Object-Oriented Software Engineering
Using UML, Patterns, and Java

Chapter 7
Dynamic Modeling
Sequence Diagram

(Textbook Chapter 5)

Course instructor : TA.Nada Alamoudi

Dynamic Modeling with UML

- **Definition of a dynamic model:**
  - Describes the components of the system that have interesting dynamic behavior.

- Two UML diagrams types for dynamic modeling:
  - Interaction diagrams describe the dynamic behavior *between* objects.
  - State chart diagrams describe the dynamic behavior *of a single object.*

UML Interaction Diagrams

- Two types of interaction diagrams:
  - Sequence Diagram:
    - Describes the dynamic behavior of several objects over time
    - Good for real-time specifications
  - Collaboration Diagram:
    - Shows the temporal relationship among objects
    - Position of objects is based on the position of the classes in the UML class diagram.
    - Does not show time.

UML State Chart Diagram

- Two types of State Chart diagrams:
  - State Chart Diagram:
    - A state machine that describes the response of an object of a given class to the receipt of outside stimuli (Events).
  - Activity Diagram:
    - A special type of state chart diagram, where all states are action states (Moore Automaton).

Dynamic Modeling

- Definition of a dynamic model:
  - Describes the components of the system that have interesting dynamic behavior
- The dynamic model is described with
  - State diagrams: One state diagram for each class with interesting dynamic behavior
    - Classes without interesting dynamic behavior are not modeled with state diagrams
  - Sequence diagrams: For the interaction between classes
- Purpose:
  - Detect and supply operations for the object model.

How do we detect Operations?

- We look for objects, who are interacting and extract their "protocol"
- We look for objects, who have interesting behavior on their own
- Good starting point: Flow of events in a use case description
- From the flow of events we proceed to the sequence diagram to find the participating objects.

What is an Event?

- Something that happens at a point in time
- An event sends information from one object to another
- Events can have associations with each other:
  - Causally related:
    - An event happens always before another event
    - An event happens always after another event
  - Causally unrelated:
    - Events that happen concurrently

Sequence Diagram

- A sequence diagram is a graphical description of the objects participating in a use case
- Heuristic for finding participating objects:
  - A event always has a sender and a receiver
  - Find them for each event => These are the objects participating in the use case.

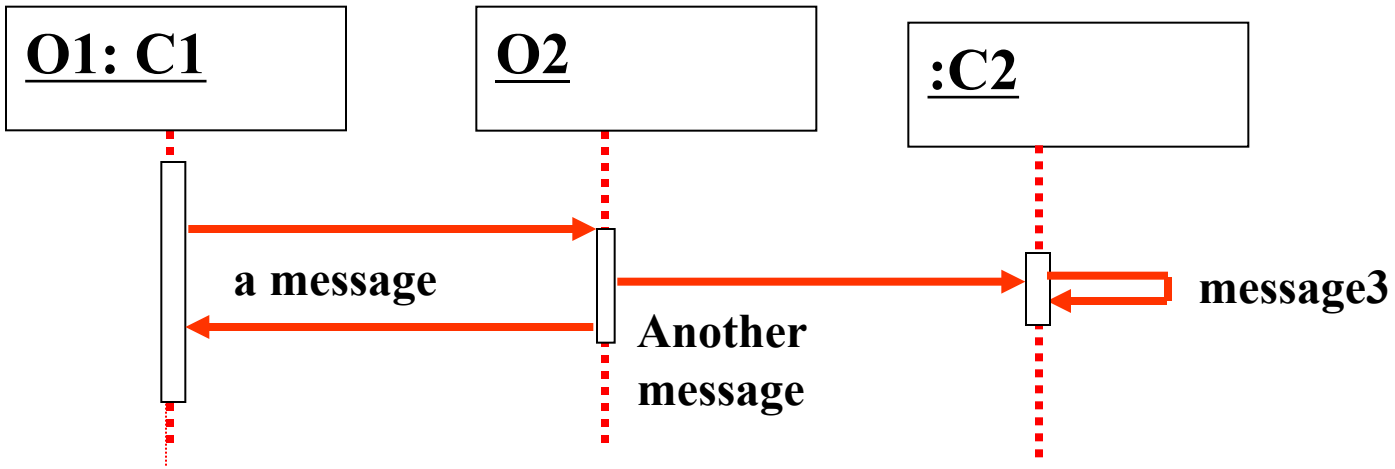
Sequence Diagram Properties

- UML sequence diagram represent *behavior in terms of interactions*
- Useful to identify or find missing objects
- Time consuming to build, but worth the investment
- Complement the class diagrams (which represent structure).

Semantic

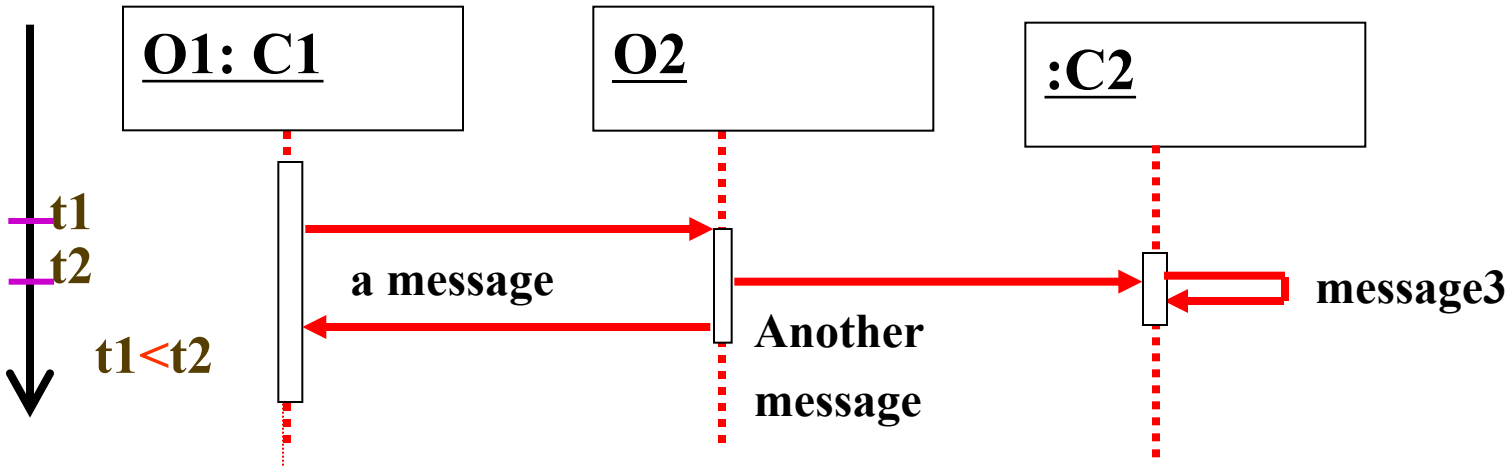
O1	Object O1	Ali
:C	an anonymous object from classe C	:PERSON
/R:C	an anonymous object from classe C playing the role R	/Reader:PERSON
/R	an anonymous object playing the role R	/Reader
O/R:C	An object O from classe C playing the role R	Ali/Reader:PERSON

General representation



- The object which begins the interaction takes place in the left
- Each object is represented by a rectangle and a vertical bar "life line"
- The horizontal organization of objects has no particular meaning:
  - No consequence for semantics of the diagram.

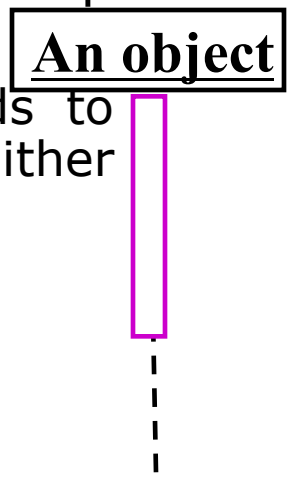
General representation



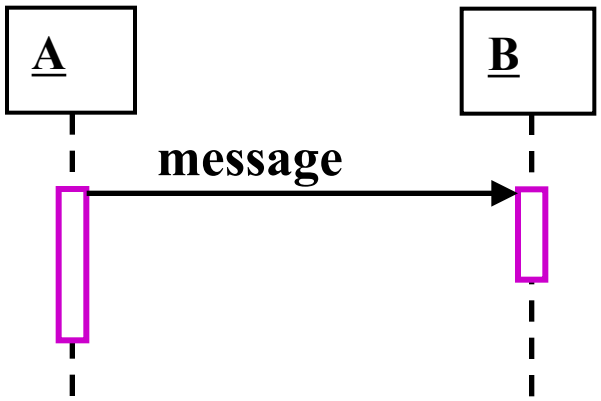
- The order of sending messages is given by their position on the lifelines of objects (on the vertical axis of the diagram):
  - Time flows "from top to bottom" of this axis.
- The vertical dimension represents the flow of time.
  - It can be graduated in order to express temporal constraints.

General representation

- It is possible to represent explicitly the different periods of activity of an object by means of a rectangular strip superposed on the object life line.
  - A period of activity/execution occurrence corresponds to the time during which an object performs an action, either directly or through another object.

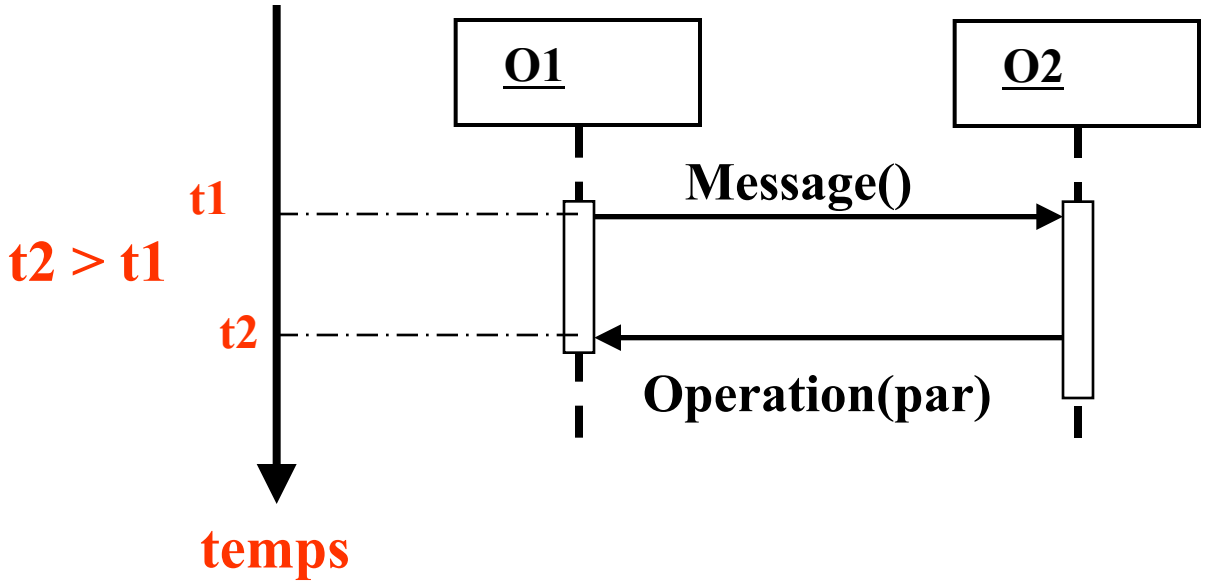


- **Example of an object that activates another:**
  - ✓ **The activity period of the object A overlaps that of B**



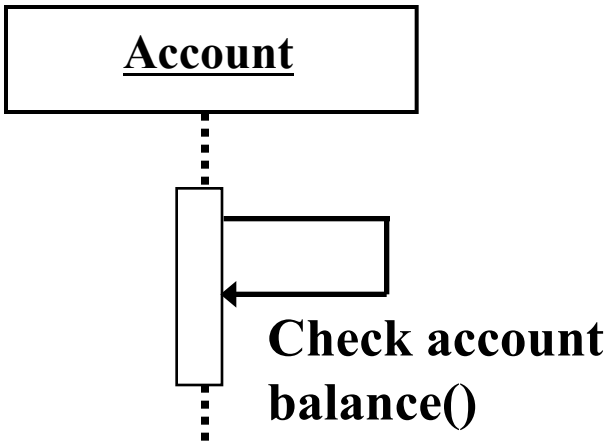
Activation and sending of messages

- The numbering of messages is **optional**. It is replaced by the order of the messages (vertical line from top to bottom).



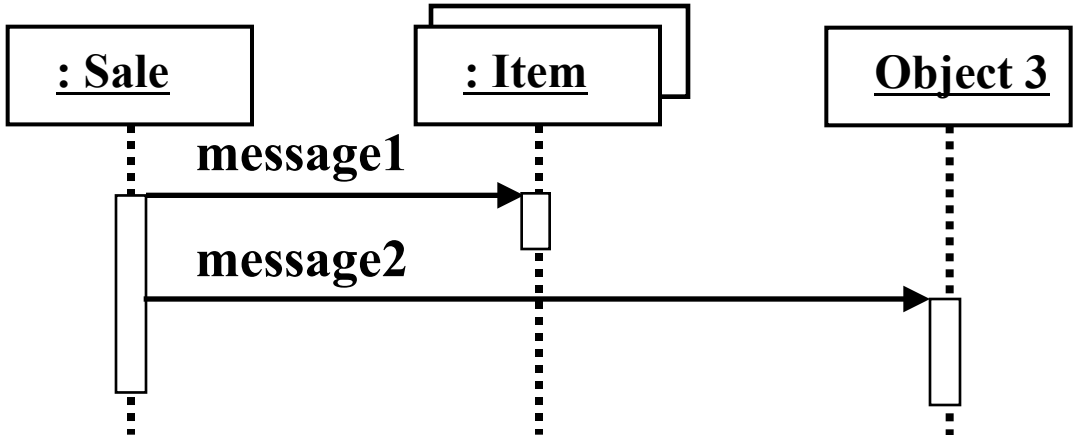
Activation and sending of messages

- A message may be reflexive:



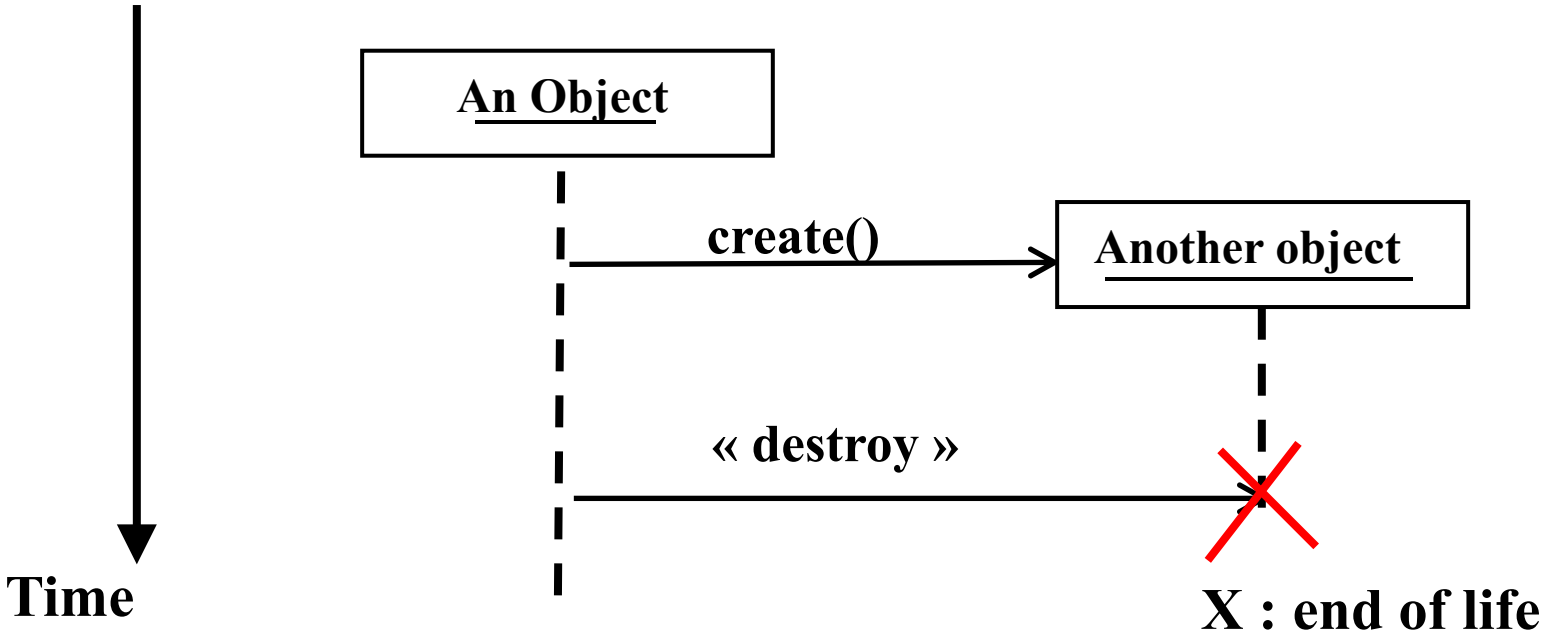
Messages for a collection (multi-object)

- To send a message to all the elements of a collection (list) consists in sending a message to each of them.



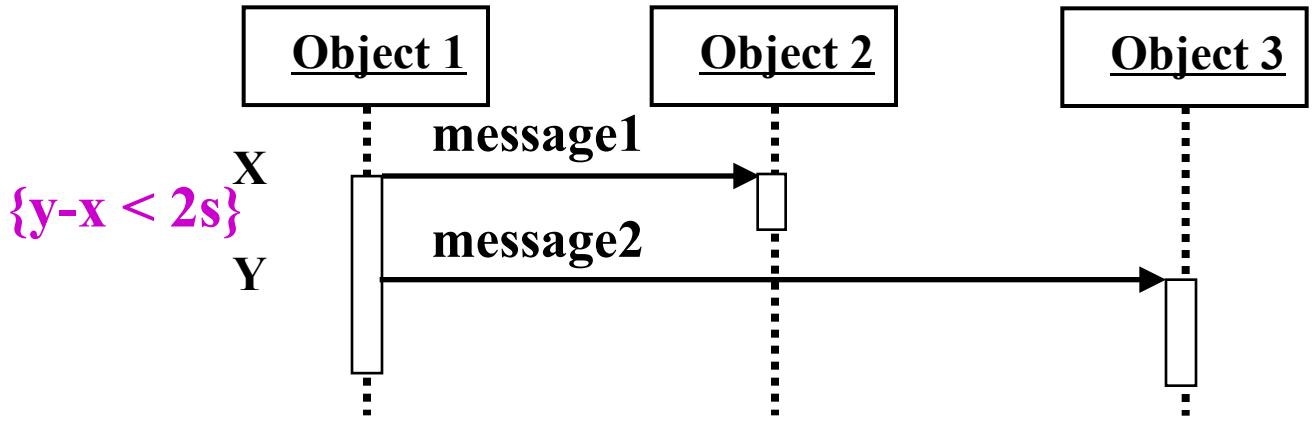
The life line of objects

- **The object life line:**
  - is represented by a dotted line.
  - Can begin and break off in a diagram of sequence if the object is created and/or destroyed during the duration defined by the specified diagram.



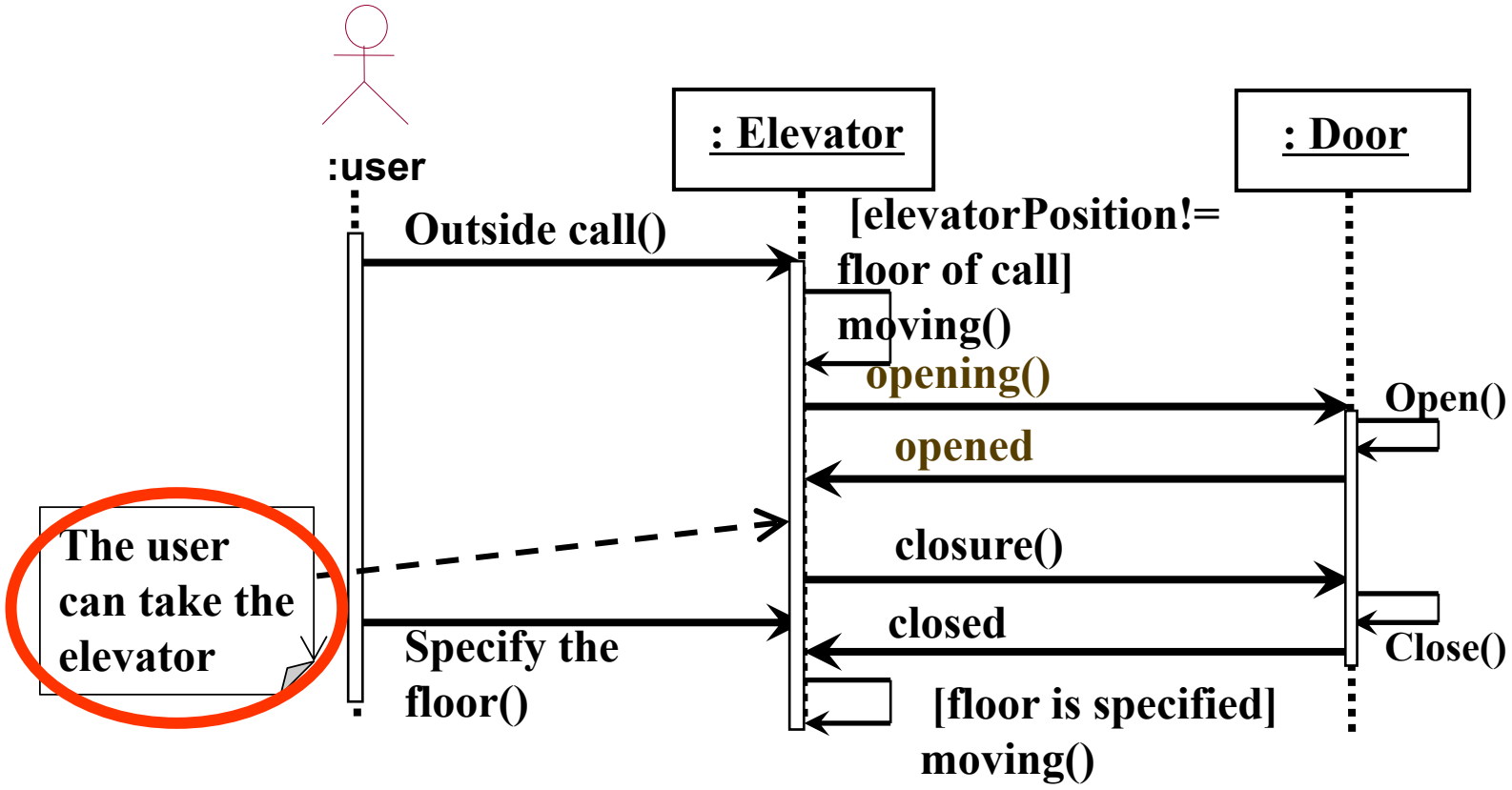
Temporal constraints

- Temporal constraints may be represented in the margin by appointing the moment of messages transmission .

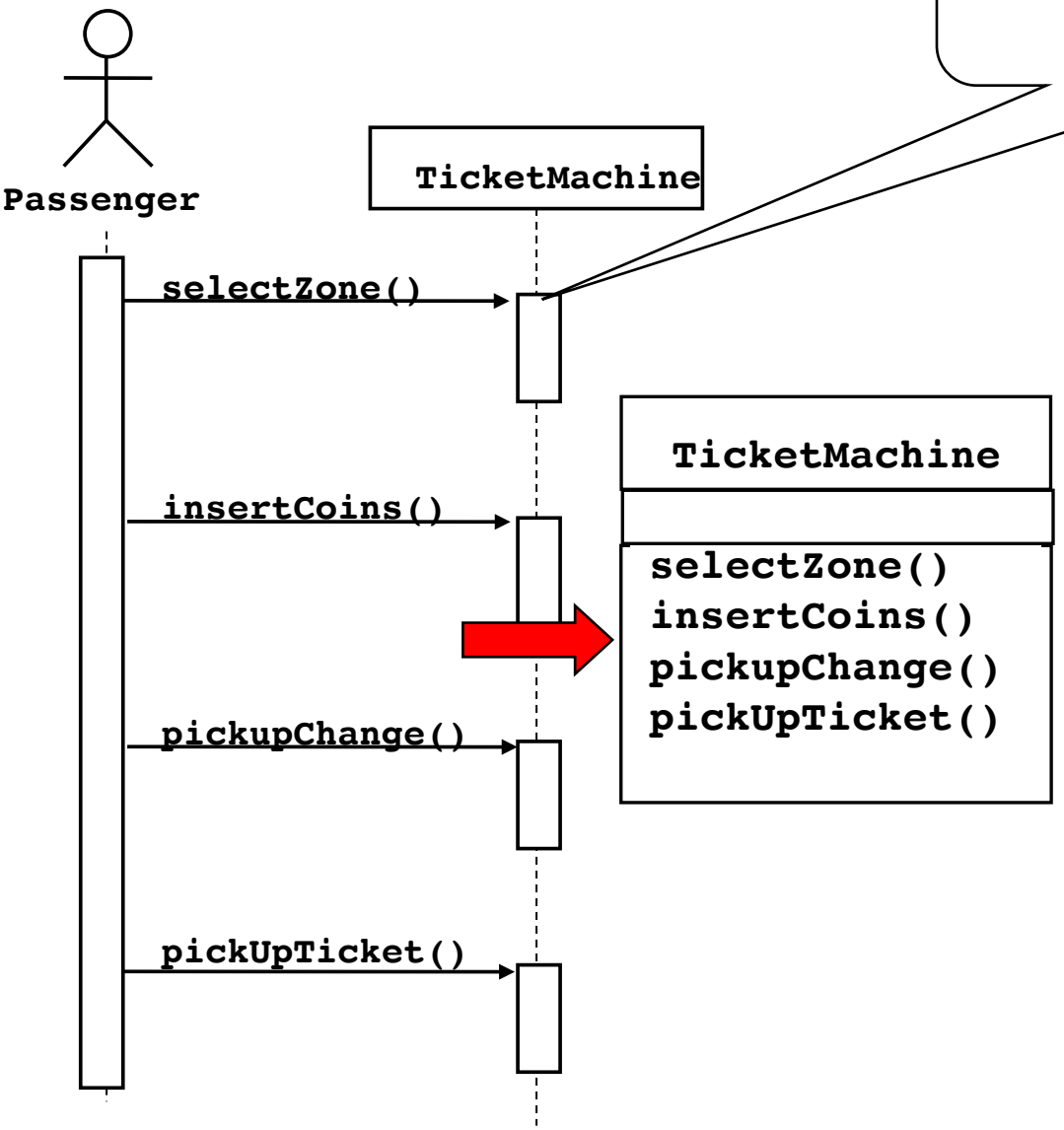


Comprehensive example

- **Sequence diagram related to the Use case: to move in the elevator.**



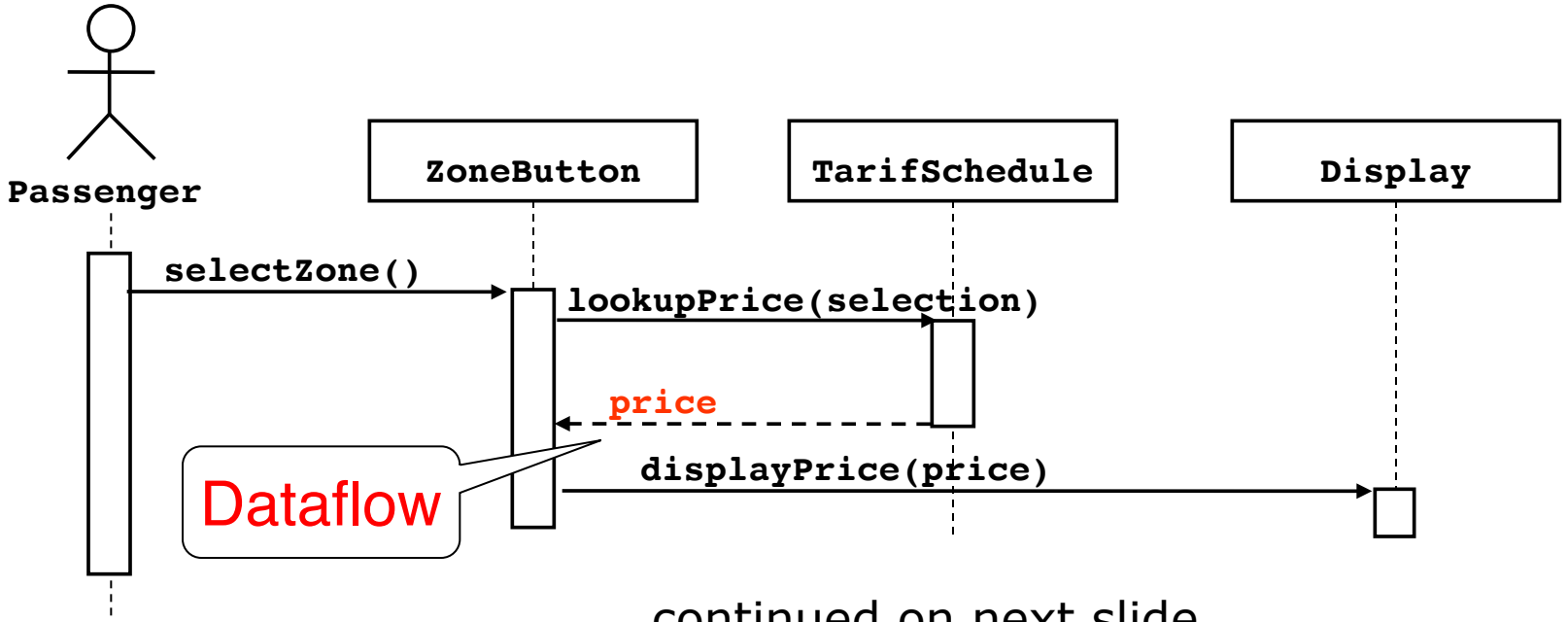
Sequence Diagrams



Focus on Controlflow

- Used during analysis
  - To refine use case descriptions
  - to find additional objects ("participating objects")
- Used during system design
  - to refine subsystem interfaces
- ***Instances*** are represented by rectangles. ***Actors*** by sticky figures
- ***Lifelines*** are represented by dashed lines
- ***Messages*** are represented by arrows
- ***Activations*** are represented by narrow rectangles.

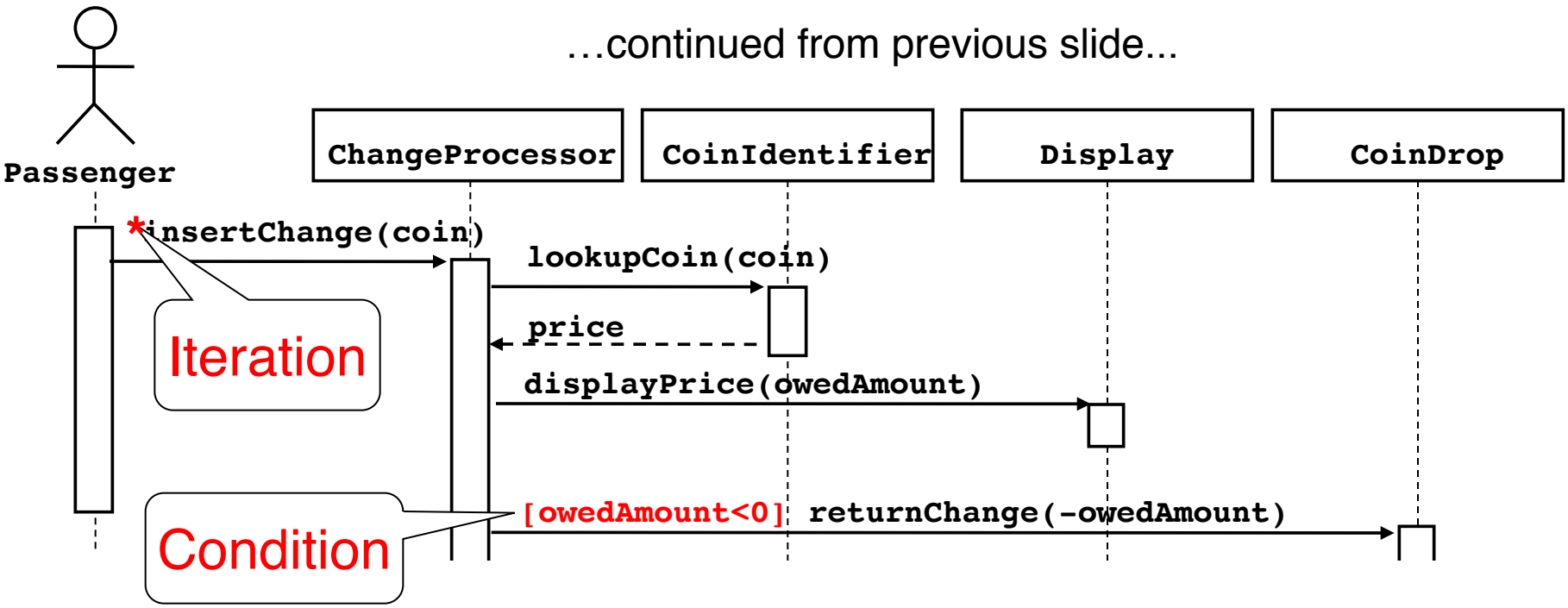
Sequence Diagrams can also model the Flow of Data



...continued on next slide...

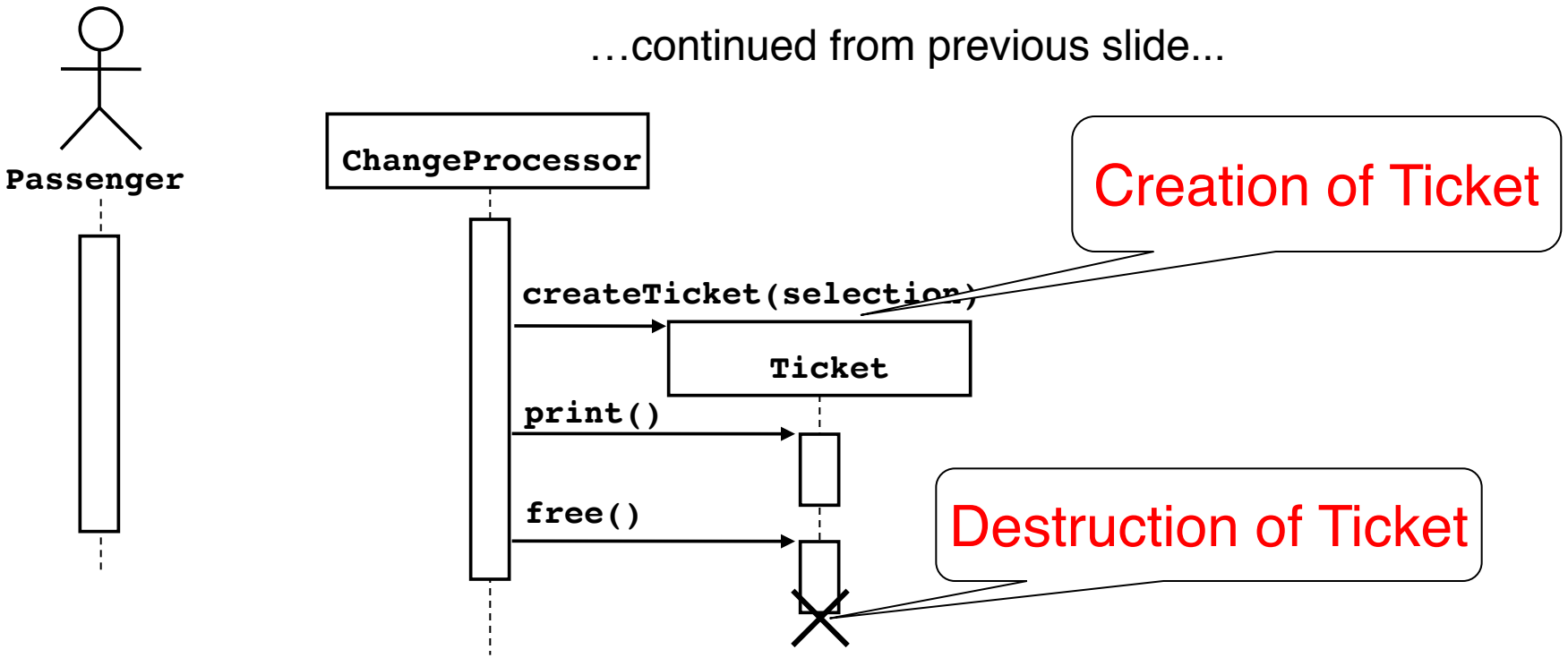
- The source of an arrow indicates the activation which sent the message
- Horizontal dashed arrows indicate data flow, for example return results from a message

Sequence Diagrams: Iteration & Condition



- Iteration is denoted by a * preceding the message name
- Condition is denoted by boolean expression in [ ] before the message name

Creation and destruction



- Creation is denoted by a message arrow pointing to the object
- Destruction is denoted by an X mark at the end of the destruction activation
  - In garbage collection environments, destruction can be used to denote the end of the useful life of an object.

An Example

- Flow of events in "Get SeatPosition" use case :

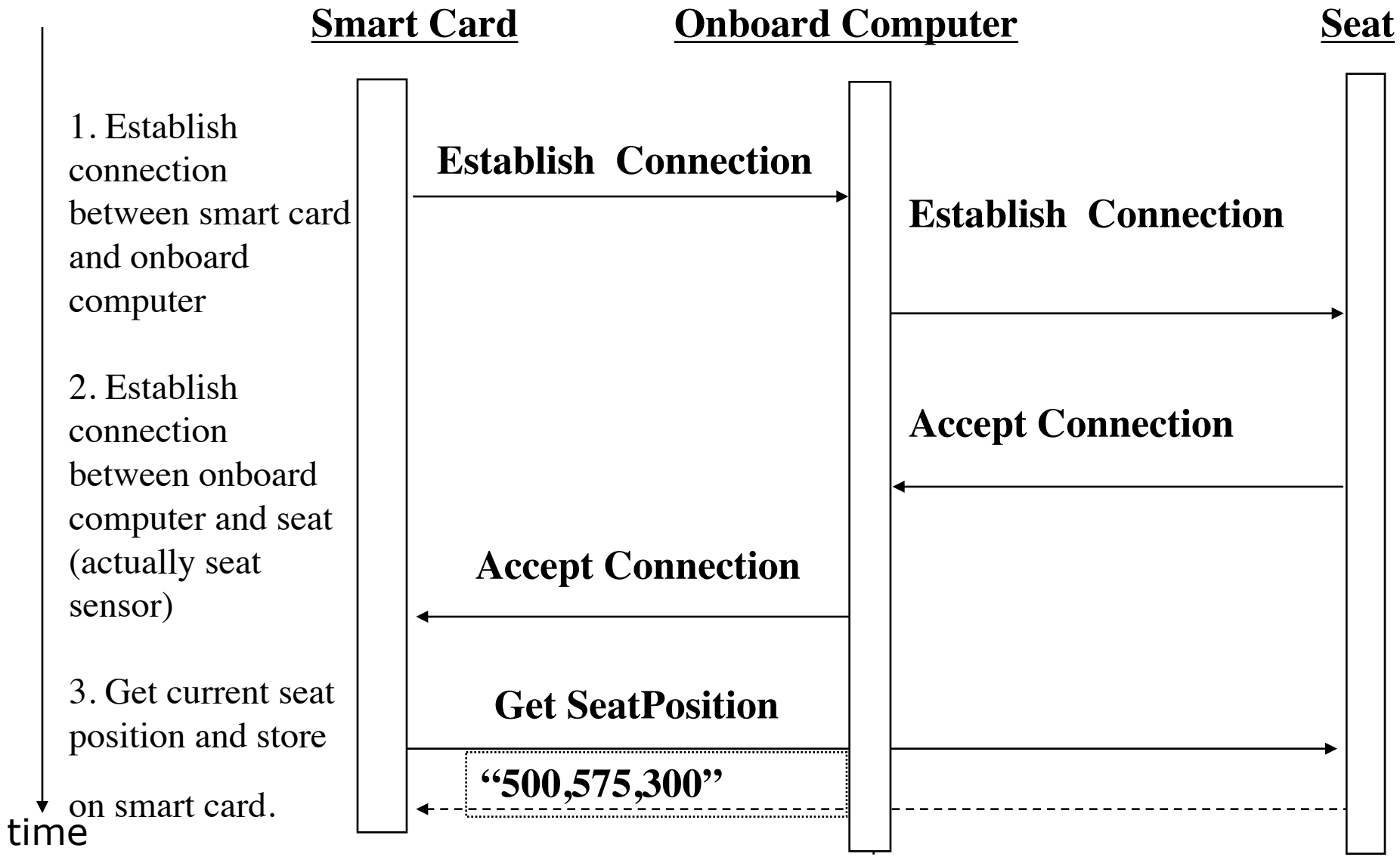
  1. Establish connection between smart card and onboard computer

  2. Establish connection between onboard computer and sensor for seat

  3. Get current seat position and store on smart card

- Where are the objects?

Sequence Diagram for "Get SeatPosition"



Heuristics for Sequence Diagrams

- Layout:
  1st column: Should be the actor of the use case
  2nd column: Should be a boundary object
  3rd column: Should be the control object that manages the rest of the use case

- Creation of objects:
  - Create control objects at beginning of event flow
  - The control objects create the boundary objects
- Access of objects:
  - Entity objects can be accessed by control and boundary objects
  - Entity objects should not access boundary or control objects.

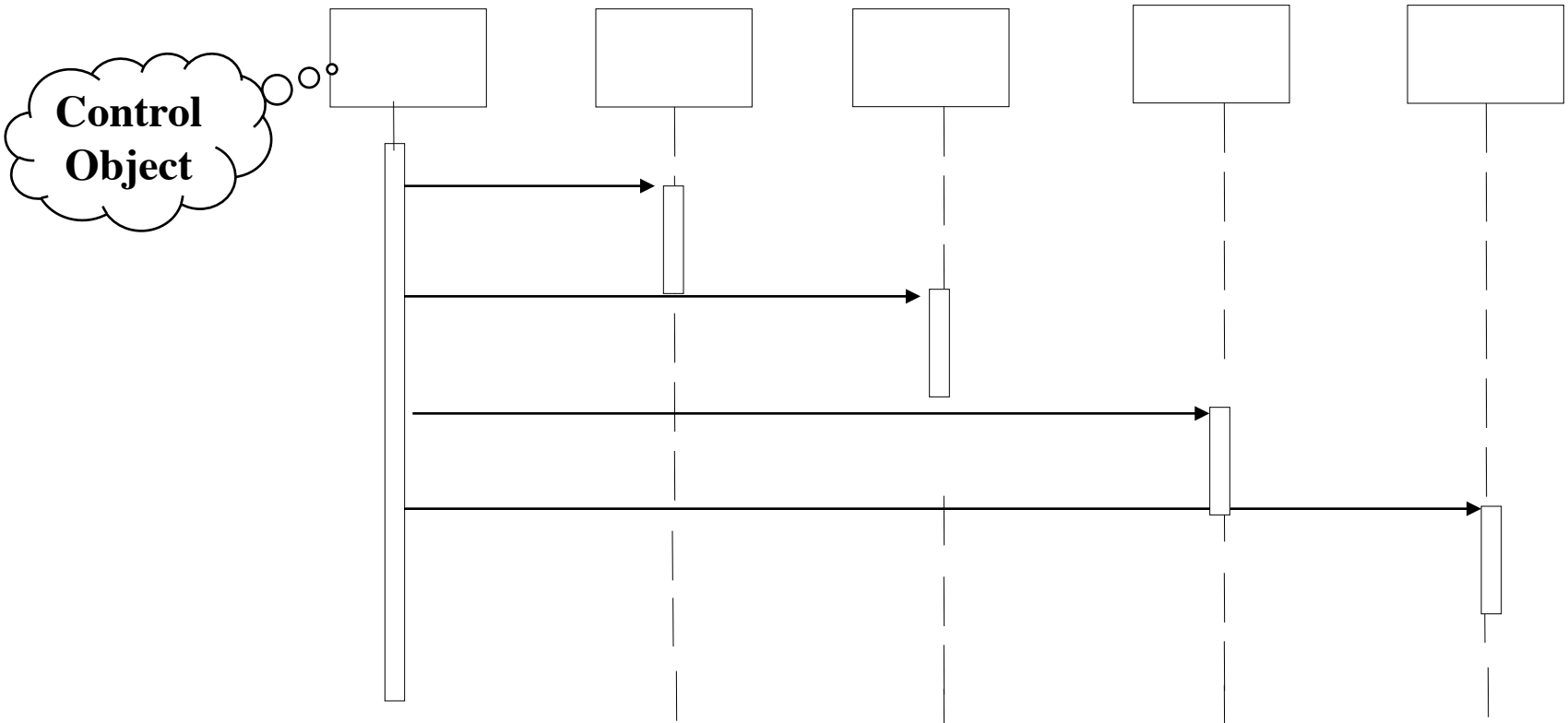
What else can we get out of Sequence Diagrams?

- Sequence diagrams are derived from use cases

- The structure of the sequence diagram helps us to determine how decentralized the system is
- We distinguish two structures for sequence diagrams
  - Fork Diagrams and Stair Diagrams (Ivar Jacobsen)

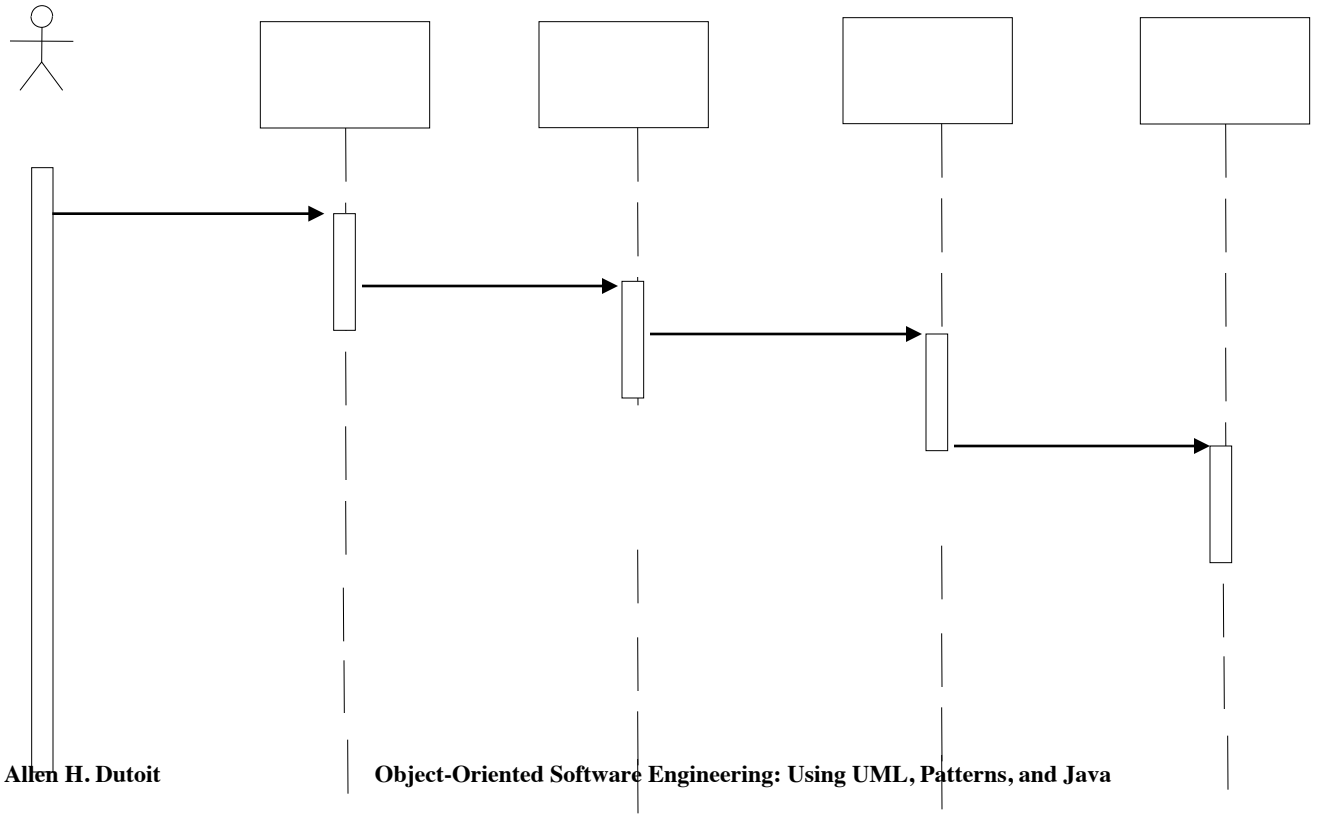
Fork Diagram

- The dynamic behavior is placed in a single object, usually a control object
  - It knows all the other objects and often uses them for direct questions and commands



Stair Diagram

- The dynamic behavior is distributed. Each object delegates responsibility to other objects
  - Each object knows only a few of the other objects and knows which objects can help with a specific behavior



Fork or Stair?

- Object-oriented supporters claim that the stair structure is better
- Modeling Advice:
  - Choose the stair - a decentralized control structure - if
    - The operations have a strong connection
    - The operations will always be performed in the same order
  - Choose the fork - a centralized control structure - if
    - The operations can change order
    - New operations are expected to be added as a result of new requirements.

Dynamic Modeling

- We distinguish between two types of operations:
  - Activity: Operation that takes time to complete
    - associated with states
  - Action: Instantaneous operation
    - associated with events
- A state chart diagram relates events and states for one class
- An object model with several classes with interesting behavior has *a set* of state diagrams