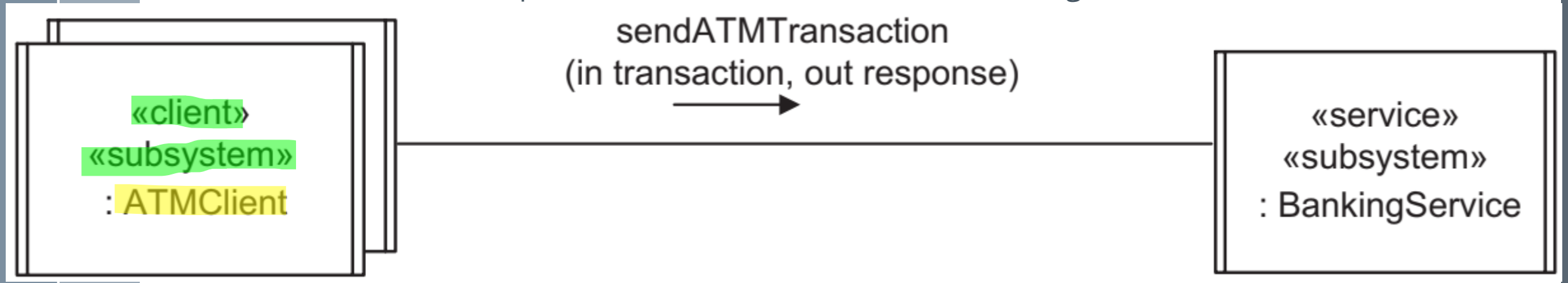
Subsystem Structuring Criteria

- Service subsystems
  - might be associated with an I/O device or a set of related I/O devices
    - such as a file service or line printer service
    - A data service supports remote access to a centralized database or file store.
    - An I/O service processes requests for a physical resource that resides at that node.


«user interaction»
«component»
: Operator
Presentation
{Layer 3}
alarmRequest
(in request, out alarmData)
«input»
«component»
: MonitoringSensor
Component
«proxy»
«component»
: RemoteSystem
Proxy
{Layer 2}
post (event)
monitoringRequest
(in request,
out monitoringData)
post (alarm)
post (event)
post (alarm)
«service»
: AlarmService
«service»
: Monitoring
DataService
{Layer 1}

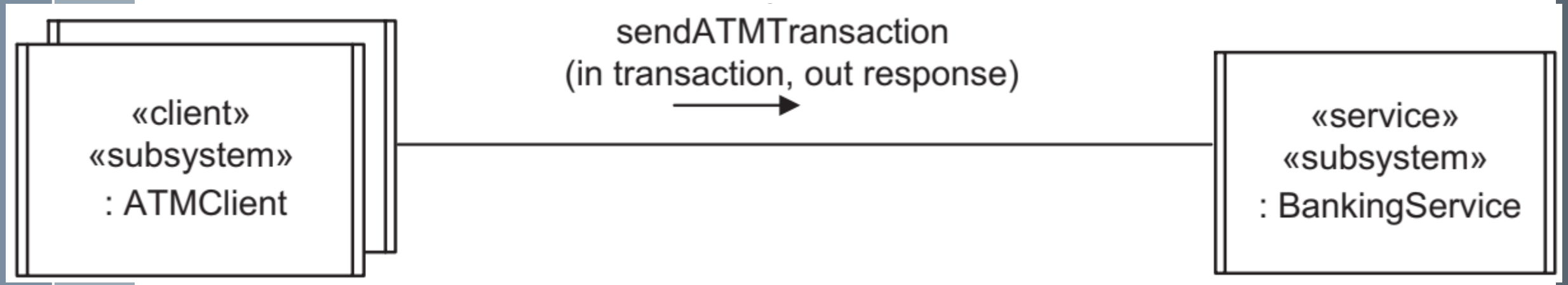
Subsystem Structuring Criteria

- Client Subsystem
  - Control subsystems
    - controls a given part of the system
    - receives its inputs from the external environment and generates outputs to the external environment,
      - usually without any human intervention
    - is often state-dependent
    - It might receive some high-level commands from another subsystem that gives it overall direction,
    - after which it provides the lower-level control, sending status information to other



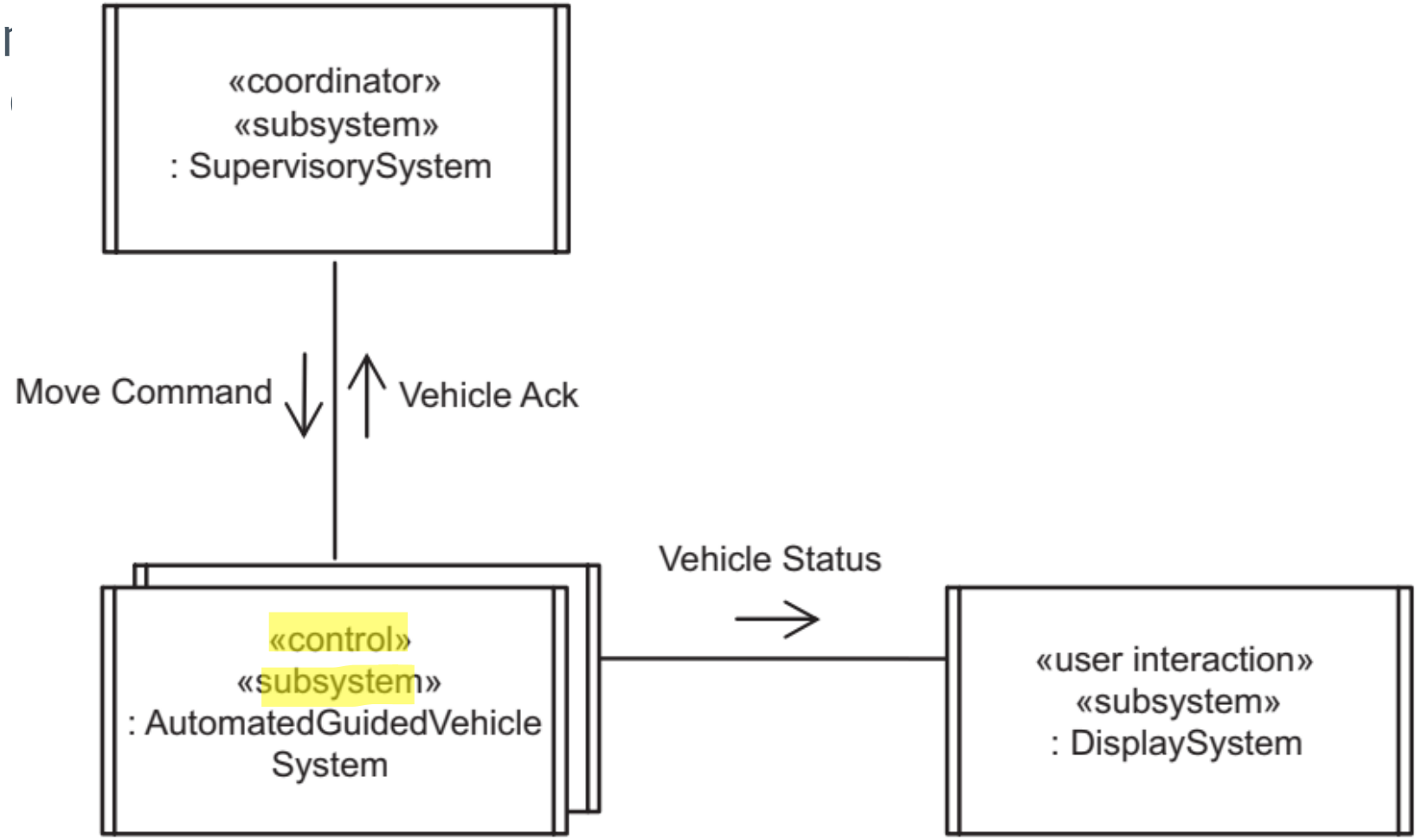
Subsystem Structuring Criteria

- Client Subsystem
  - Control subsystems
  - Example
    - ATM Client subsystem
    - which combines the roles of control and user interaction
    - There are multiple instances of the ATM Client,
      - one for each ATM
    - each instance is independent of the others and only communicates with the Banking Service subsystem



Subsystem Structuring Criteria

- Client Subsystem
  - Control subsystems
  - Another Exar
    - Automated

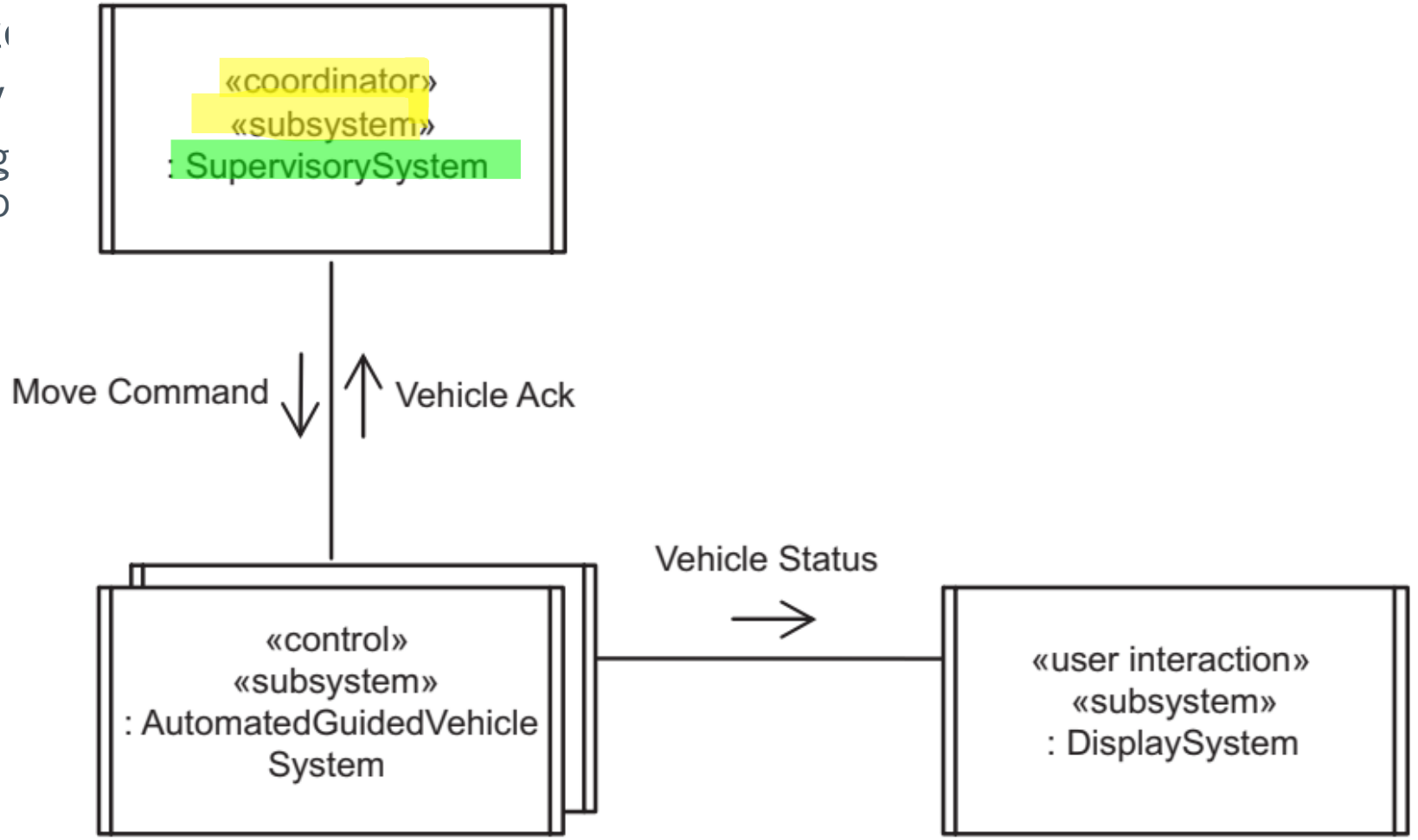


Subsystem Structuring Criteria

- Coordinator Subsystem
  - coordinate the execution of other subsystems
    - such as control subsystems or service subsystems
    - In software architectures with multiple control subsystems, it is sometimes necessary to have a coordinator subsystem
    - If the multiple control subsystems are completely independent of each other, no coordination is required
      - as with the ATM Clients
    - For example,
      - the coordinator subsystem might decide what item of work a control subsystem should do next.

Subsystem Structuring Criteria

- Coordinator Subsystem
  - Example
    - Factory Aut
    - Supervisory
      - that assig
        to a facto



Sul

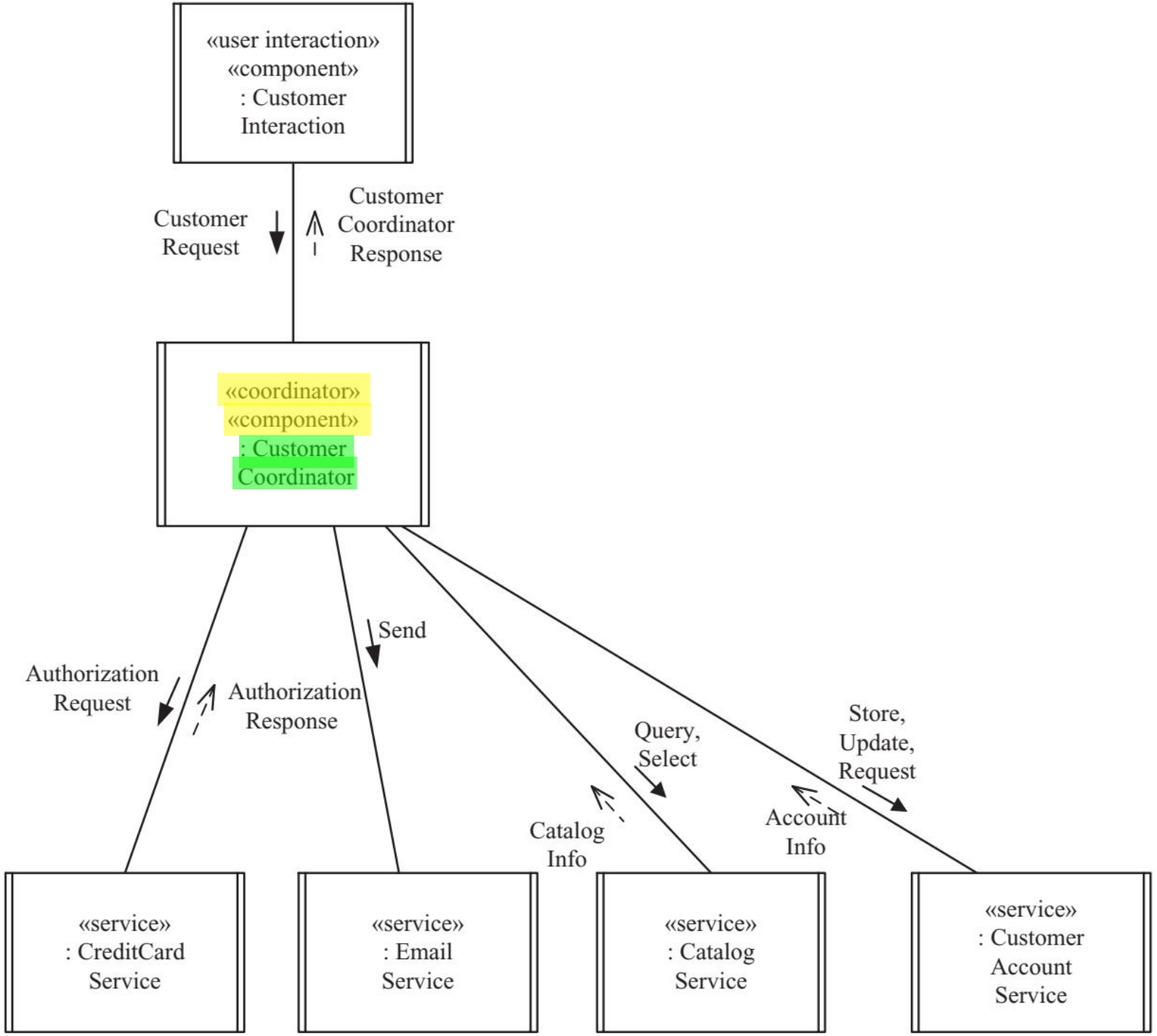
› Cc

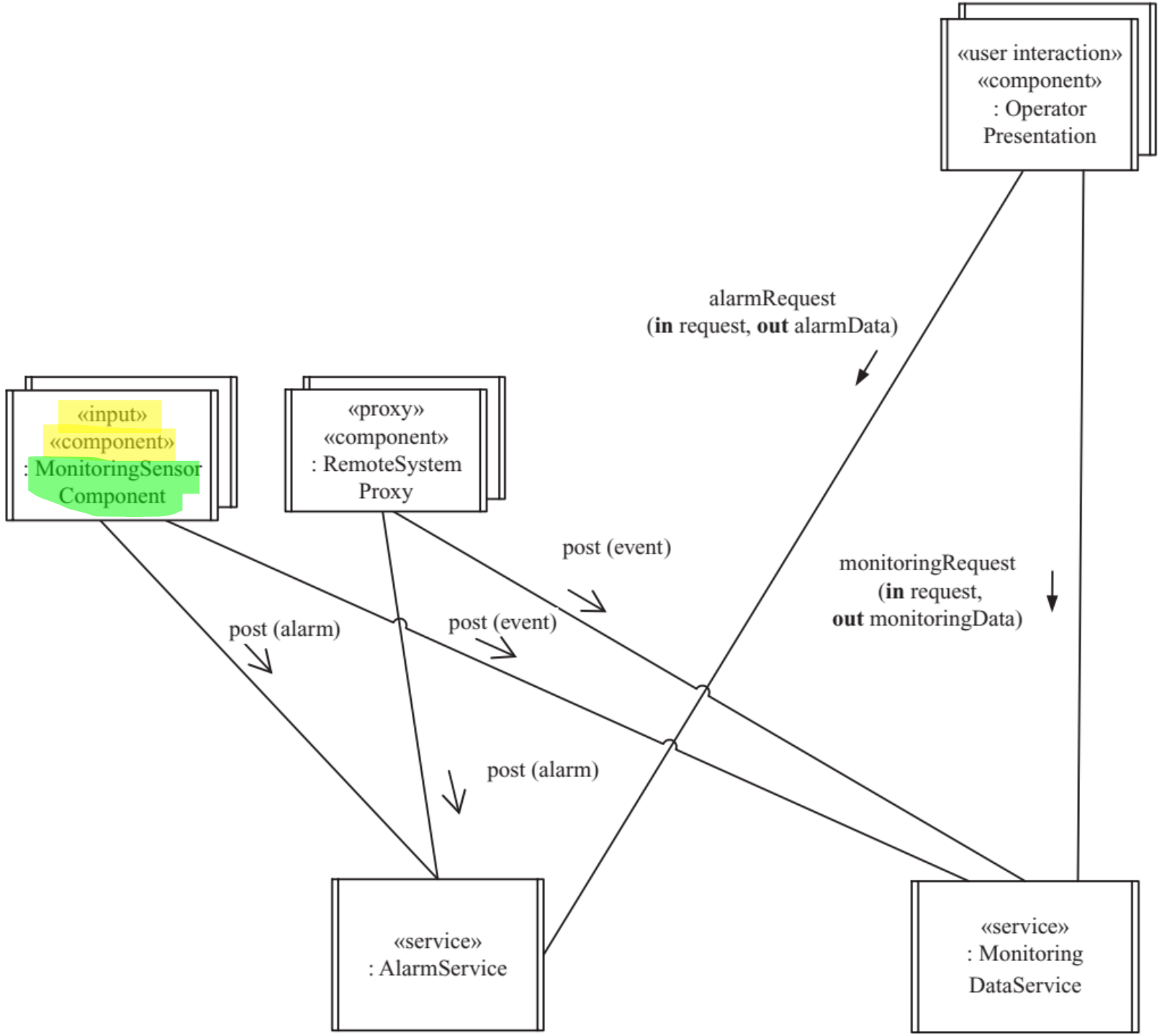
–

ctures

mponent

e,





osystems

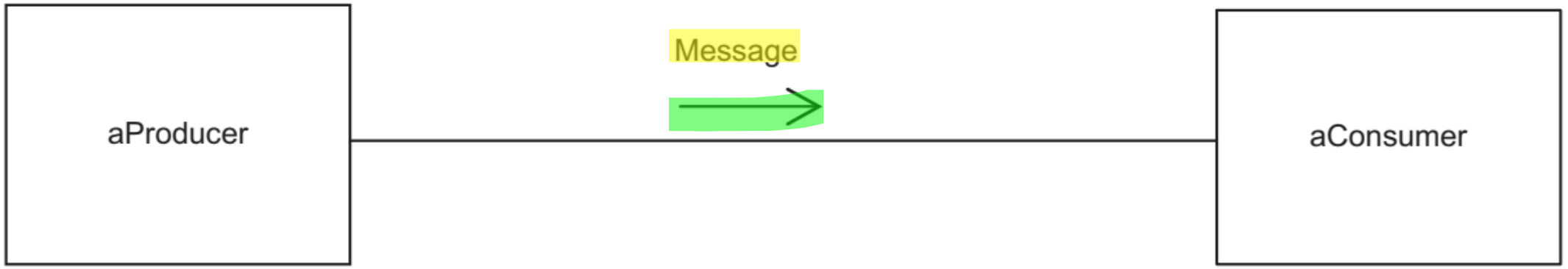
controls

stem

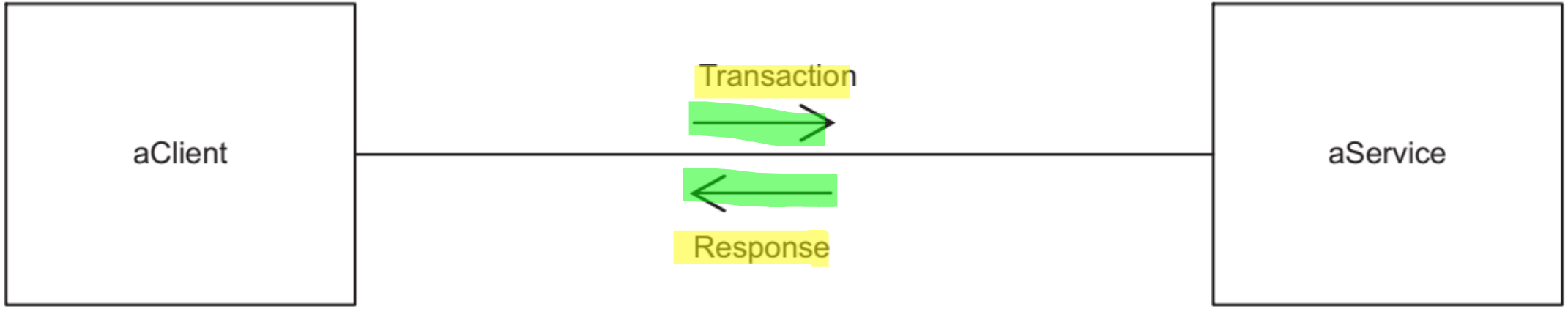
d events to

Decision about message communication

(1) Unidirectional message communication between producer and consumer

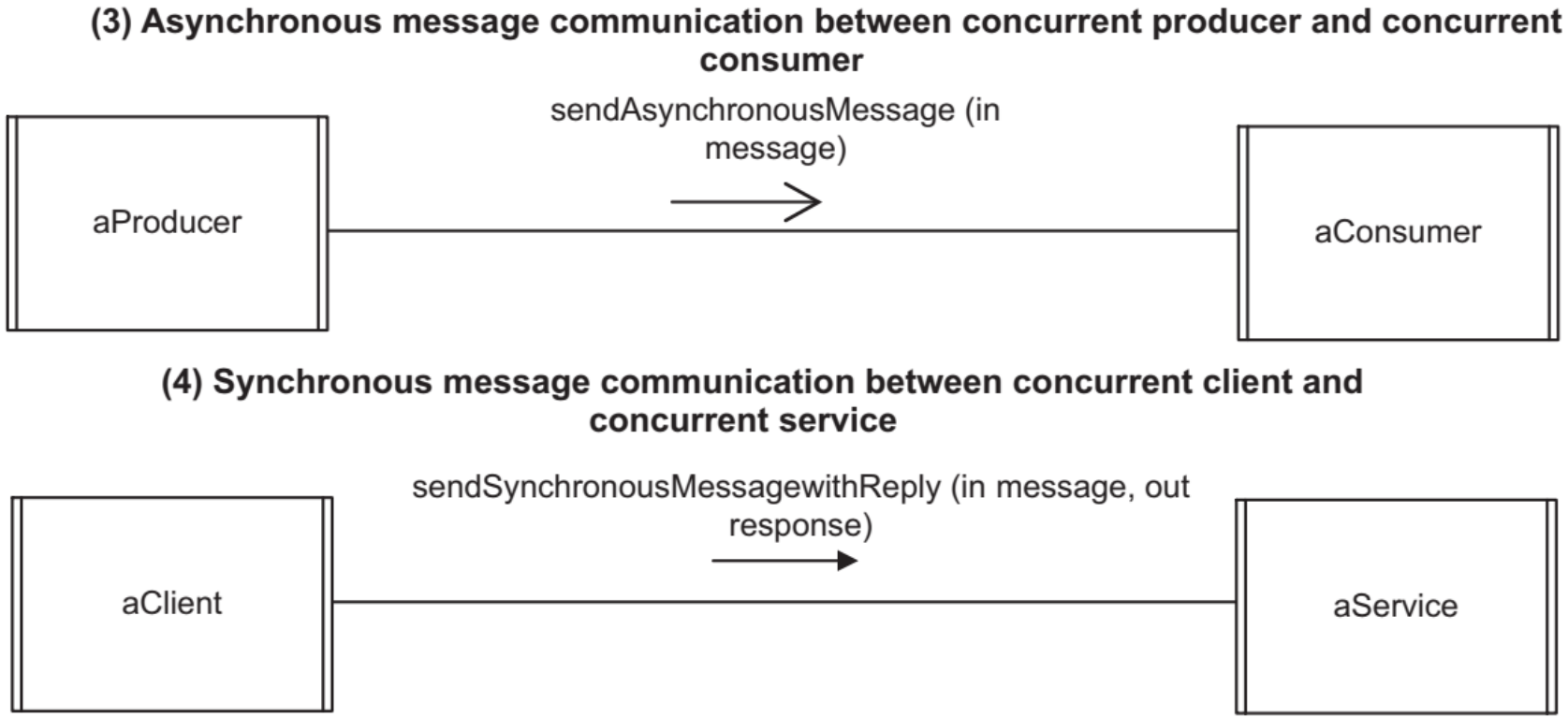


(2) Bidirectional message communication between client and service



Decision about message communication

- Message passing between subsystems
  - In design model
    - Decision for details such as type of message communication



Question?

Bioinformation.ir

info@Bioinformation.ir