☛ **Functional Modeling**

Use case diagrams

☛ **Object Modeling**

Class diagrams

☛ **Dynamic Modeling**

Sequence diagrams
State diagrams

Objectives

Discuss what sequence diagram is

Sequence Diagram Applications

Components

How to draw sequence diagrams?

State diagrams

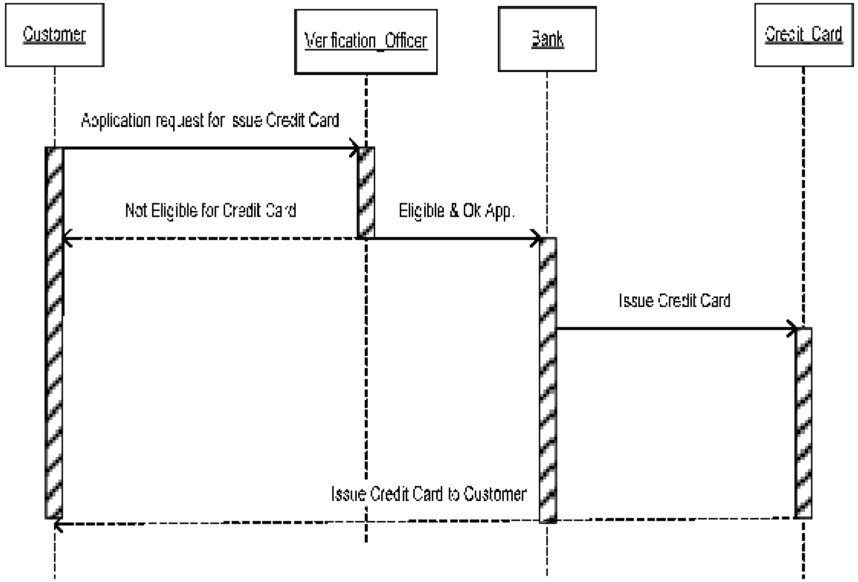
How to draw state diagrams?

Sequence diagrams

- sequence diagram to illustrate use-case realizations
  - to show how objects interact to perform the behavior of all or part of a use case
  - they describe how—and in what order—a group of objects works together
  - they clarify the roles of objects in a flow and thus provide basic input for determining class responsibilities and interfaces.
  - chronological sequences but doesn’t include relationships
  - show the explicit sequence of messages and are better when it is important to visualize the time ordering of messages.

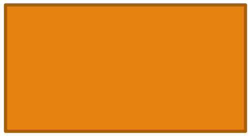
Sequence diagrams

- there are two types of sequence diagrams
- Code based https://www.visual-paradigm.com/tutorials/seqrev.jsp
- UML based

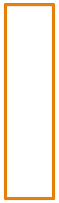

Customer
Verification_Officer
Bank
Credit_Card
Application request for issue Credit Card
Not Eligible for Credit Card
Eligible & Ok App.
Issue Credit Card
Issue Credit Card to Customer

Contents: Components

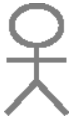
Object Symbol
represents a class, or object, in UML.



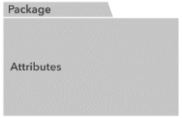
Activation Box
represents the time needed for an object to complete a task.



Actor Symbol
represents by a figure

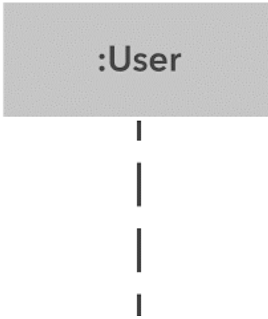


Package Symbol
The shape has a small inner rectangle for labeling the diagram.



Lifeline Symbol

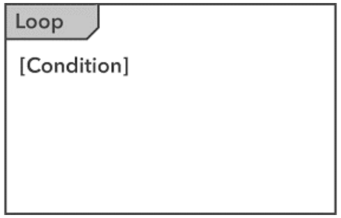
represent the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbolbol



Contents

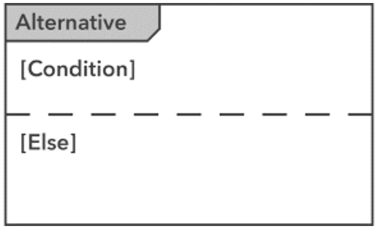
Option Loop Symbol

is used to model "if then" scenarios, i.e., a circumstance that will only occur under certain conditions.



Alternative Symbol

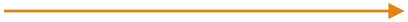
used to symbolize a choice (that is usually mutually exclusive) between two or more message sequences.



Contents : Message symbols

Synchronous

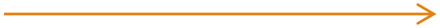
- used when a sender must wait for a response to a message before it continues.
- should show both the call and the reply.



Asynchronus

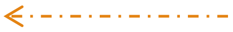
that don't require a response before the sender continues.

only the call should be included in the diagram.



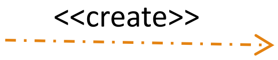
Asynchronus Return

Represented by a dashed line with a lined arrowhead.

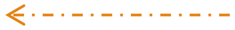


Asynchronus Create

are sent to lifelines in order to create themselves

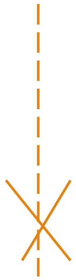


Contents : Message symbols



Reply

are replies to calls.



Delete

indicates the destruction of an object and is placed in its path on the lifeline

Steps to draw a sequence diagram

Step 1: Define who will initiate the interaction

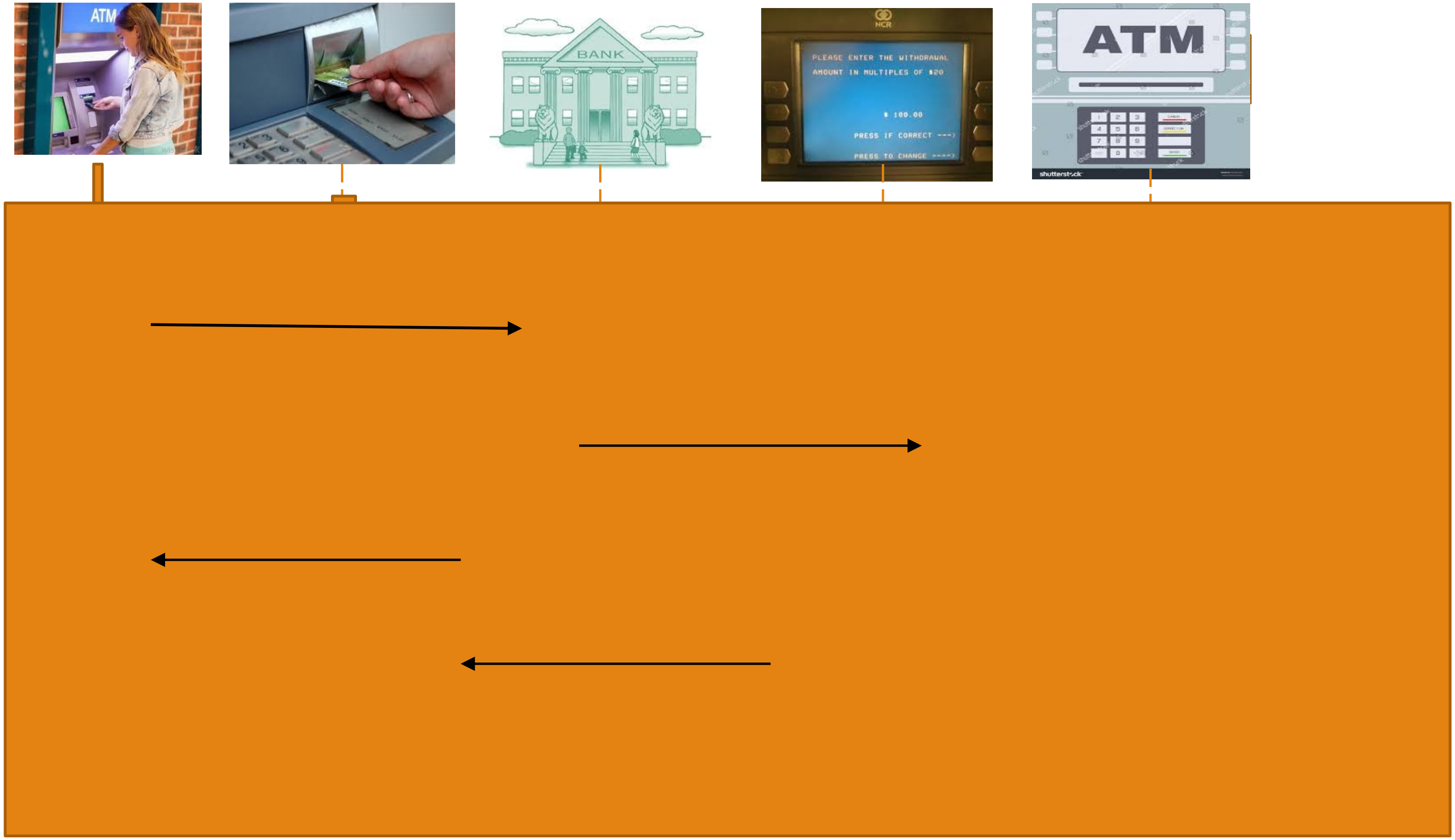
Step 2: Draw the first message to a sub system

Step 3: Draw message to other sub systems

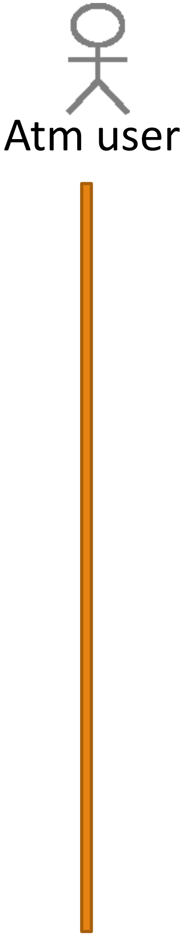
Step 4: Draw return message to actor

Step 5: Send/respond to anonymous actors

Example

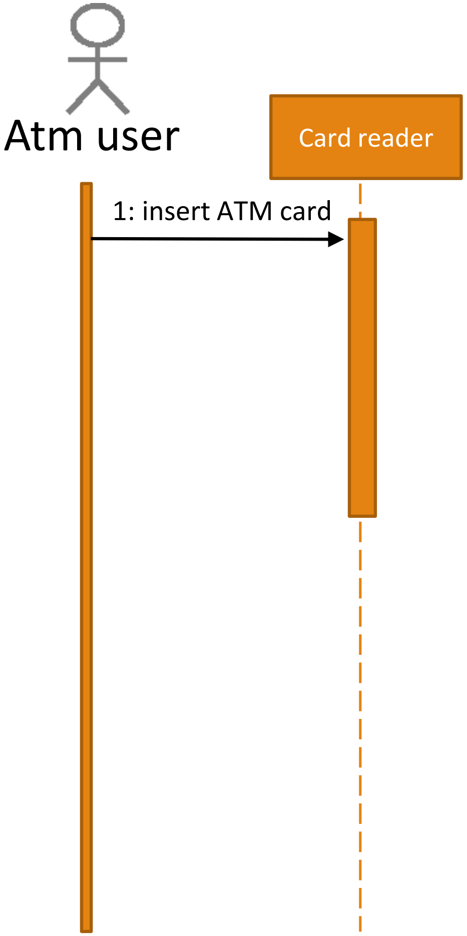

ATM
BANK
NCR
PLEASE ENTER THE WITHDRAWAL AMOUNT IN MULTIPLES OF $20
$ 100.00
PRESS IF CORRECT ---›
PRESS TO CHANGE ---›
ATM
shutterstock

Step 1: Draw who will initiate the interaction

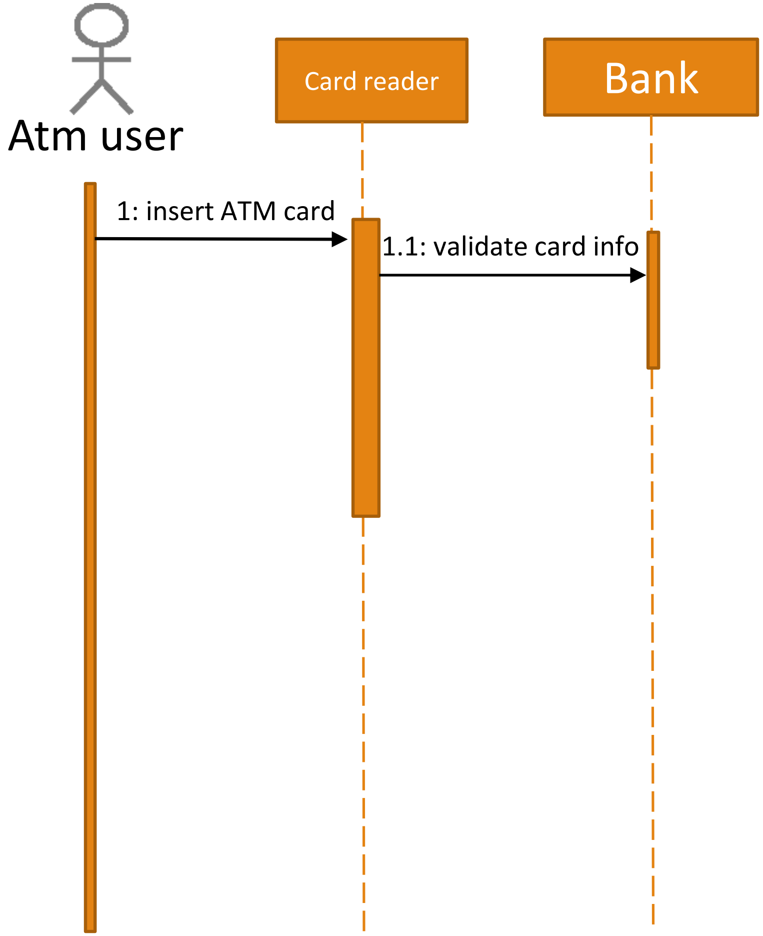


Atm user

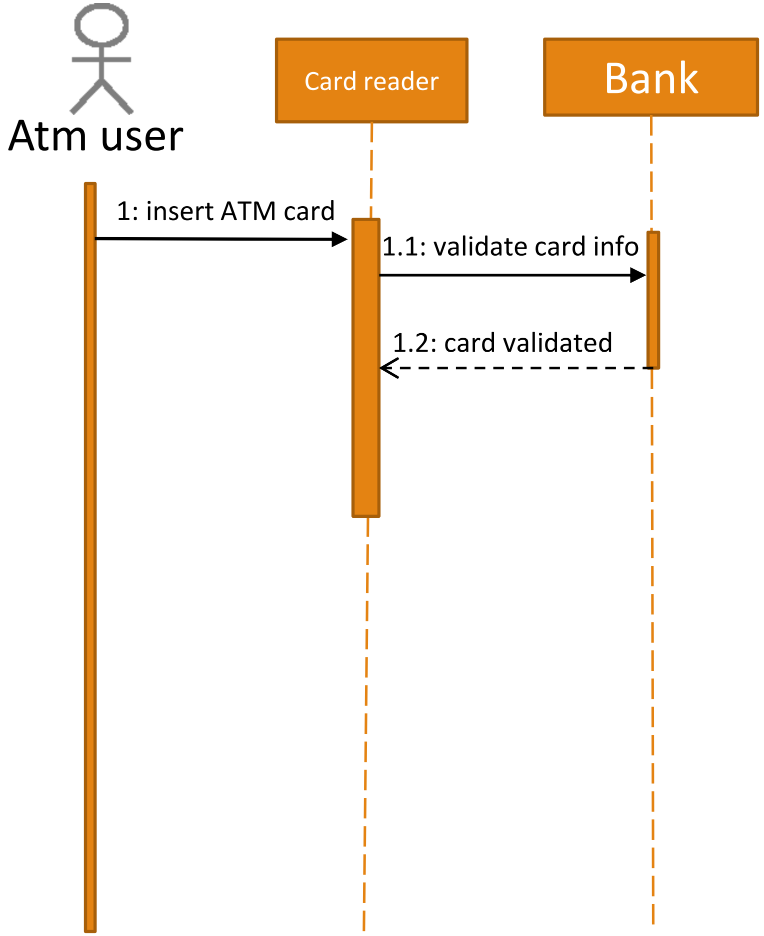
Step 2: Draw the first message to sub-system



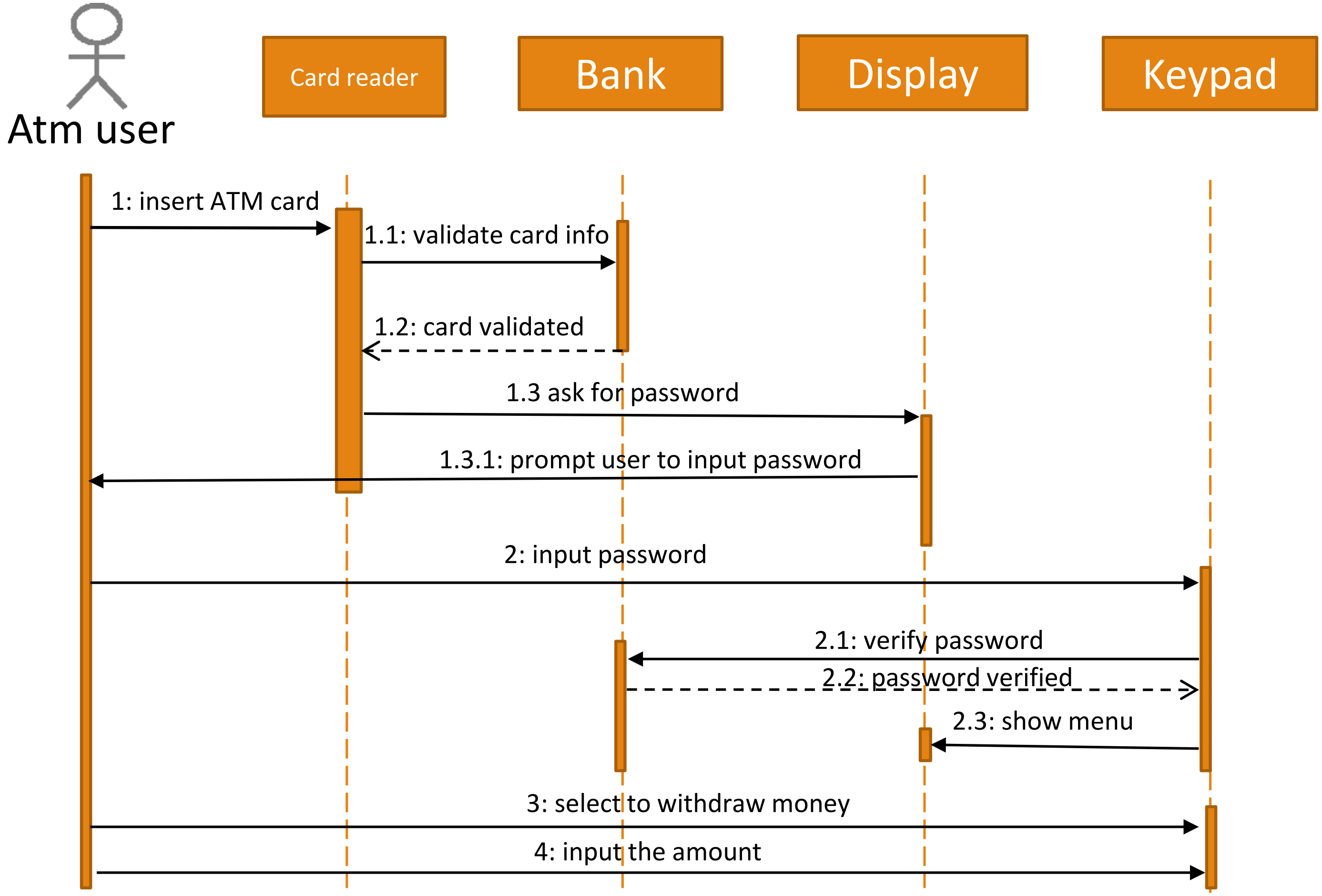
Step 3: Draw message to actor



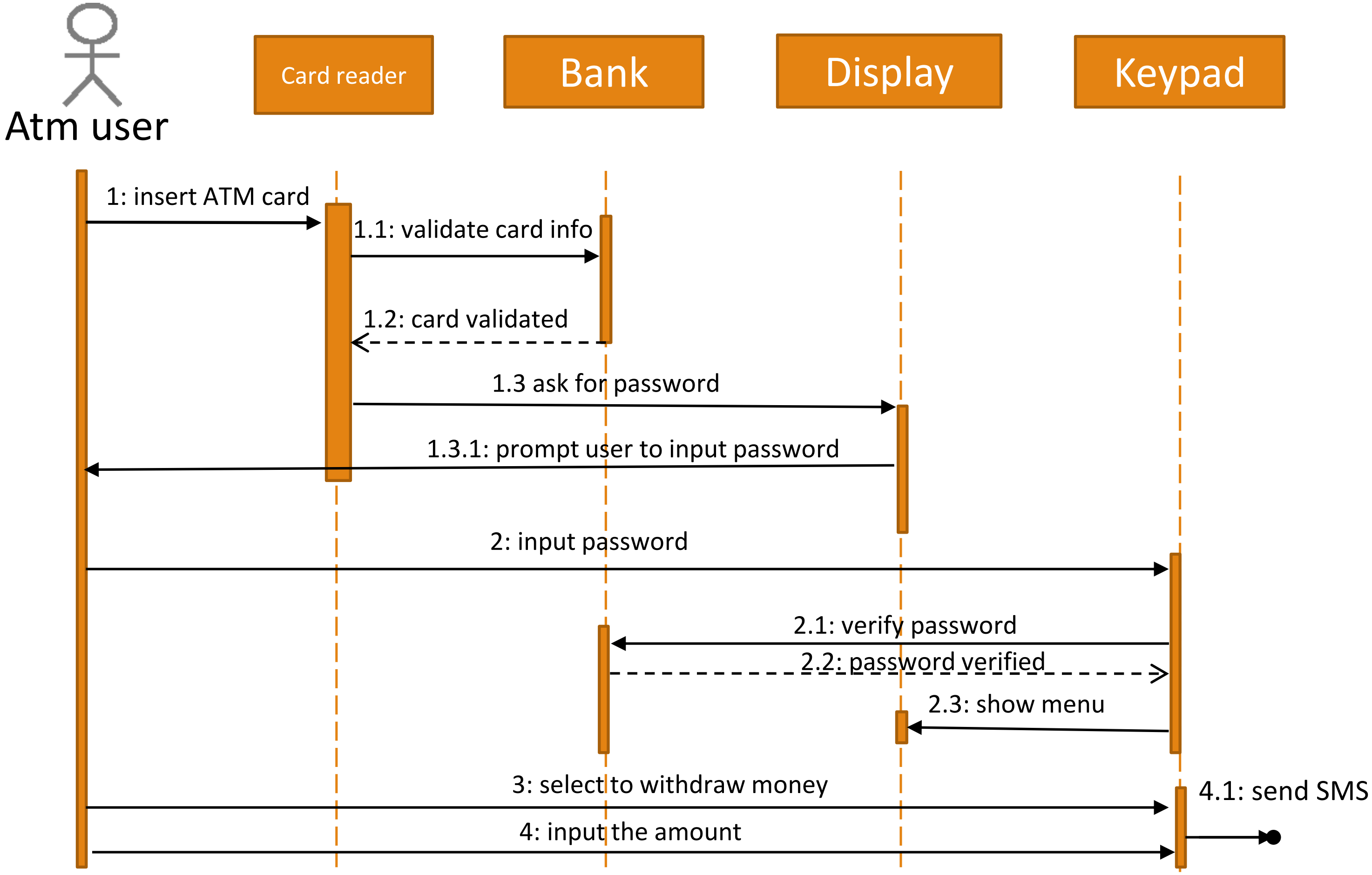
Step 4: Draw message to other sub-systems



Example


Atm user
Card reader
Bank
Display
Keypad
1: insert ATM card
1.1: validate card info
1.2: card validated
1.3 ask for password
1.3.1: prompt user to input password
2: input password
2.1: verify password
2.2: password verified
2.3: show menu
3: select to withdraw money
4: input the amount

Step 5: Send/respond to anonymous actors



What else can we get out of Sequence Diagrams?

Sequence diagrams are derived from use cases

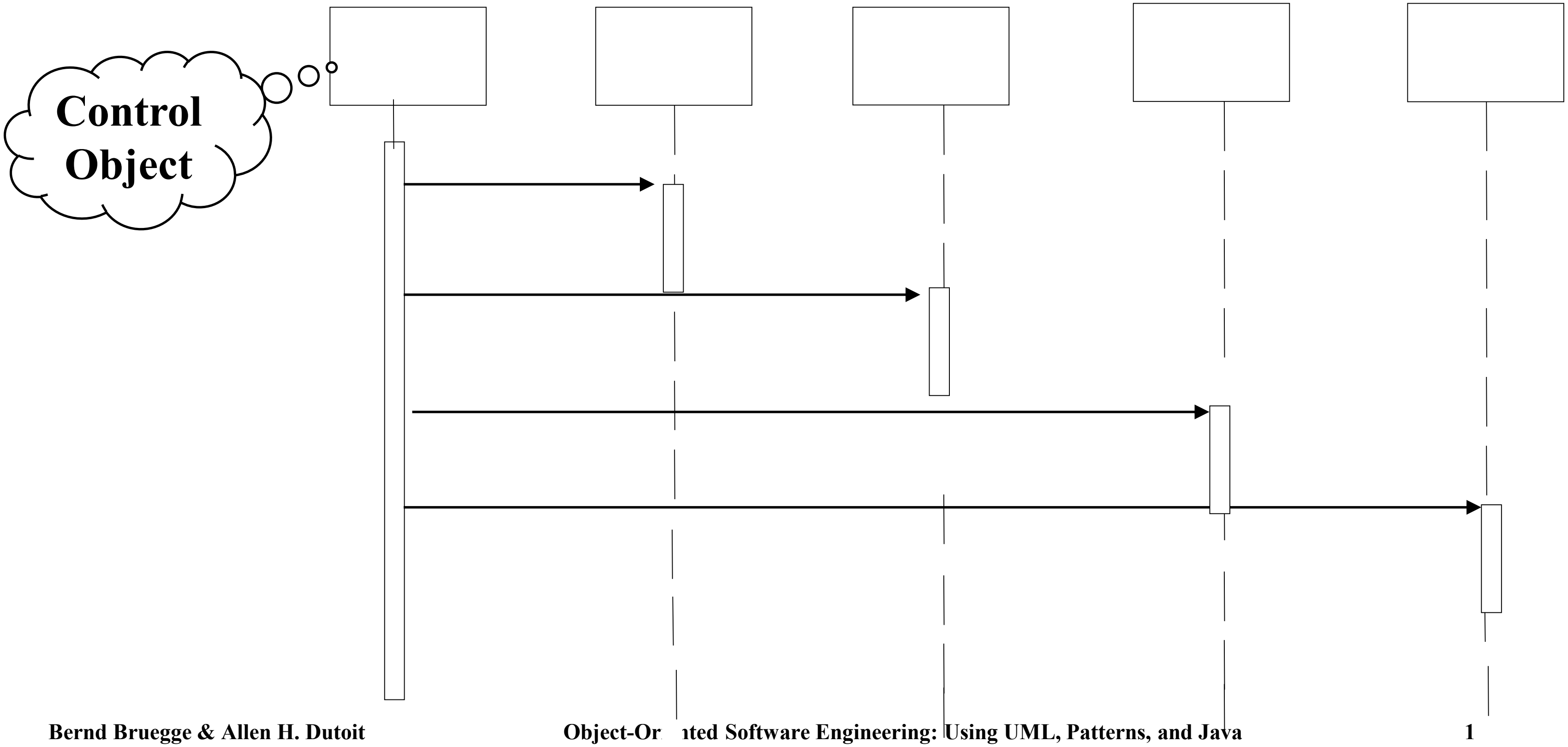
The structure of the sequence diagram helps us to determine how decentralized the system is

We distinguish two structures for sequence diagrams

- Fork Diagrams and Stair Diagrams (Ivar Jacobsen)

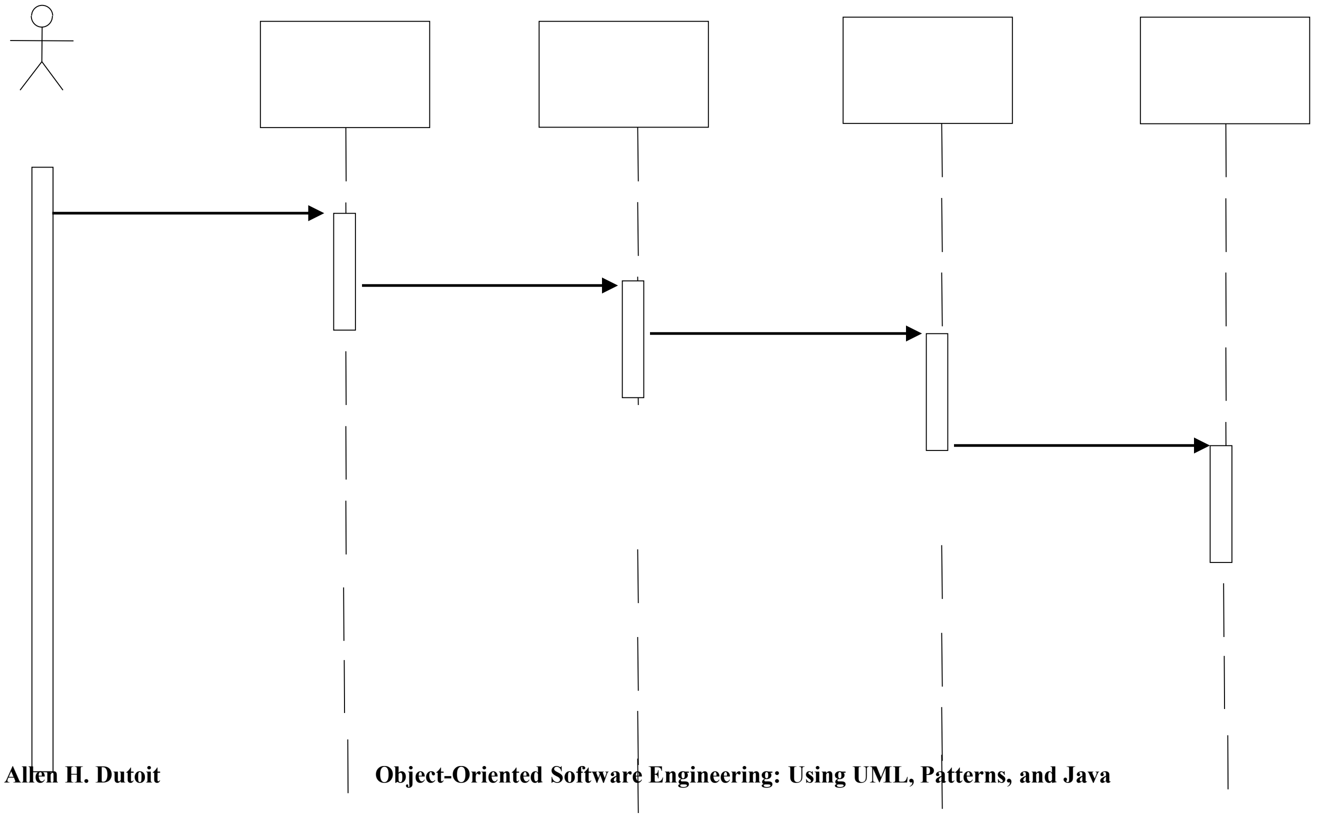
Fork Diagram

- 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



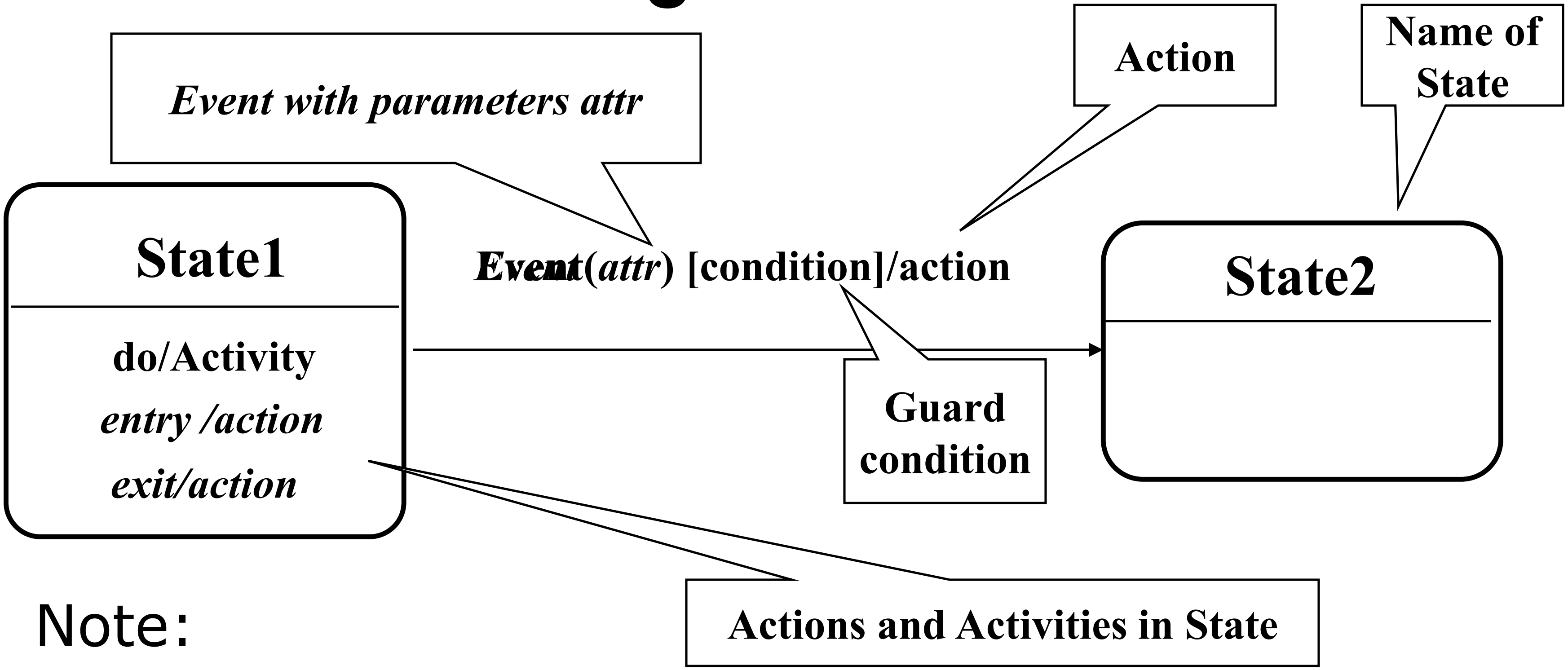
Fork or Stair?

- 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

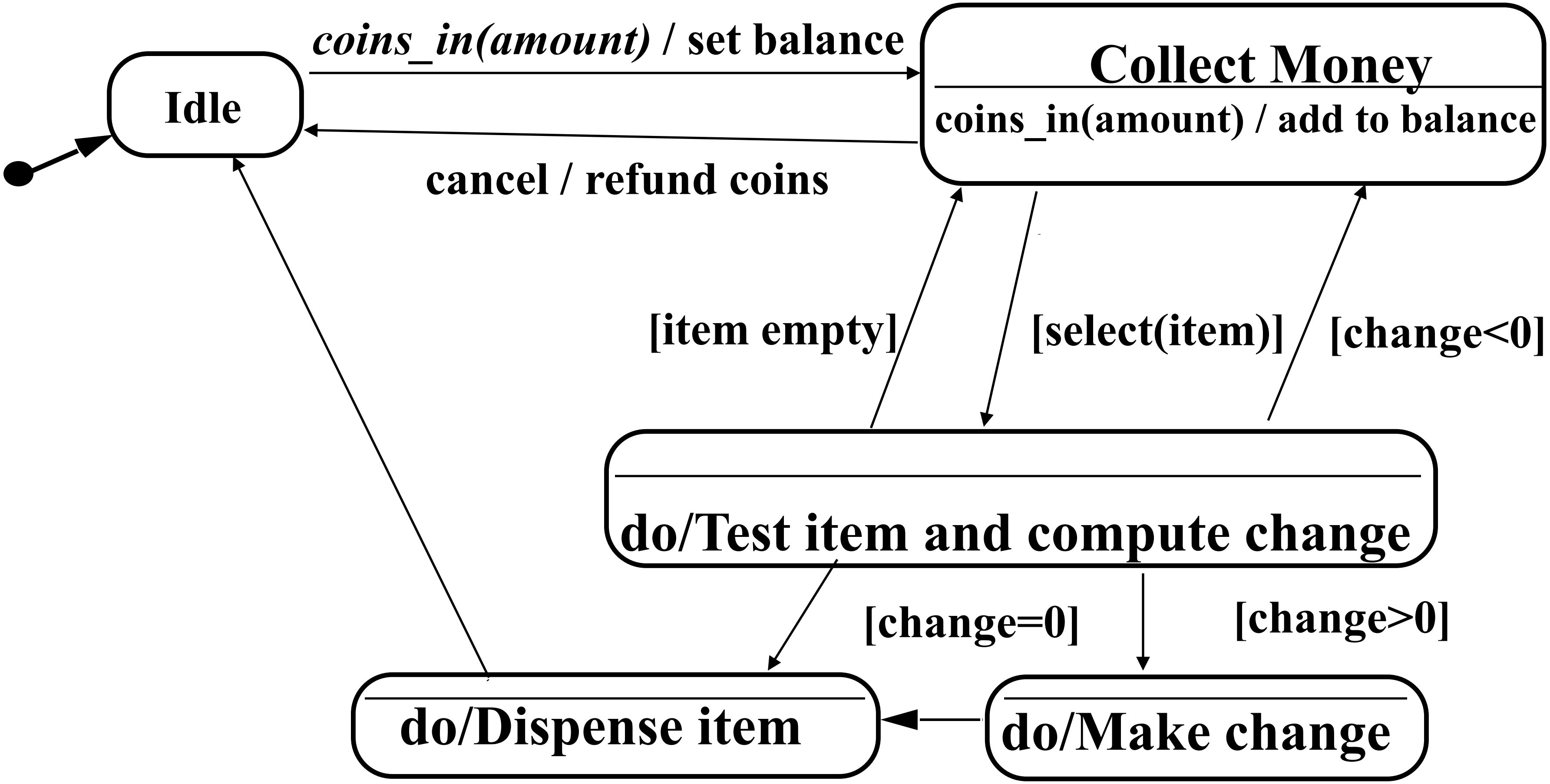
- We distinguish between two types of operations:
  - Activity: Operation that takes time to complete
    - associated with states
  - Action: Instantaneous operation
    - associated with events
- A state chart diagram relates events and states for one class
- An object model with several classes with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

UML Statechart Diagram Notation


Event with parameters attr
Action
Name of State
State1
do/Activity
entry /action
exit/action
Event(attr) [condition]/action
State2
Guard condition
Actions and Activities in State

- Note:
  - *Events are italics*
  - Conditions are enclosed with brackets: []
  - Actions and activities are prefixed with a slash /

Example of a StateChart Diagram



State

- An abstraction of the attributes of a class
  - State is the aggregation of several attributes a class
- A state is an equivalence class of all those attribute values and links that do no need to be distinguished
  - Example: State of a bank
- State has duration

State Chart Diagram vs Sequence Diagram

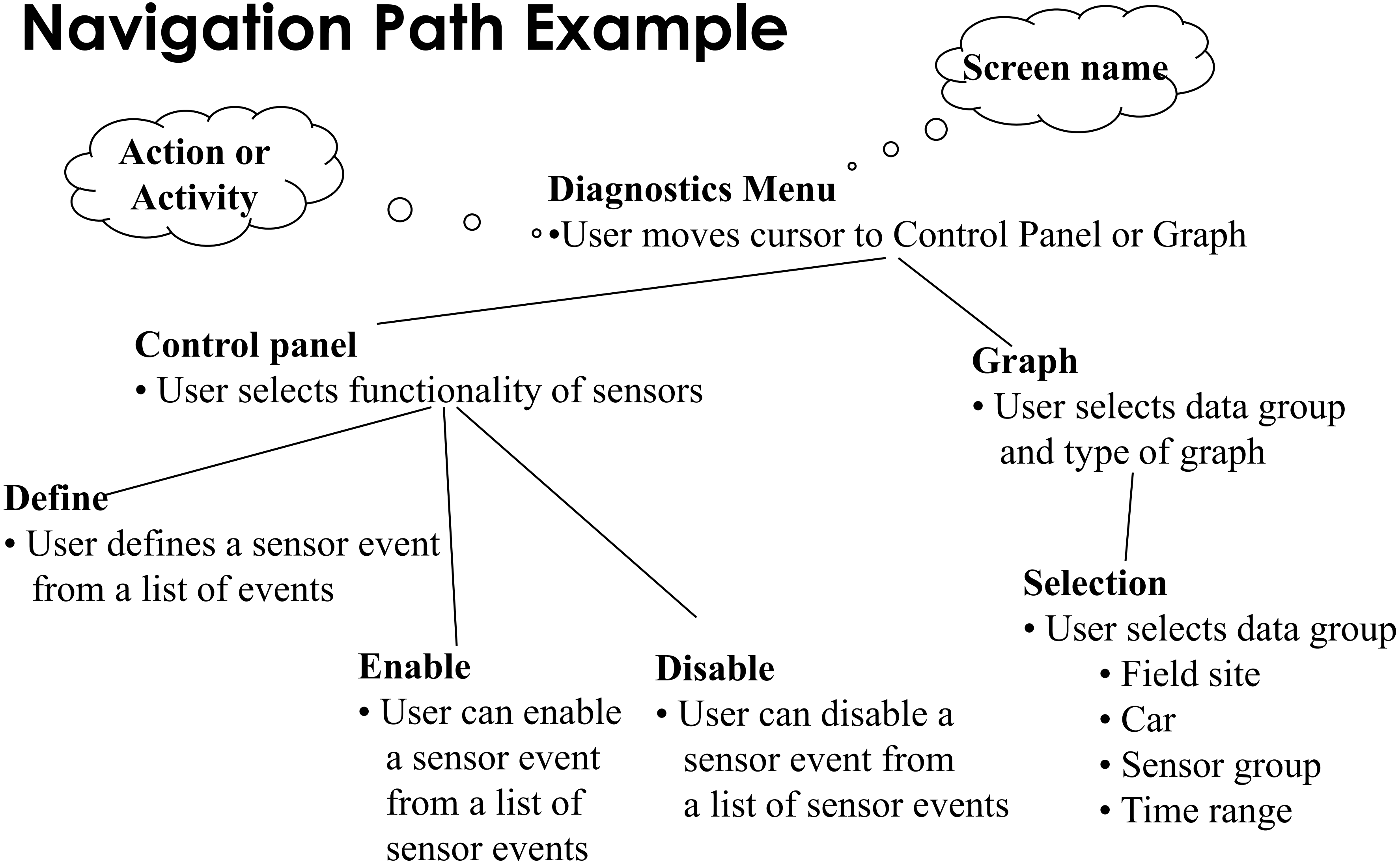
- State chart diagrams help to identify:
  - Changes to an individual object over time

- Sequence diagrams help to identify:
  - The temporal relationship of between objects over time
  - Sequence of operations as a response to one ore more events.

Dynamic Modeling of User Interfaces

- Statechart diagrams can be used for the design of user interfaces
- States: Name of screens
- Actions or activities are shown as bullets under the screen name

Navigation Path Example



Screen name

Action or Activity

Diagnostics Menu

- User moves cursor to Control Panel or Graph

Control panel

- User selects functionality of sensors

Define

- User defines a sensor event from a list of events

Enable

- User can enable a sensor event from a list of sensor events

Disable

- User can disable a sensor event from a list of sensor events

Graph

- User selects data group and type of graph

Selection

- User selects data group
  - Field site
  - Car
  - Sensor group
  - Time range

Practical Tips for Dynamic Modeling

- Construct dynamic models only for classes with significant dynamic behavior
  - Avoid “analysis paralysis”
- Consider only relevant attributes
  - Use abstraction if necessary
- Look at the granularity of the application when deciding on actions and activities
- Reduce notational clutter
  - Try to put actions into superstate boxes (look for identical actions on events leading to the same state).

Problem Statement: Direction Control for a Toy Car

- Power is turned on
  - Car moves forward and car headlight shines
- Power is turned off
  - Car stops and headlight goes out.
- Power is turned on
  - Headlight shines
- Power is turned off
  - Headlight goes out
- Power is turned on
  - Car runs backward with its headlight shining

- Power is turned off
  - Car stops and headlight goes out
- Power is turned on
  - Headlight shines
- Power is turned off
  - Headlight goes out
- Power is turned on
  - Car runs forward with its headlight shining

Find the Functional Model: Use Cases

- Use case 1: System Initialization
  - Entry condition: Power is off, car is not moving
  - Flow of events:
    1. Driver turns power on
  - Exit condition: Car moves forward, headlight is on

- Use case 2: Turn headlight off
  - Entry condition: Car moves forward with headlights on
  - Flow of events:
    1. Driver turns power off, car stops and headlight goes out.
    2. Driver turns power on, headlight shines and car does not move.
    3. Driver turns power off, headlight goes out
  - Exit condition: Car does not move, headlight is out

Use Cases continued

- Use case 3: Move car backward
  - Entry condition: Car is stationary, headlights off
  - Flow of events:
    1. Driver turns power on
  - Exit condition: Car moves backward, headlight on

- Use case 4: Stop backward moving car
  - Entry condition: Car moves backward, headlights on
  - Flow of events:
    1. Driver turns power off, car stops, headlight goes out.
    2. Power is turned on, headlight shines and car does not move.
    3. Power is turned off, headlight goes out.
  - Exit condition: Car does not move, headlight is out

Use Cases Continued

- Use case 5: Move car forward
  - Entry condition: Car does not move, headlight is out
  - Flow of events
    1. Driver turns power on
  - Exit condition:
    - Car runs forward with its headlight shining

Use Case Pruning

- Do we need use case 5?
- Let us compare use case 1 and use case 5:

Use case 1: System Initialization

- Entry condition: Power is off, car is not moving
- Flow of events:
  1. Driver turns power on
- Exit condition: Car moves forward, headlight is on

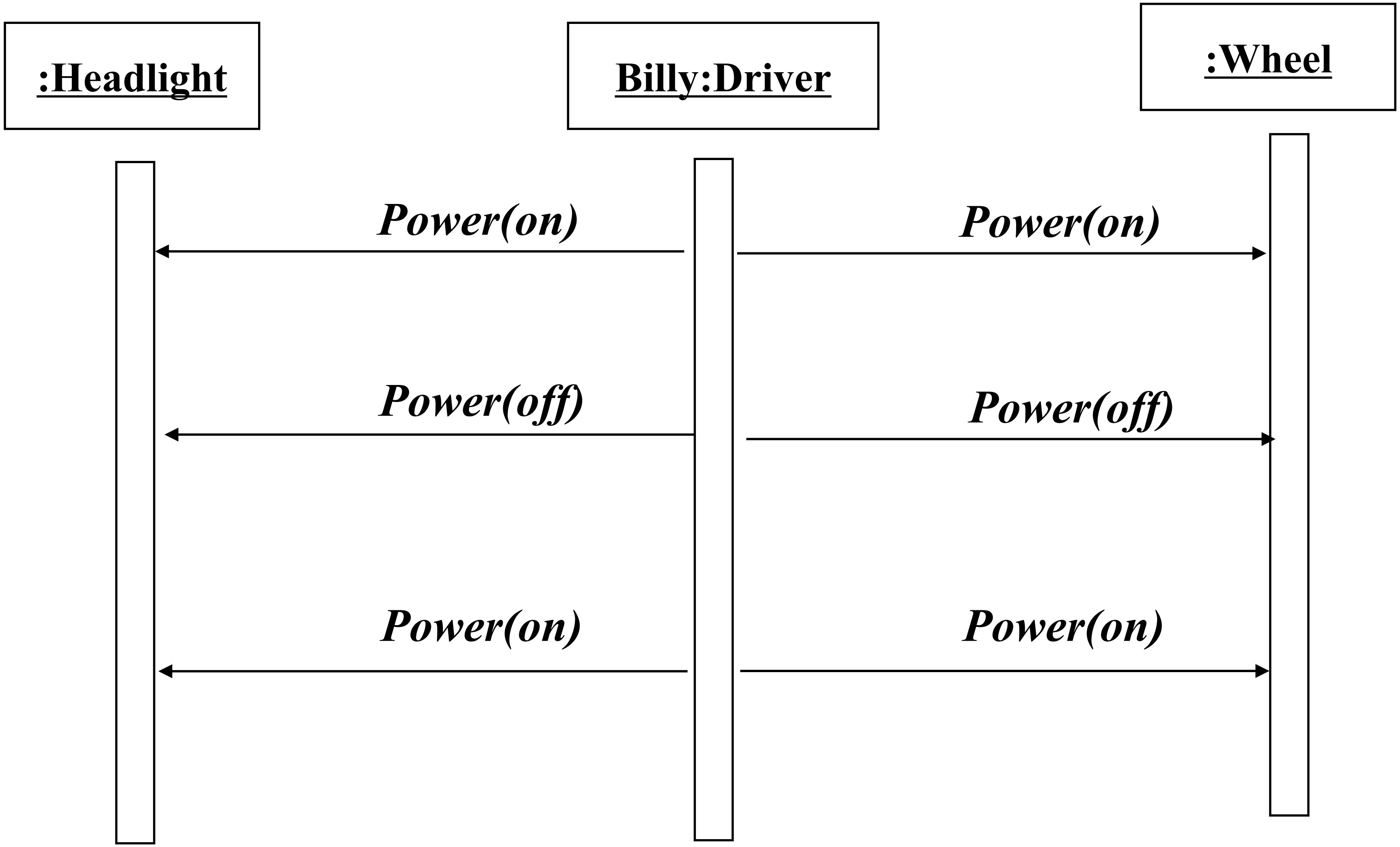
Use case 5: Move car forward

- Entry condition: Car does not move, headlight is out
- Flow of events
  1. Driver turns power on
- Exit condition:
  - Car runs forward with its headlight shining

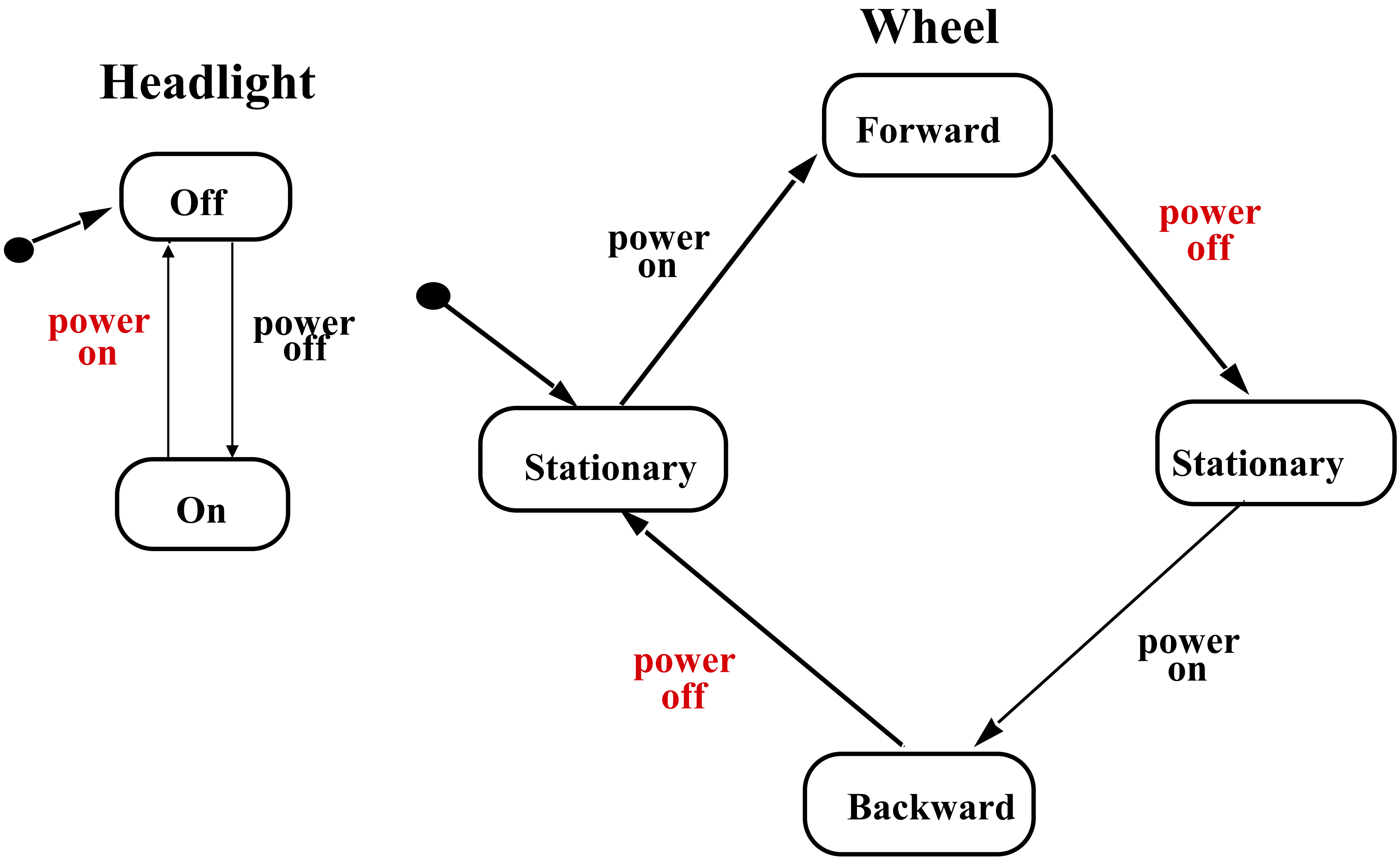
Dynamic Modeling:
Create the Sequence Diagram

- Name: Drive Car
- Sequence of events:
  - Billy turns power on
  - Headlight goes on
  - Wheels starts moving forward
  - Wheels keeps moving forward
  - Billy turns power off
  - Headlight goes off
  - Wheels stops moving
  - . . .

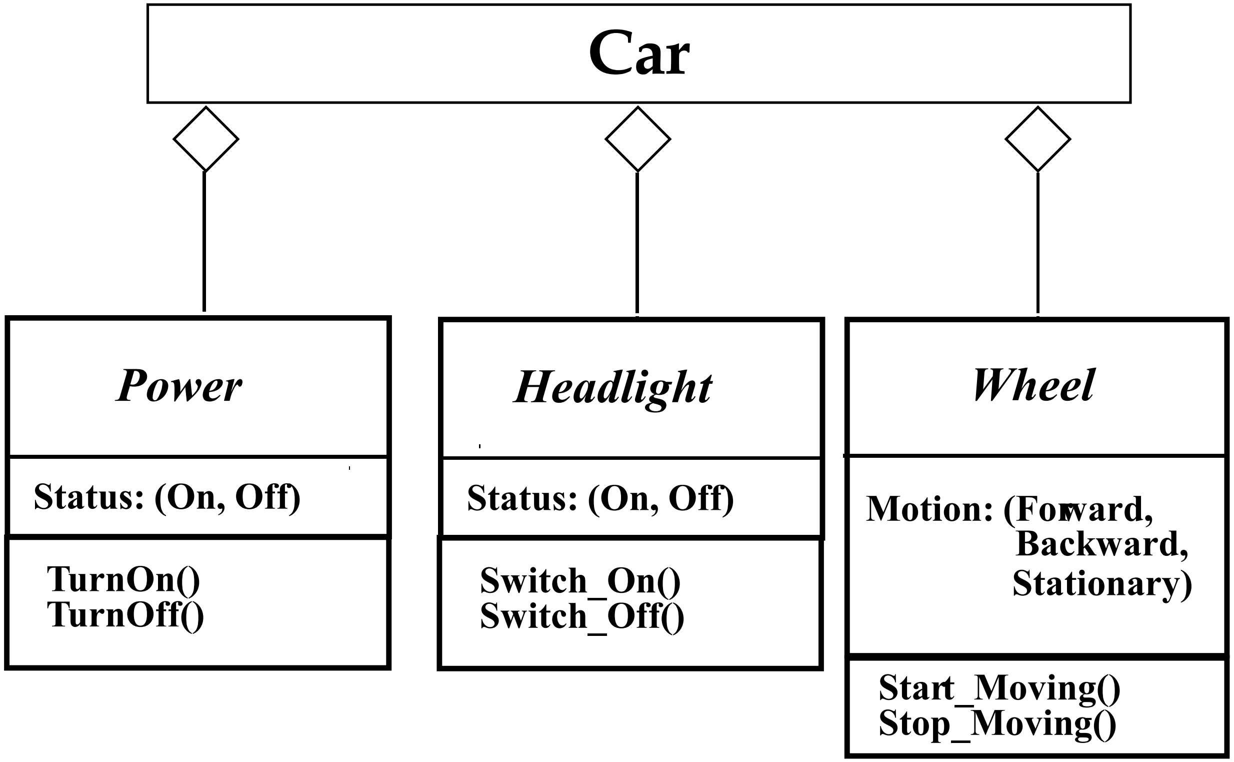
Sequence Diagram for Drive Car Scenario



Toy Car: Dynamic Model



Toy Car: Object Model



Model Validation and Verification

- Verification is an equivalence check between the transformation of two models
- Validation is the comparison of the model with reality
  - Validation is a critical step in the development process Requirements should be validated with the client and the user.
  - Techniques: Formal and informal reviews (Meetings, requirements review)
- Requirements validation involves several checks
  - Correctness, Completeness, Ambiguity, Realistism

Checklist for a Requirements Review

- Is the model correct?
    - A model is correct if it represents the client's view of the the system
- Is the model complete?
    - Every scenario is described
- Is the model consistent?
    - The model does not have components that contradict each other
- Is the model unambiguous?
    - The model describes one system, not many
- Is the model realistic?
    - The model can be implemented

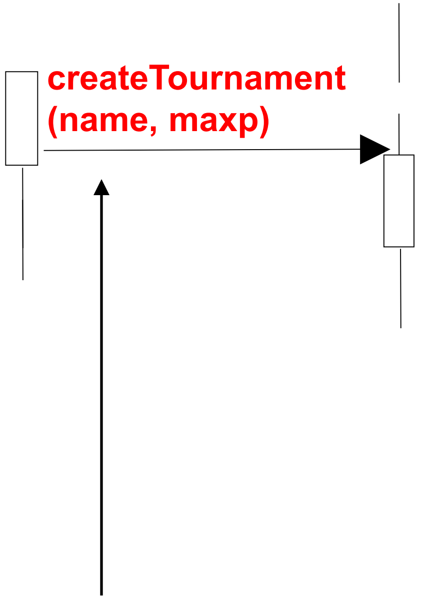
Examples for syntactical Problems

- Different spellings in different UML diagrams

- Omissions in diagrams

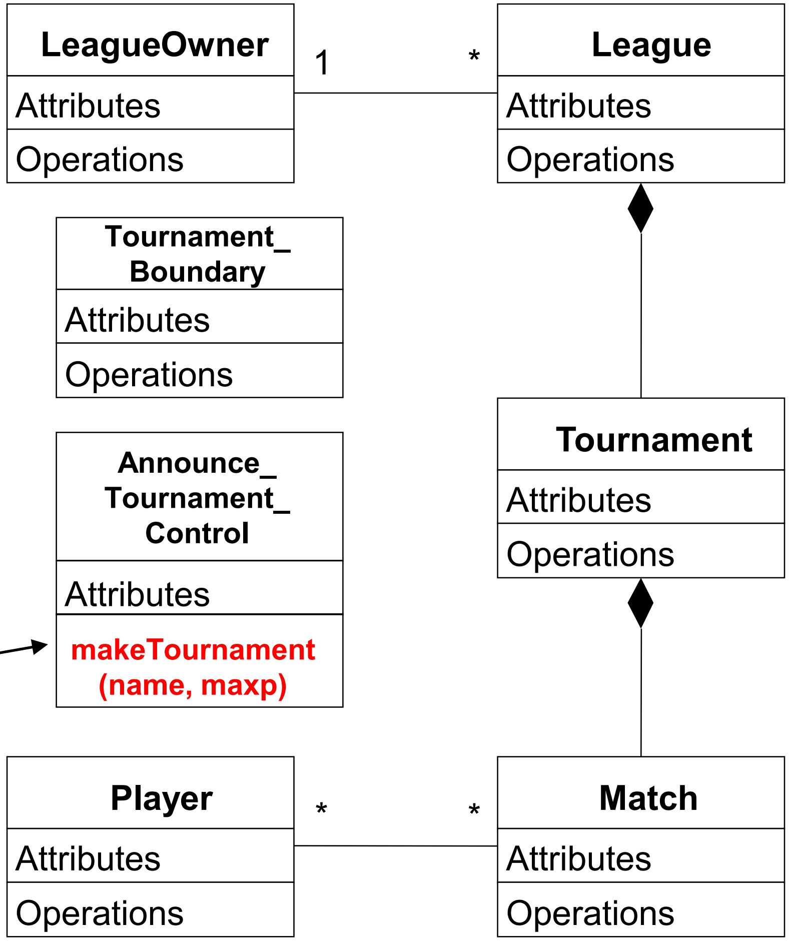
Different spellings in different UML diagrams

UML Sequence Diagram



createTournament
(name, maxp)

UML Class Diagram



LeagueOwner
Attributes
Operations

1 *

League
Attributes
Operations

Tournament_ Boundary
Attributes
Operations

Announce_ Tournament_ Control
Attributes
makeTournament
(name, maxp)

Tournament
Attributes
Operations

Player
Attributes
Operations

* *

Match
Attributes
Operations

Different spellings
in different models
for the same operation

Checklist for the Requirements Review (2)

- Syntactical check of the models
  - Check for consistent naming of classes, attributes, methods in different subsystems
  - Identify dangling associations (“pointing to nowhere”)
  - Identify double- defined classes
  - Identify missing classes (mentioned in one model but not defined anywhere)
  - Check for classes with the same name but different meanings

Requirements Analysis Document Template

1. Introduction
2. Current system
3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Nonfunctional requirements
   - 3.4 Constraints (“Pseudo requirements”)
   - ➡ 3.5 System models
     - 3.5.1 Scenarios
     - 3.5.2 Use case model
     - 3.5.3 Object model
       - 3.5.3.1 Data dictionary
       - 3.5.3.2 Class diagrams
     - 3.5.4 Dynamic models
     - 3.5.5 User interfae
4. Glossary

Section 3.5 System Model

3.5.1 Scenarios

- As-is scenarios, visionary scenarios

3.5.2 Use case model

- Actors and use cases

3.5.3 Object model

- Class diagrams (classes, associations, attributes and operations)

3.5.4 Dynamic model

- State diagrams for classes with significant dynamic behavior
- Sequence diagrams for collaborating objects

3.5.5 User Interface

- Navigational Paths, Screen mockups

Requirements Analysis Questions

1. What are the transformations? ☛ **Functional Modeling**

Create *scenarios and use case diagrams*

- Talk to client, observe, get historical records

2. What is the structure of the system? ☛ **Object Modeling**

Create *class diagrams*

- Identify objects.
- What are the associations between them?
- What is their multiplicity?
- What are the attributes of the objects?
- What operations are defined on the objects?

3. What is its behavior? ☛ **Dynamic Modeling**

Create *sequence diagrams*

- Identify senders and receivers
- Show sequence of events exchanged between objects.
- Identify event dependencies and event concurrency.

Create *state diagrams*

- Only for the dynamically interesting objects.

Summary

- In this lecture, we reviewed the construction of the dynamic model from use case and object models. In particular, we described:
- Sequence and statechart diagrams for identifying new classes and operations.

- In addition, we described the requirements analysis document and its components

Verification vs Validation of models

$$\begin{tikzcd}
R \arrow[r, "\textbf{Analysis}"] \arrow[d, "f_R"'] & M_{\text{Analysis}} \arrow[r, "\textbf{System Design}"] \arrow[d, "f_{MA}"'] & M_{\text{System}} \arrow[r, "\textbf{Object Design}"] \arrow[d, "f_{MS}"'] & M_{\text{Object}} \arrow[r, "\textbf{Implementation}"] \arrow[d, "f_{MD}"'] & M_{\text{Impl}} \arrow[d, "f_{Impl}"'] \\
R \arrow[r] & M_{\text{Analysis}} \arrow[r] & M_{\text{System}} \arrow[r] & M_{\text{Object}} \arrow[r] & M_{\text{Impl}} \\
{} & {} \arrow[l, "\text{Validation}"'] & {} \arrow[l, "\text{Verification}"'] & {} \arrow[l, "\text{Verification}"'] & {} \arrow[l, "\text{Verification}"']
\end{tikzcd}$$

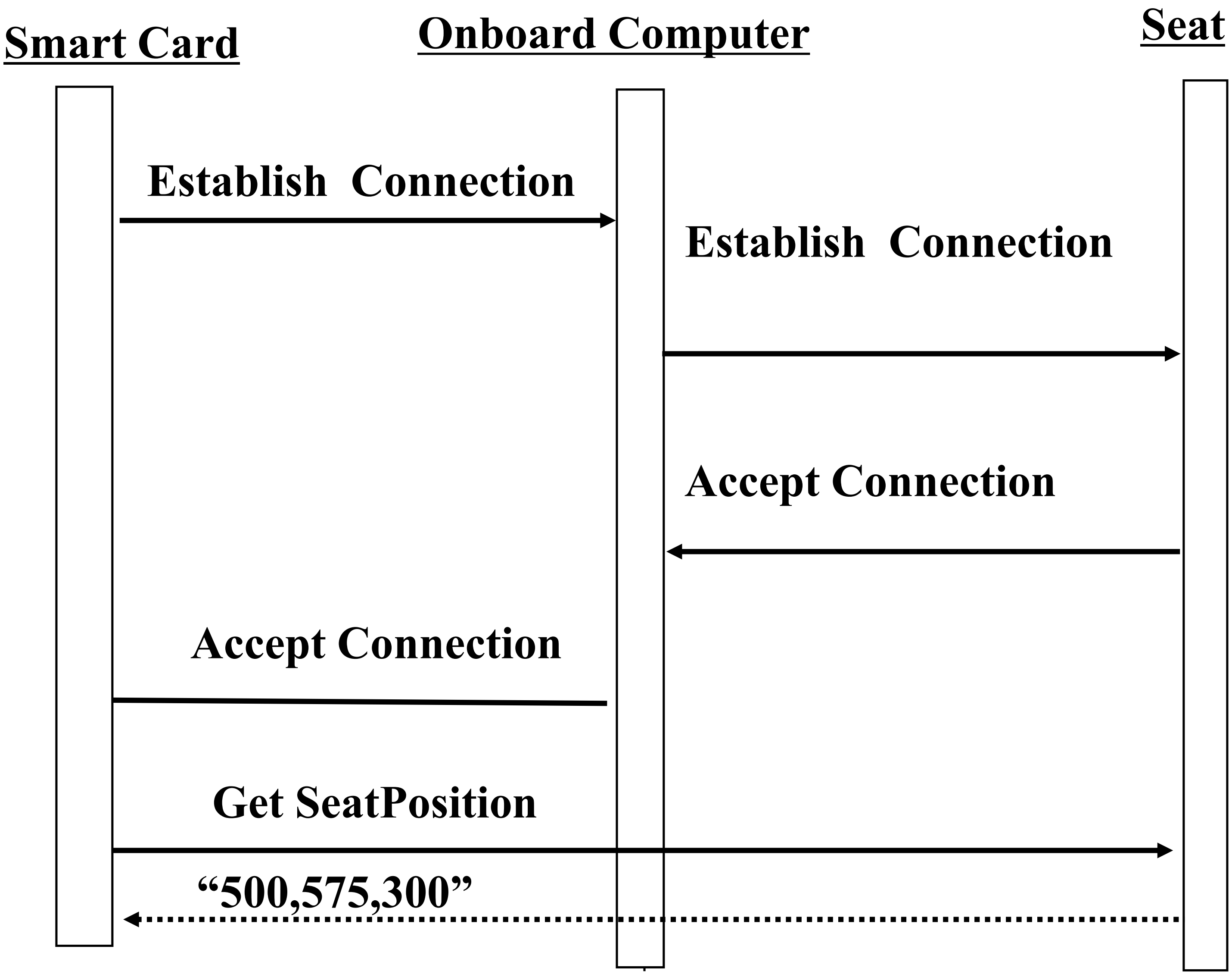
$$\begin{tikzcd}
M \arrow[r, "f_M"'] & M \\
R \arrow[u, "I"] \arrow[r, "f_R"'] & R \arrow[u, "I"']
\end{tikzcd}$$

Is this a good Sequence Diagram?

The first column is not an actor

It is not clear where the boundary object is

It is not clear where the control object is



Deliverables

Dynamic Modeling (state and chart diagrams)
Deadline: **10/08/2017 Tuesday**

Mid-term presentation:
10/18/2017 Wednesday
USE powerpoint template (available beach board)