Software Engineering 1
Special lecture: Modelling Behaviour

In this lecture: Mostly Sequence Diagrams

Ekkart Kindler
Technical University of Denmark
Informatics and Mathematical Modelling

Overview

- Motivation and Idea
  - Automata & StateCharts
  - Interaction Diagrams
- Observations and Discussion
- Sequence Diagrams (in detail)
- Philosophy and Summary

1. Motivation and Idea

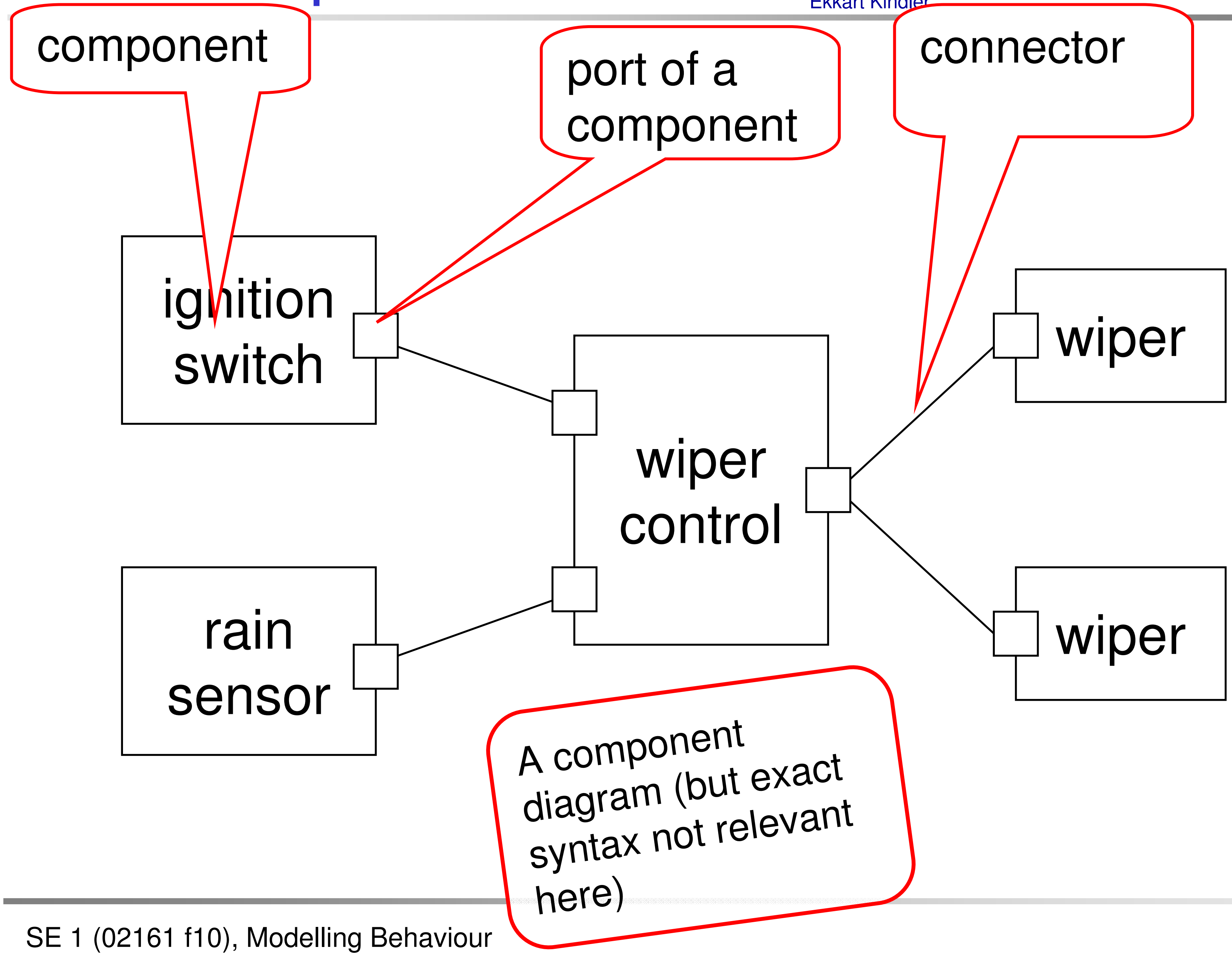
Motivation

- In this course **up to now**:
  Mainly structural models:
  class diagrams, object diagrams

- **Now**: Modelling what the software
  actually should do:
  its functionality and behaviour

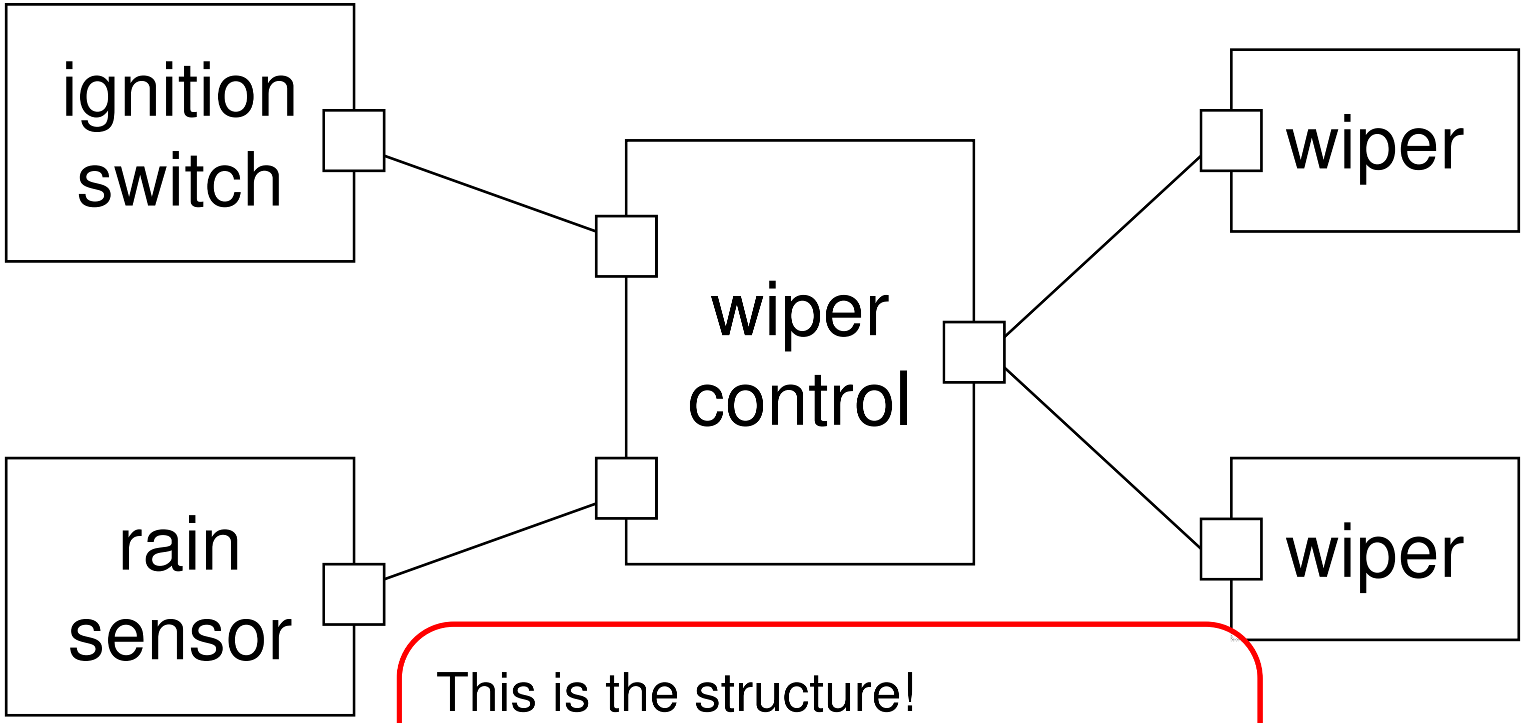
Use cases talk
about functionality;
but at a very early
stage.

An Example



A component diagram (but exact syntax not relevant here)

An Example

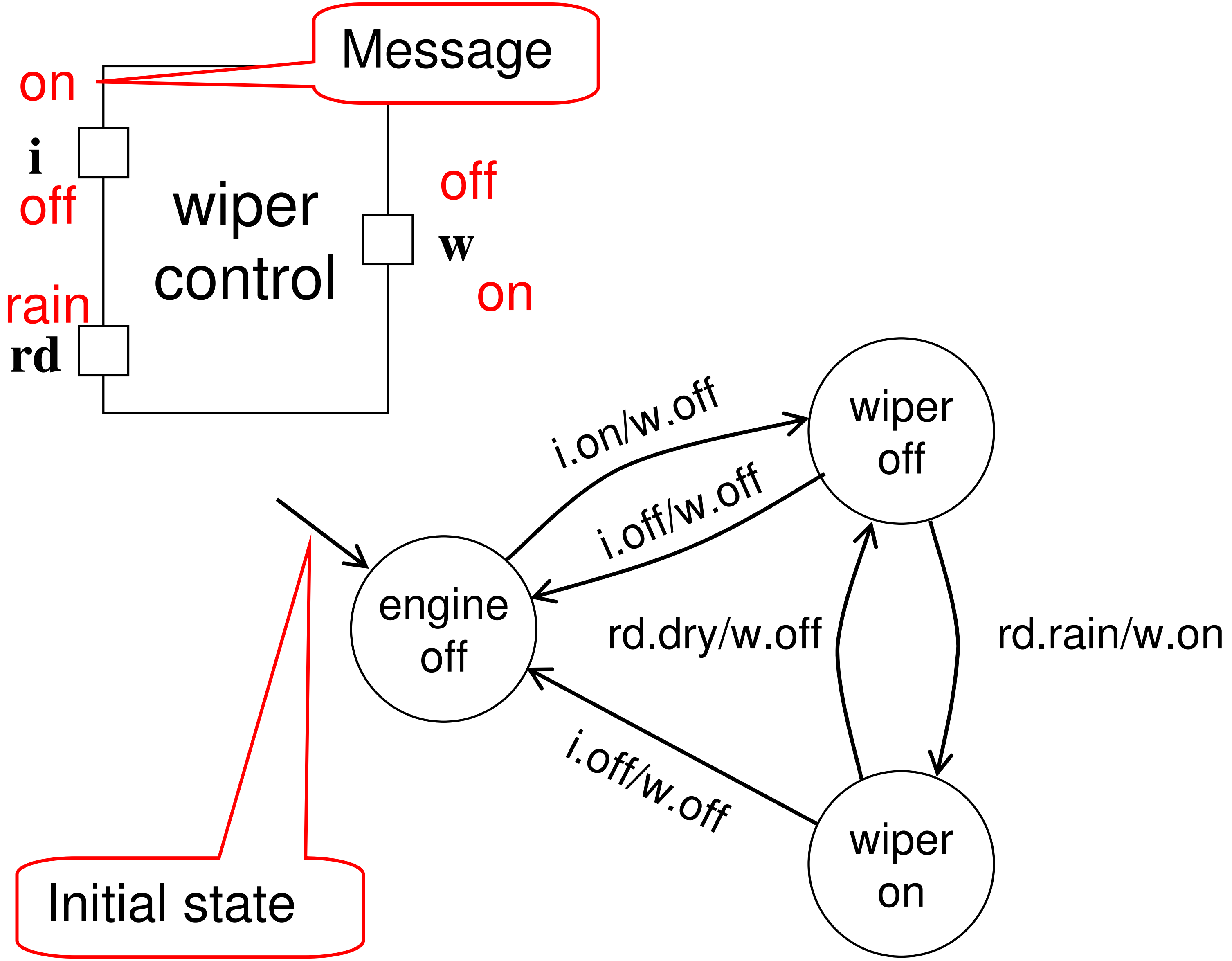


This is the structure!

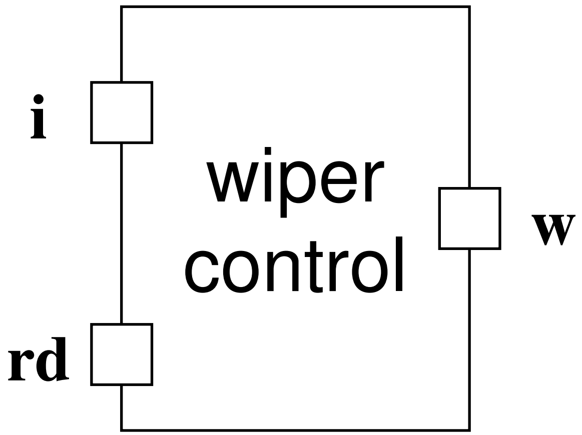
How, does the system behave?

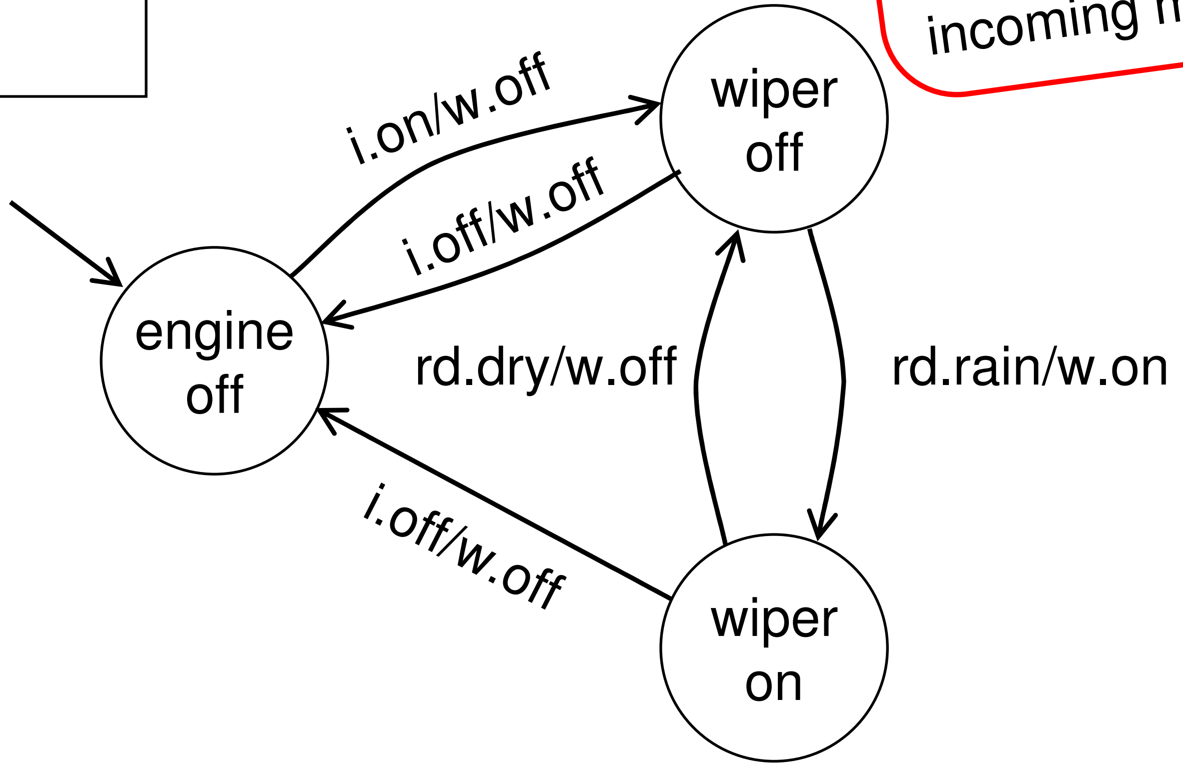
How should it behave?

Behaviour of wiper control



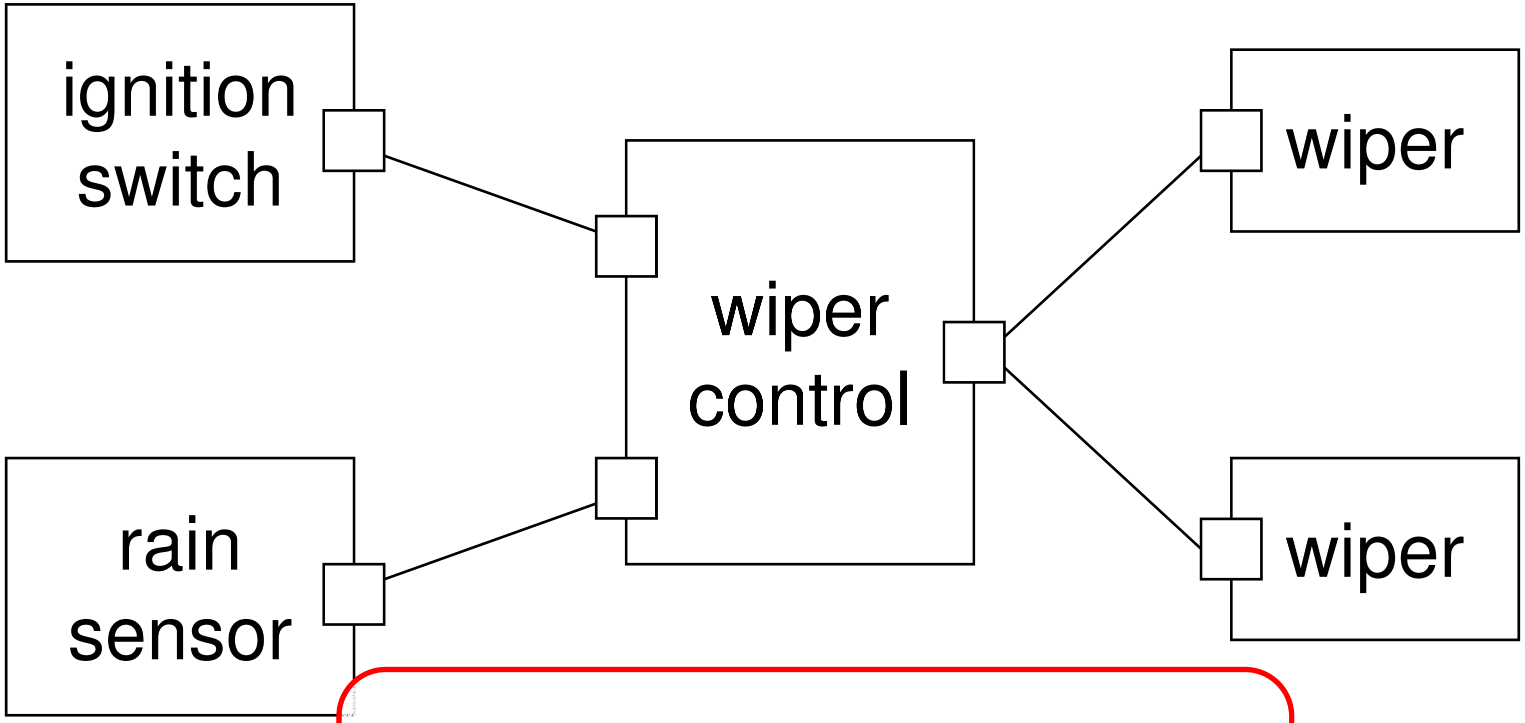
Behaviour of wiper control





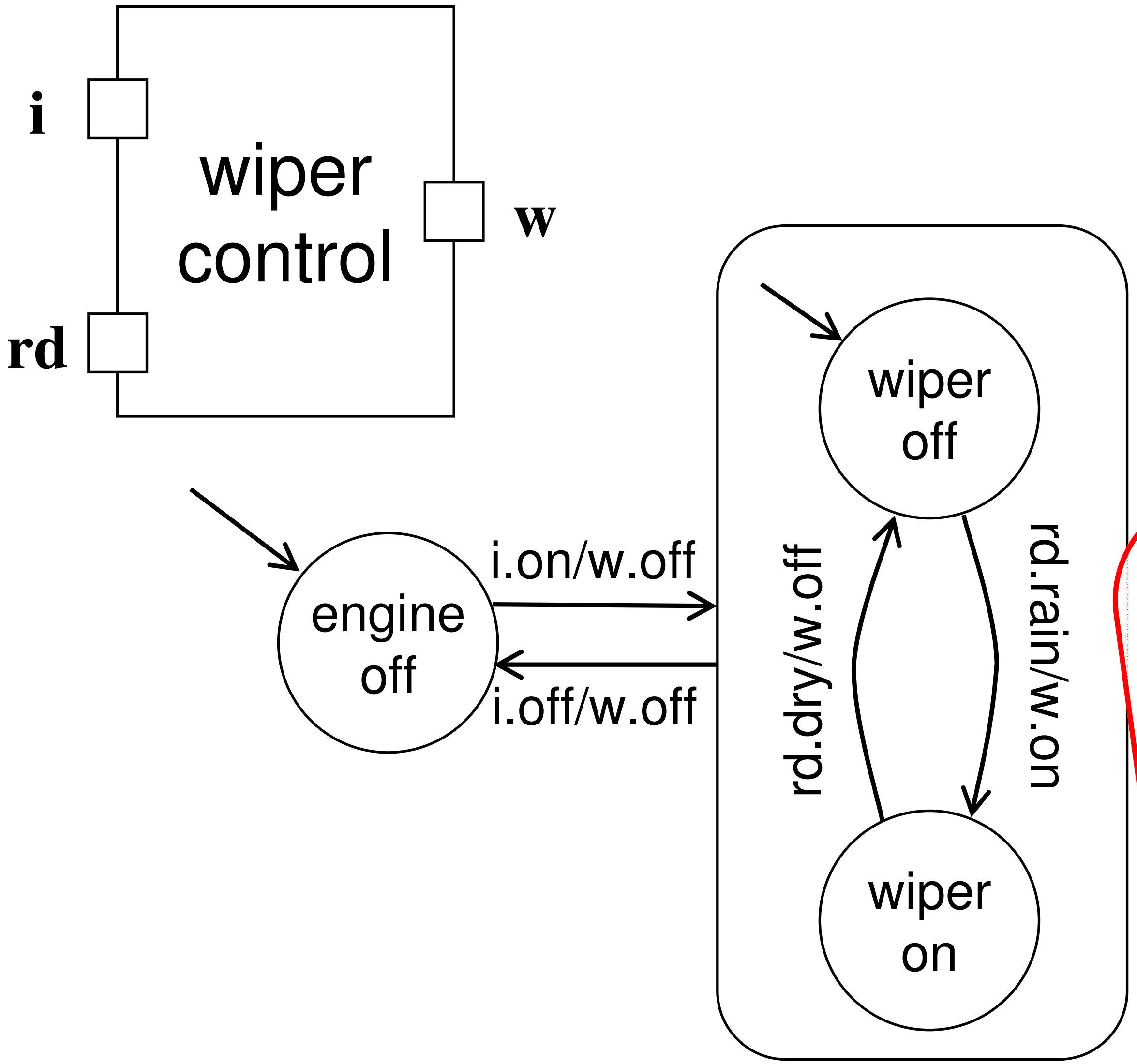
I/O-Automaton defines the behaviour of the component: Which message is sent in response to incoming messages.

An Example



One automaton for each component (plus structure) defines the complete behaviour of our “wiper system”.

Behaviour of wiper control



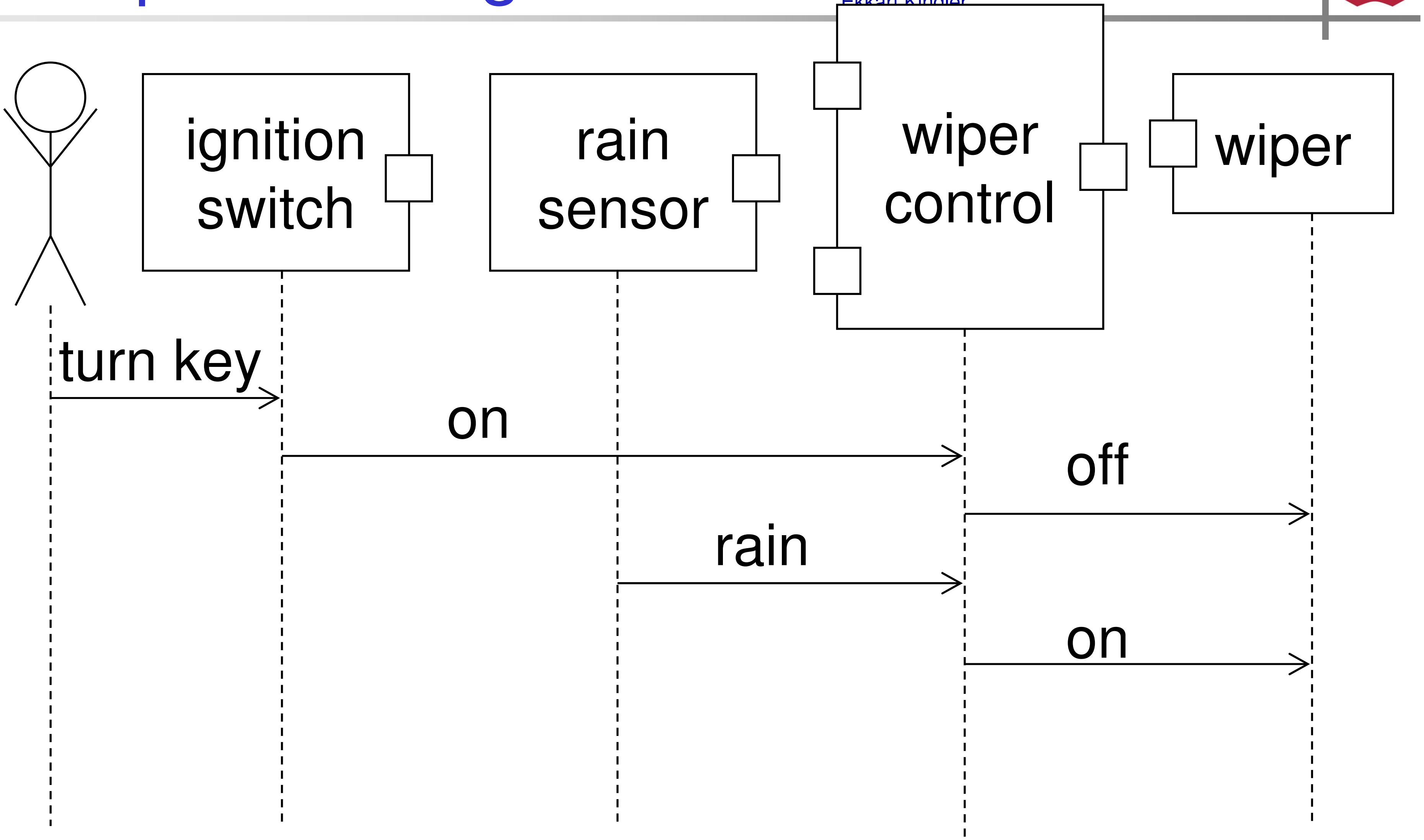
Complex state

More complex form of automaton: StateCharts (in UML called “state diagram”).

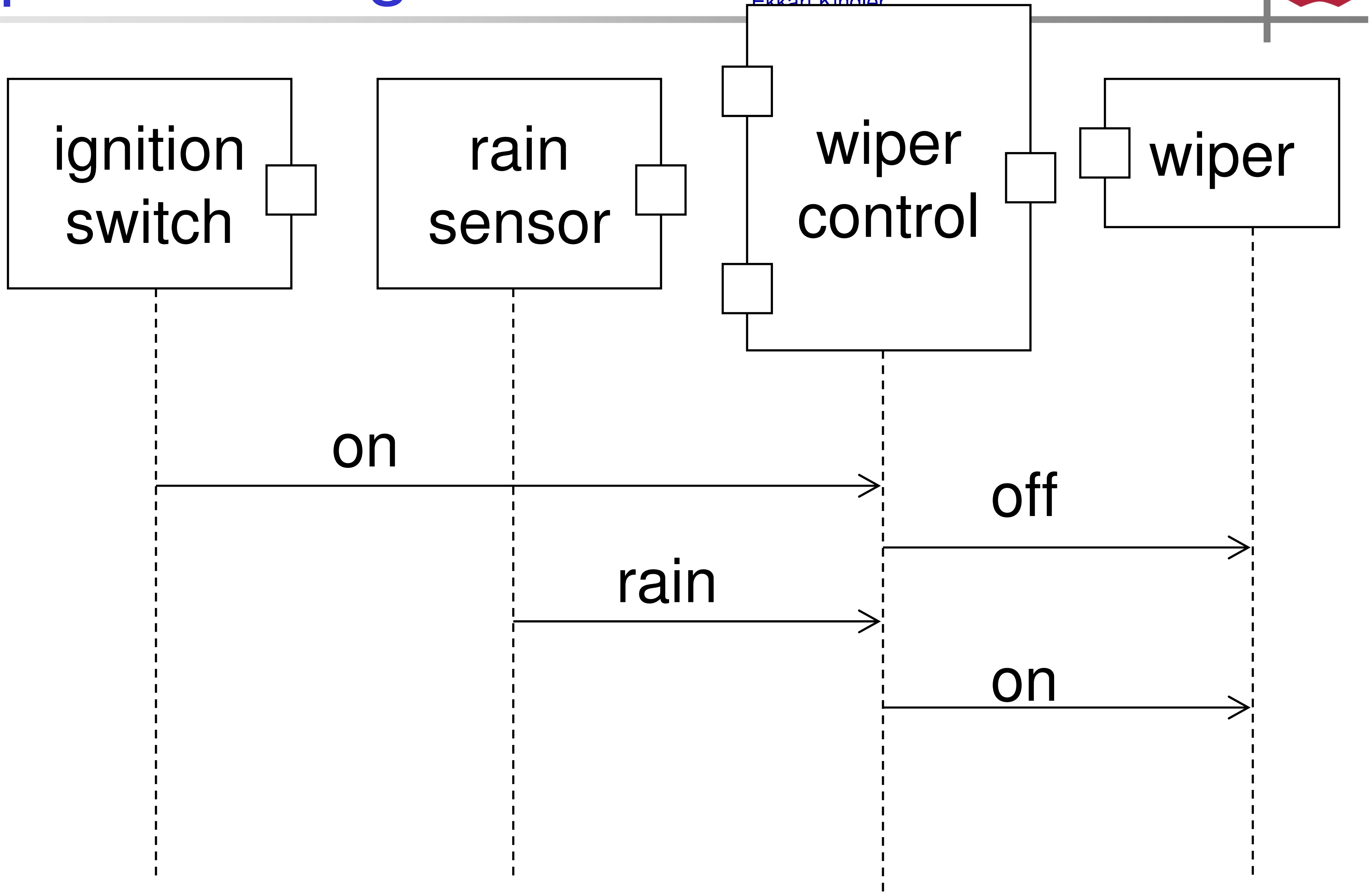
More convenient models! Many more subtleties!

Here, we do not go into the details!

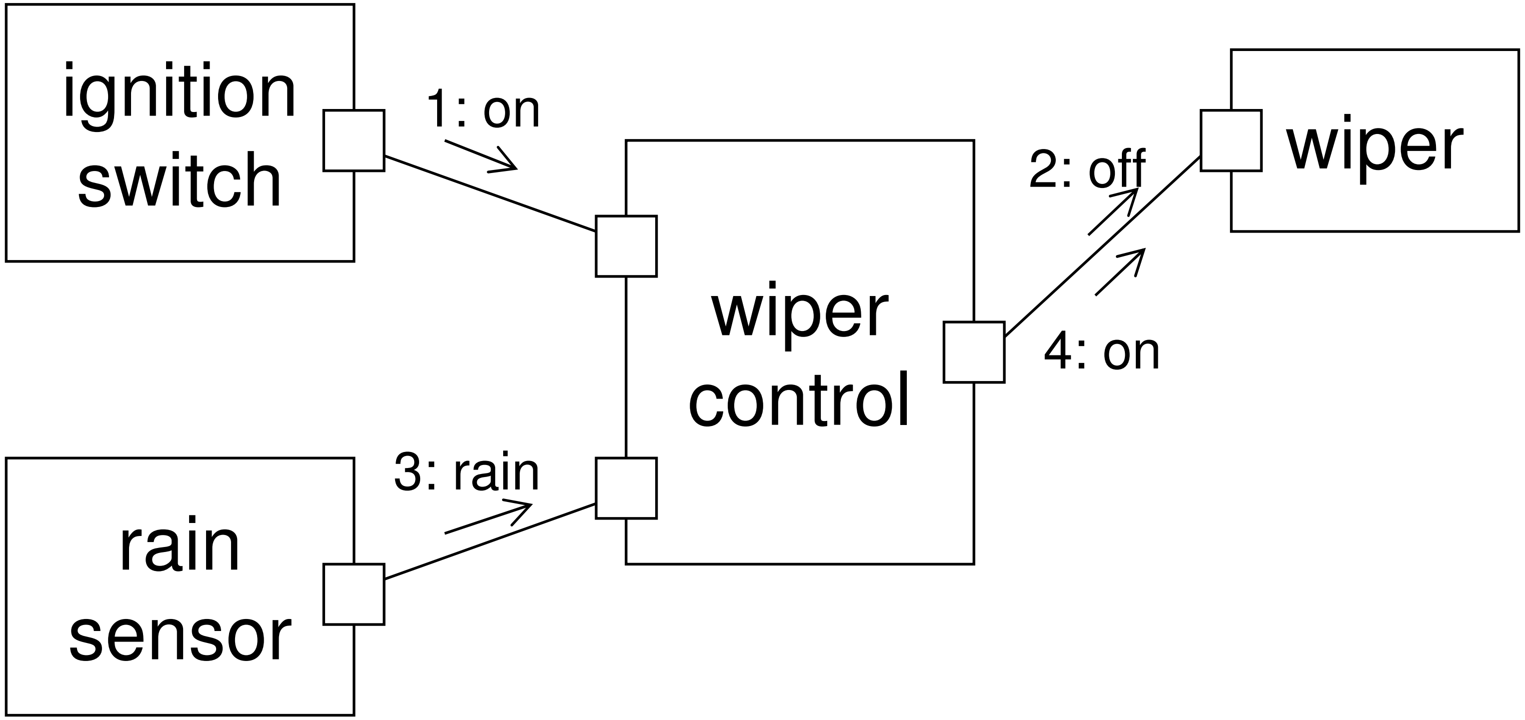
Sequence Diagram



Sequence Diagram



Communication Diagram



Question

- Many different notations for modelling behaviour:
  - Automata / StateCharts
  - Sequence Diagrams
  - Communication Diagrams
  - Activity Diagrams
  - …
- Do they do the same?
- **What is the best?**

This, actually, is a stupid question! **Why?**

Observations

- Automata define the behaviour of a single component or object (in interaction with others): **Intra-object behaviour**
- One automaton define the complete behaviour of a component / object
- Together with the structure, the behaviour is fully defined

→ Coming up with all the automata is quite some work

Observations

- Sequence diagrams and communication diagrams are just different graphical representation of the same thing:
  in UML **interaction diagrams**
- The choice between them is mostly a matter of taste

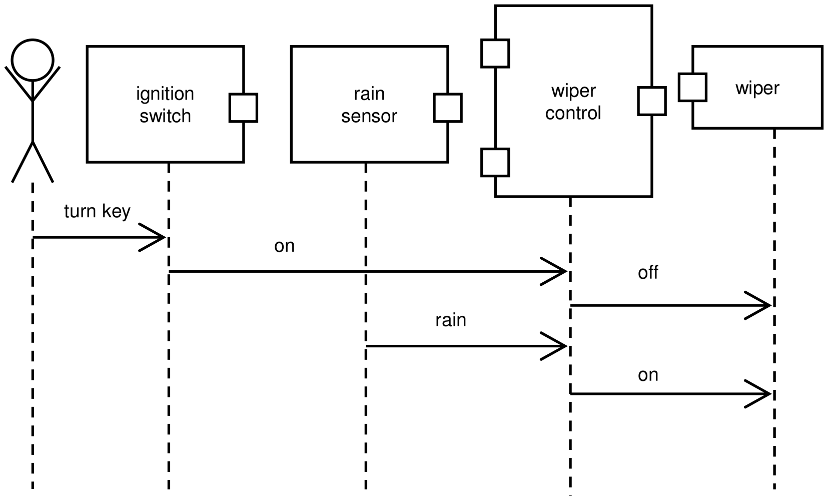
Observations

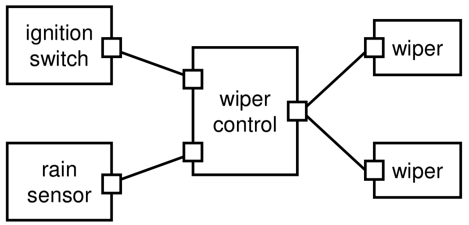
- An interaction diagram defines just one possible behaviour
- Only several interaction diagrams together will define the full behaviour (when did we provide enough of them?)
- Interaction diagrams define the interaction between different components or objects: **Inter-object behaviour**

→Interaction diagrams are good for specifying expected behaviour (also non-expected behaviour) and protocols

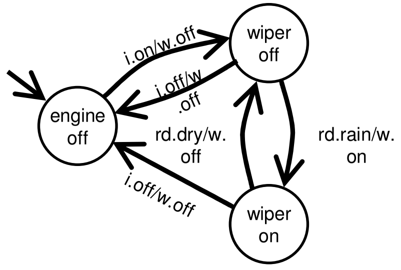
→This can later be “implemented” by automata for the components

Overview





structure



behaviour

Requirements → System model

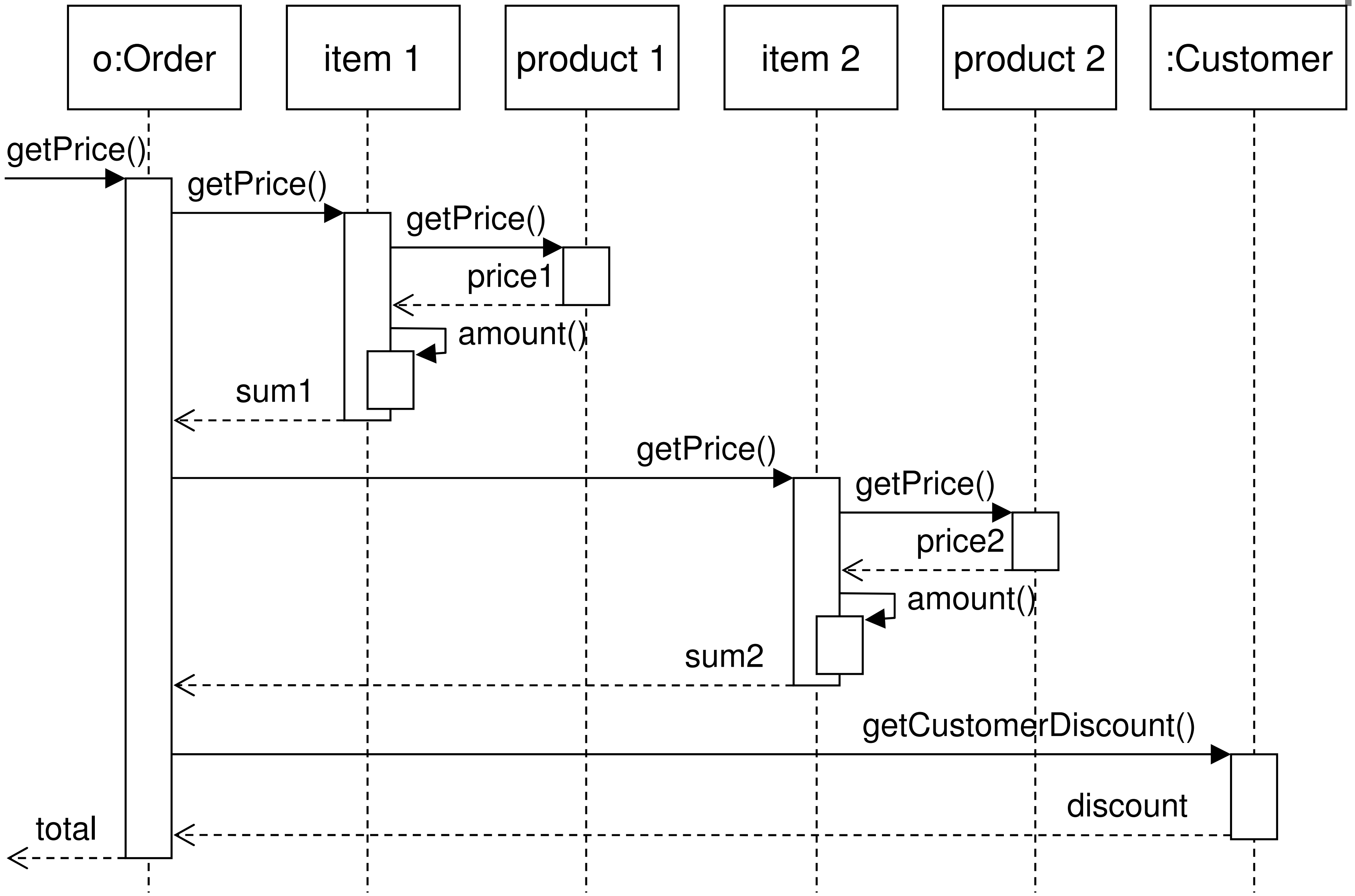
2. Sequence Diagrams

In the rest of today's lecture:
UML 2.x interaction diagrams
(in "sequence diagram notation")

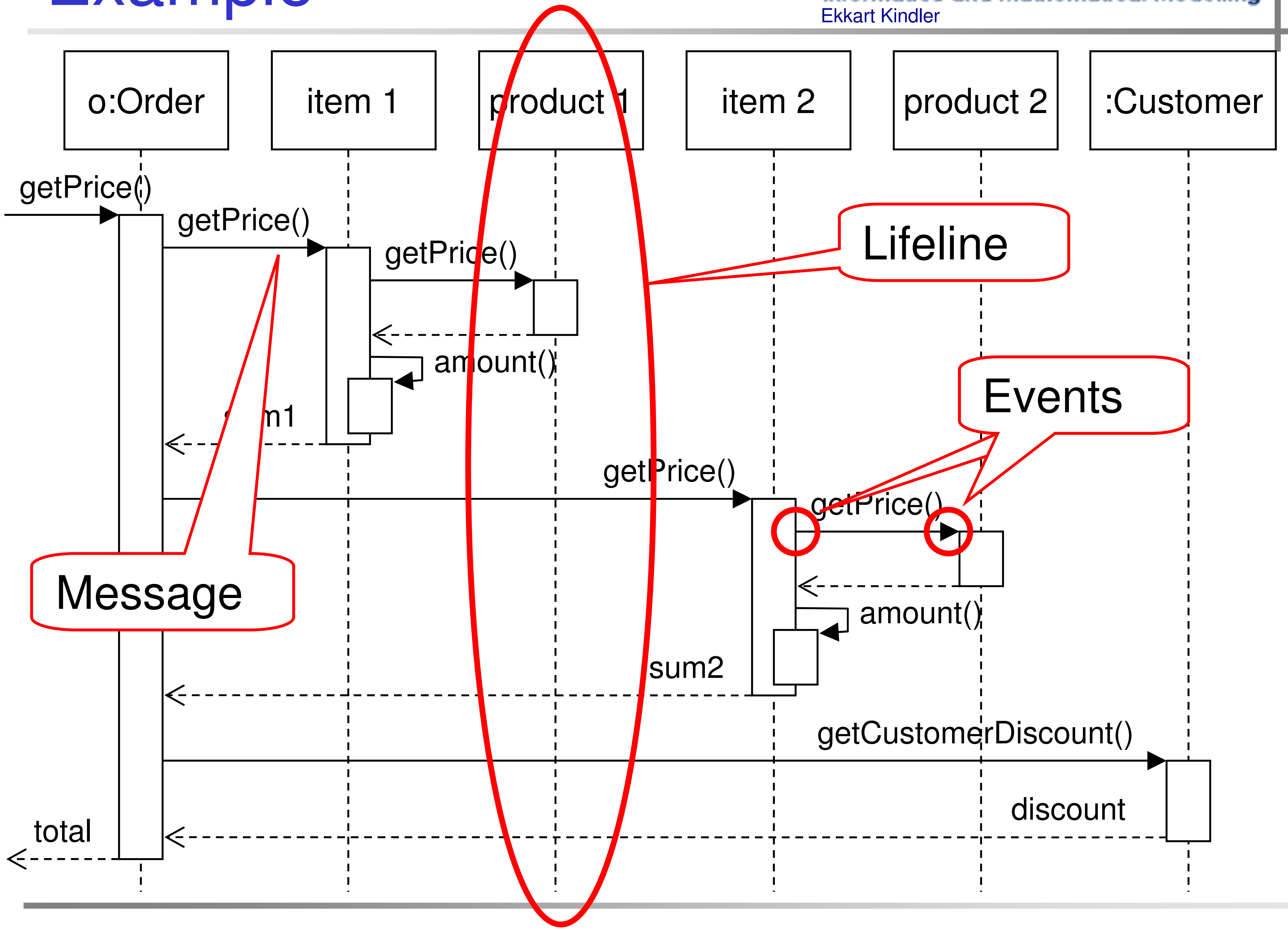
Concepts

- Lifelines (roles / instances)
- Messages
- Calls, returns & asynchronous messages
- Activation

Example


o:Order
item 1
product 1
item 2
product 2
:Customer
getPrice()
getPrice()
getPrice()
price1
amount()
sum1
getPrice()
getPrice()
price2
amount()
sum2
getCustomerDiscount()
discount
total

Example


o:Order
item 1
product 1
item 2
product 2
:Customer
getPrice()
getPrice()
getPrice()
amount()
m1
getPrice()
getPrice()
amount()
sum2
getCustomerDiscount()
discount
total
Lifeline
Events
Message

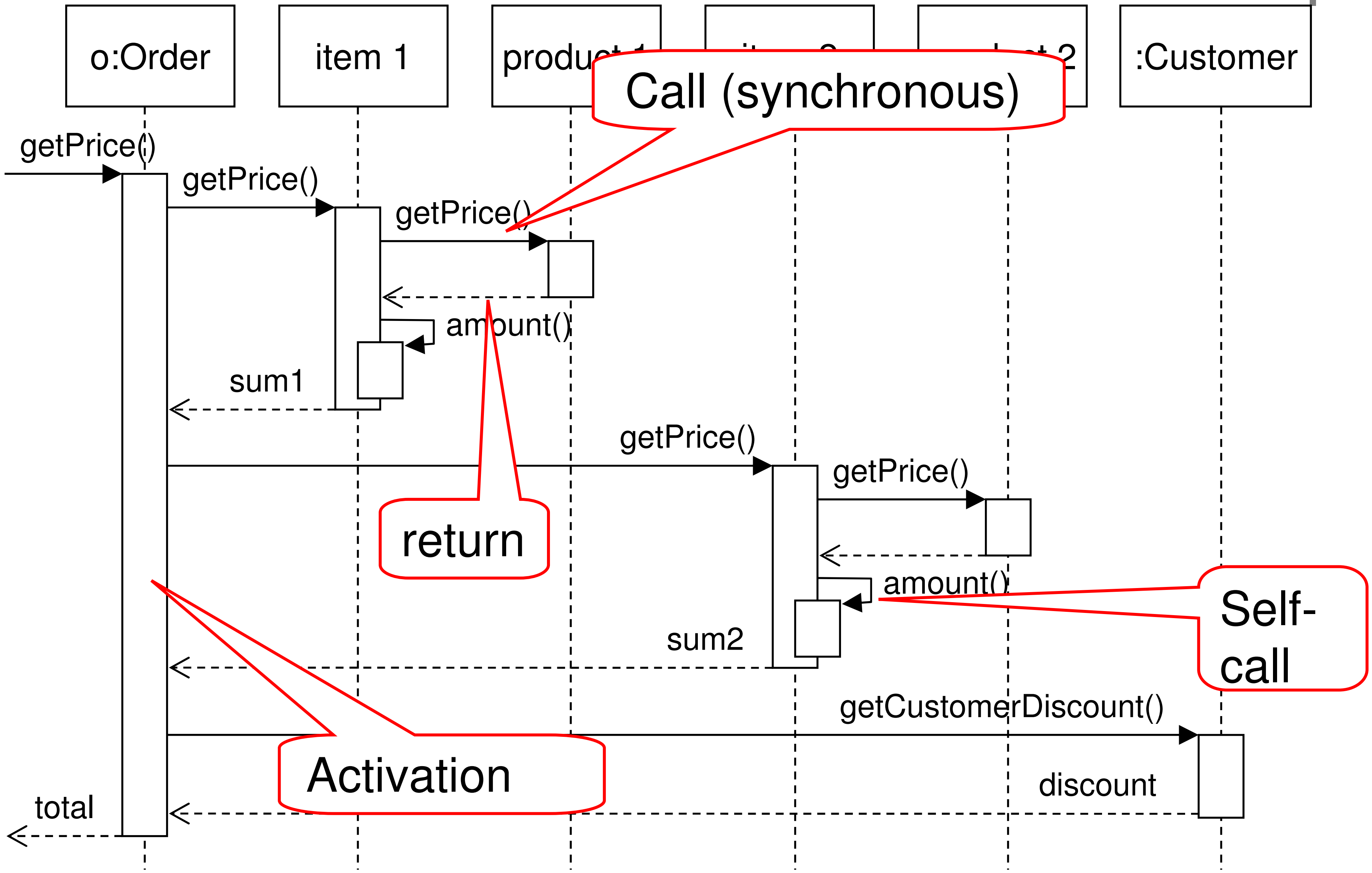
Concepts

- A lifelines represent one participant in an interaction (in UML 1.x: objects, in UML 2.x roles)
- The roles have names of the form

  name : Class

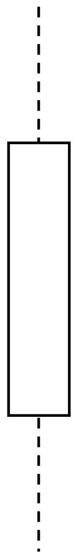
  both parts are optional
- The lifeline represents the (part of the) life of the participant and its interactions
- A messages connects two lifelines;
  the end points are events; the name of the message refers to the behaviour (method of a class)

Example

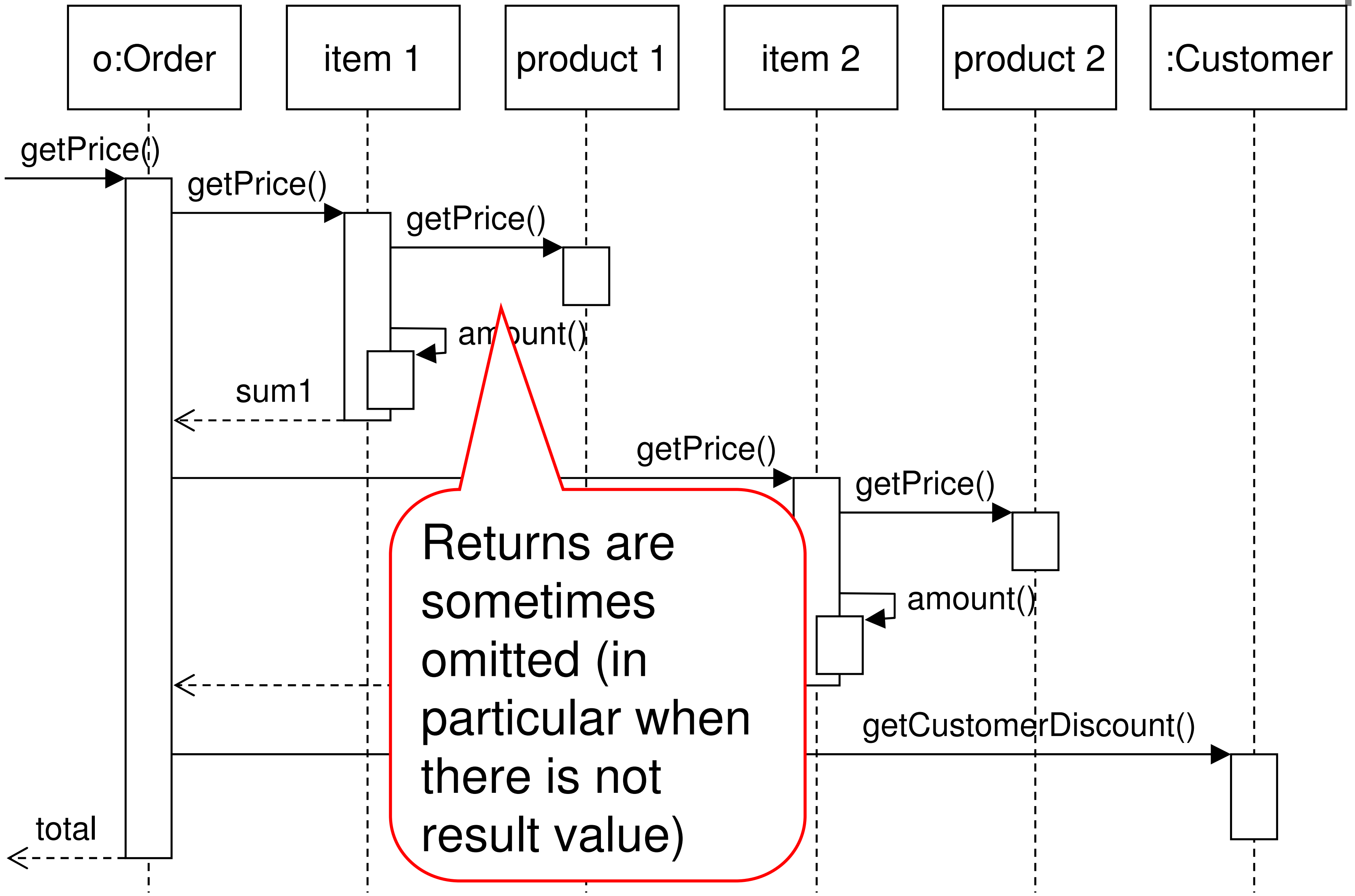

o:Order
item 1
produ
:Customer
Call (synchronous)
getPrice()
getPrice()
getPrice()
amount()
return
sum1
getPrice()
getPrice()
amount()
Self-call
sum2
getCustomerDiscount()
Activation
discount
total

Concepts

- Messages can be synchronous:
  call (→)and return (⇠)
- Messages can be asynchronous (see wiper exmpl.): →
- The activation (optional) indicates the span at which a method call is active in a participant
  (technically: there is a frame on the stack for this method)
- For self-calls, activations “pile up”



Example

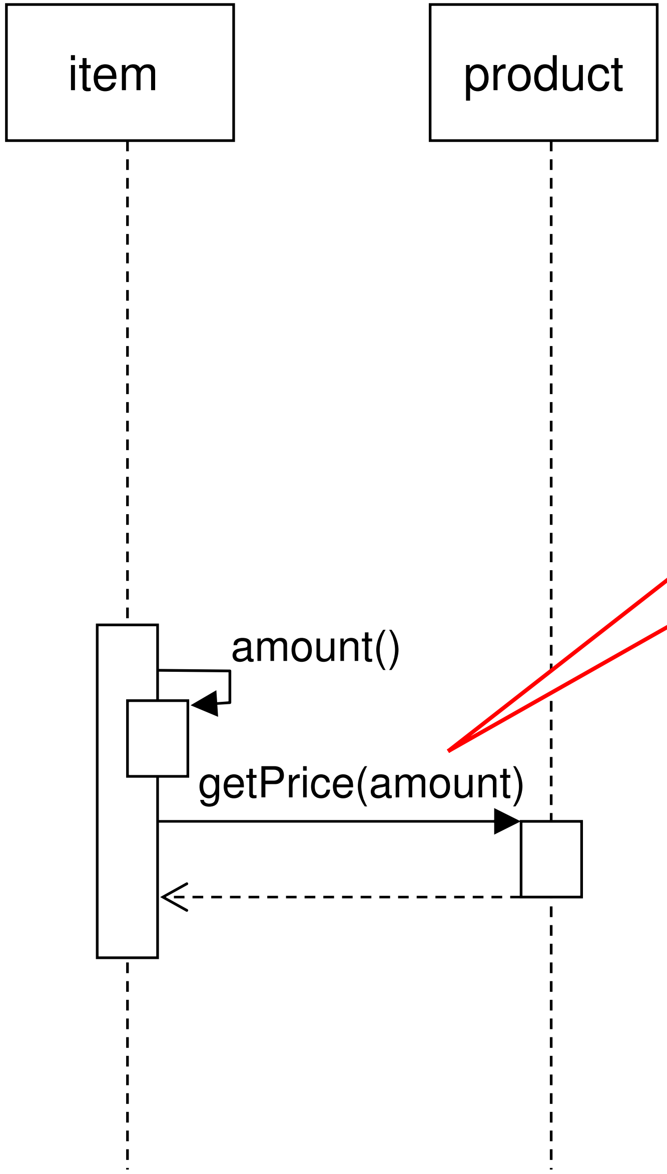


Returns are sometimes omitted (in particular when there is not result value)

Concepts

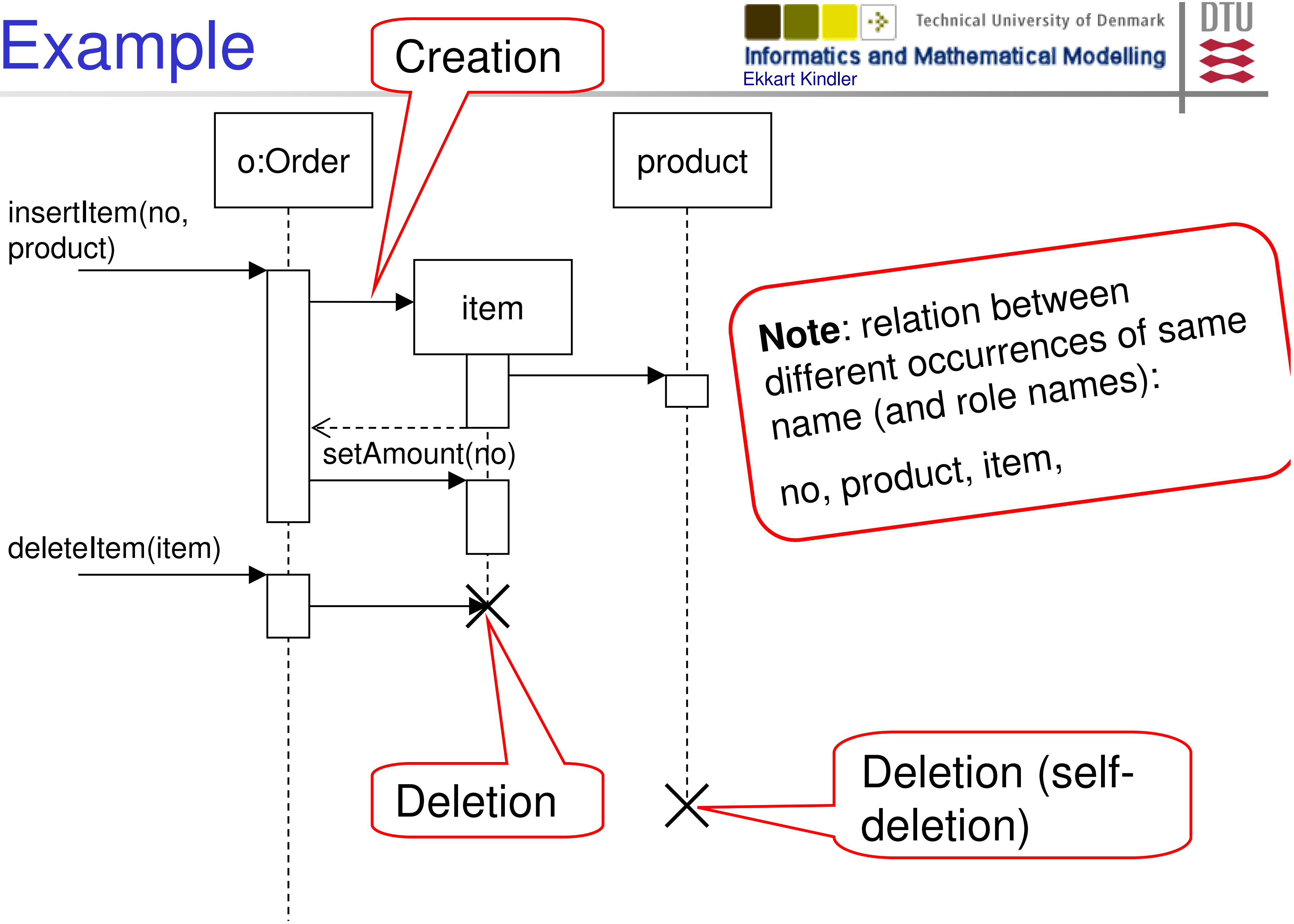
- Parameters
- Creation and deletion of objects
- Found and lost messages
- Ordering

Example



Methods may have parameters (from return values or local attributes)

Example



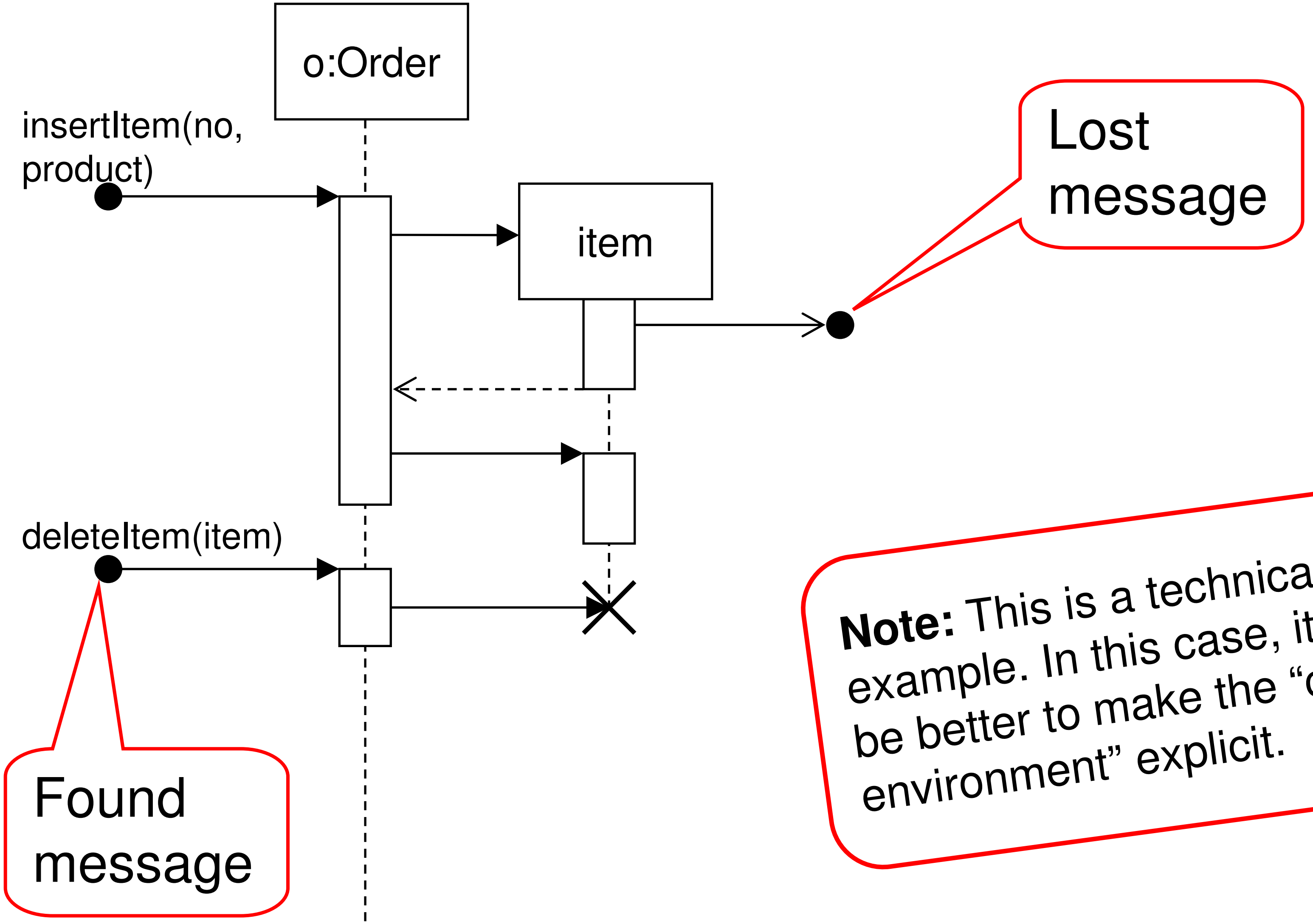
Note: relation between different occurrences of same name (and role names):

no, product, item,

Concepts

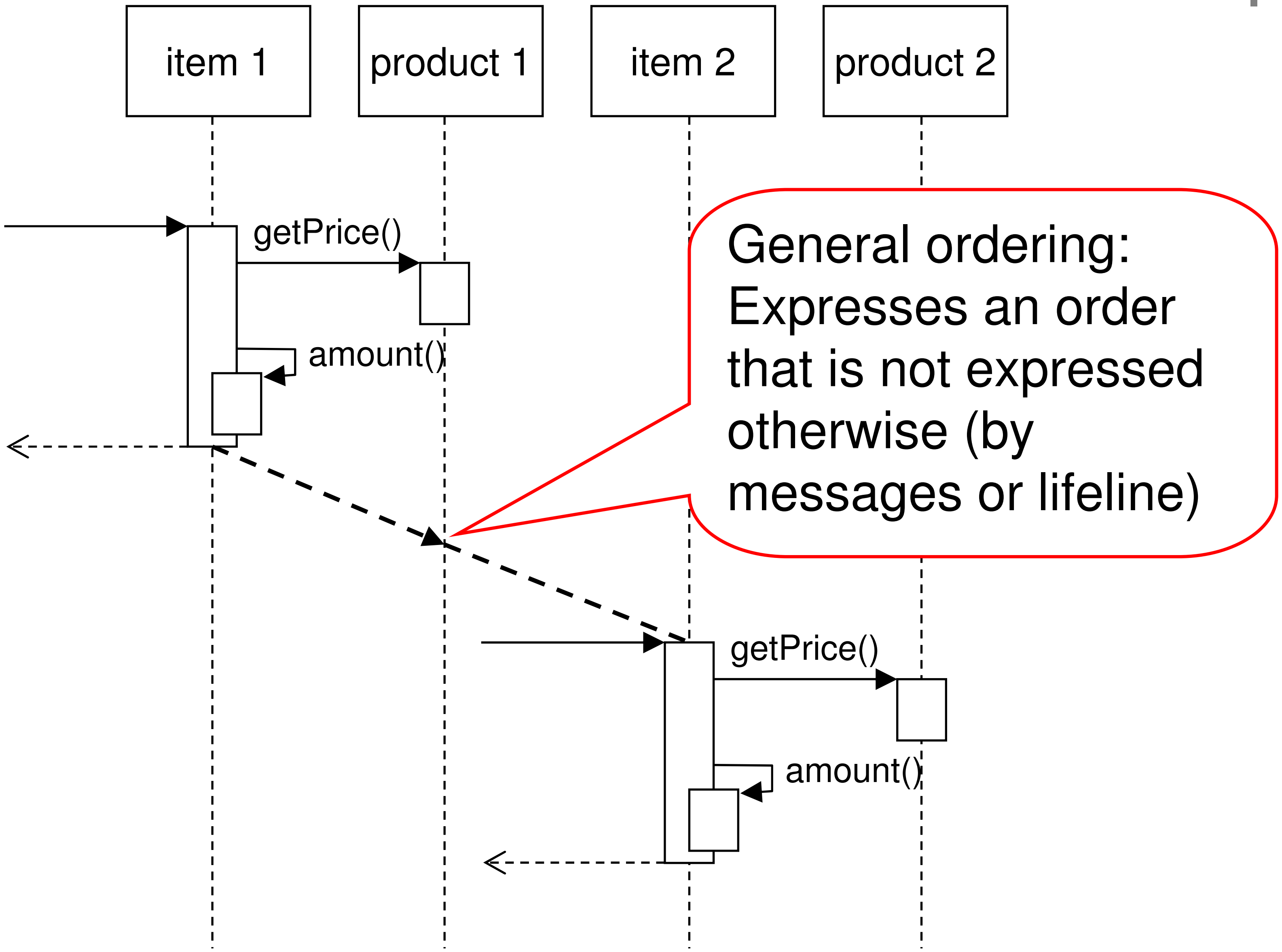
- Normally, message start and end at some event (or gate)
- Messages, that come from nowhere are called found messages
- Messages, that end nowhere are called lost messages

Example



Note: This is a technical example. In this case, it would be better to make the “calling environment” explicit.

Example

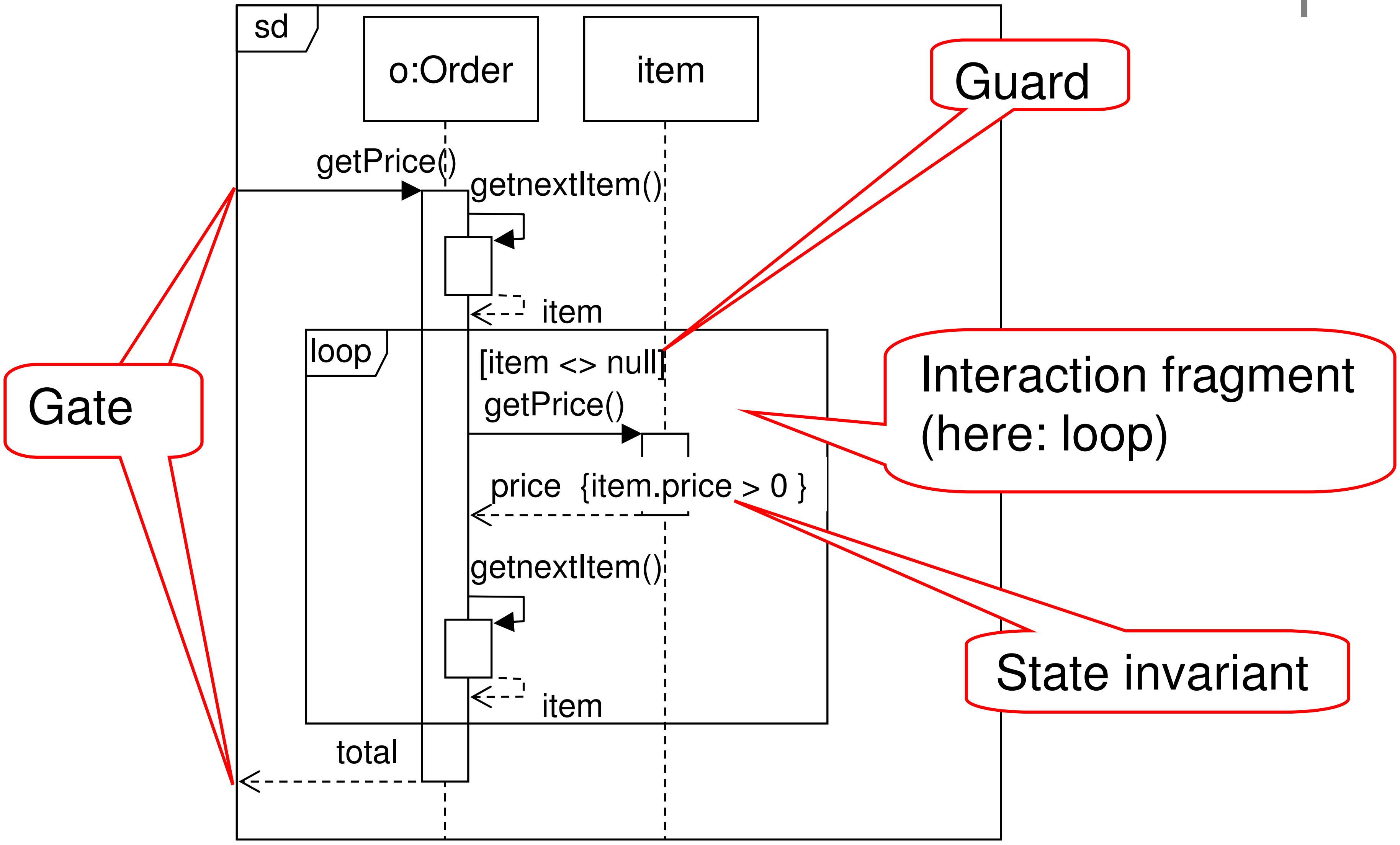


General ordering: Expresses an order that is not expressed otherwise (by messages or lifeline)

Concepts

- Interaction fragments
- State Invariants
- Continuations
- Co-regions

Example



Guard

Interaction fragment (here: loop)

State invariant

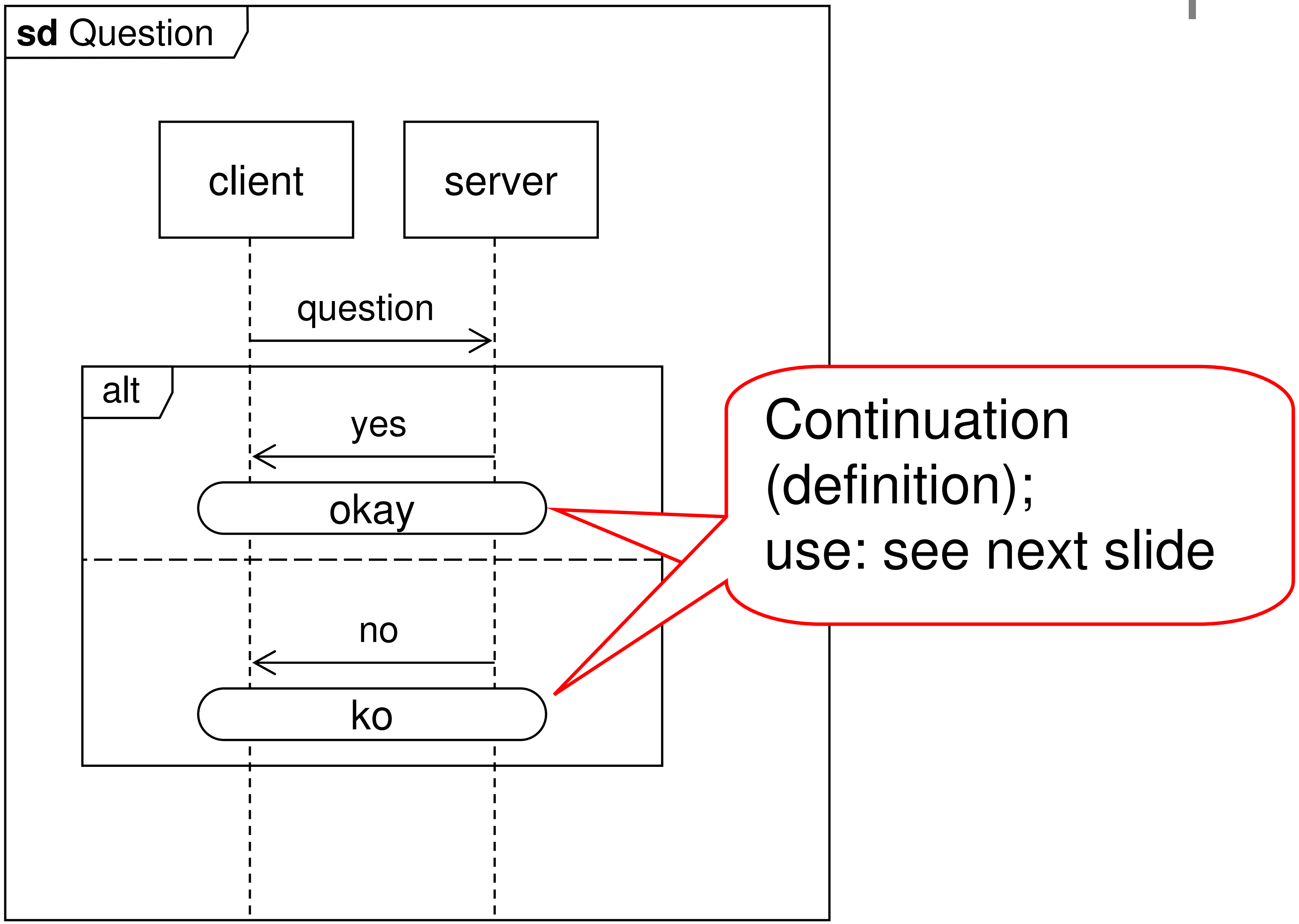
Gate

Concepts

- Interaction fragments
  - sd (surrounds the complete **s**equence **d**iagram)
  - loop (iteration)
  - alt (choice / if then else)
  - opt (optional / if then)
  - par (fragment operands run in parallel)
  - ref (reference to another definition)
  - …

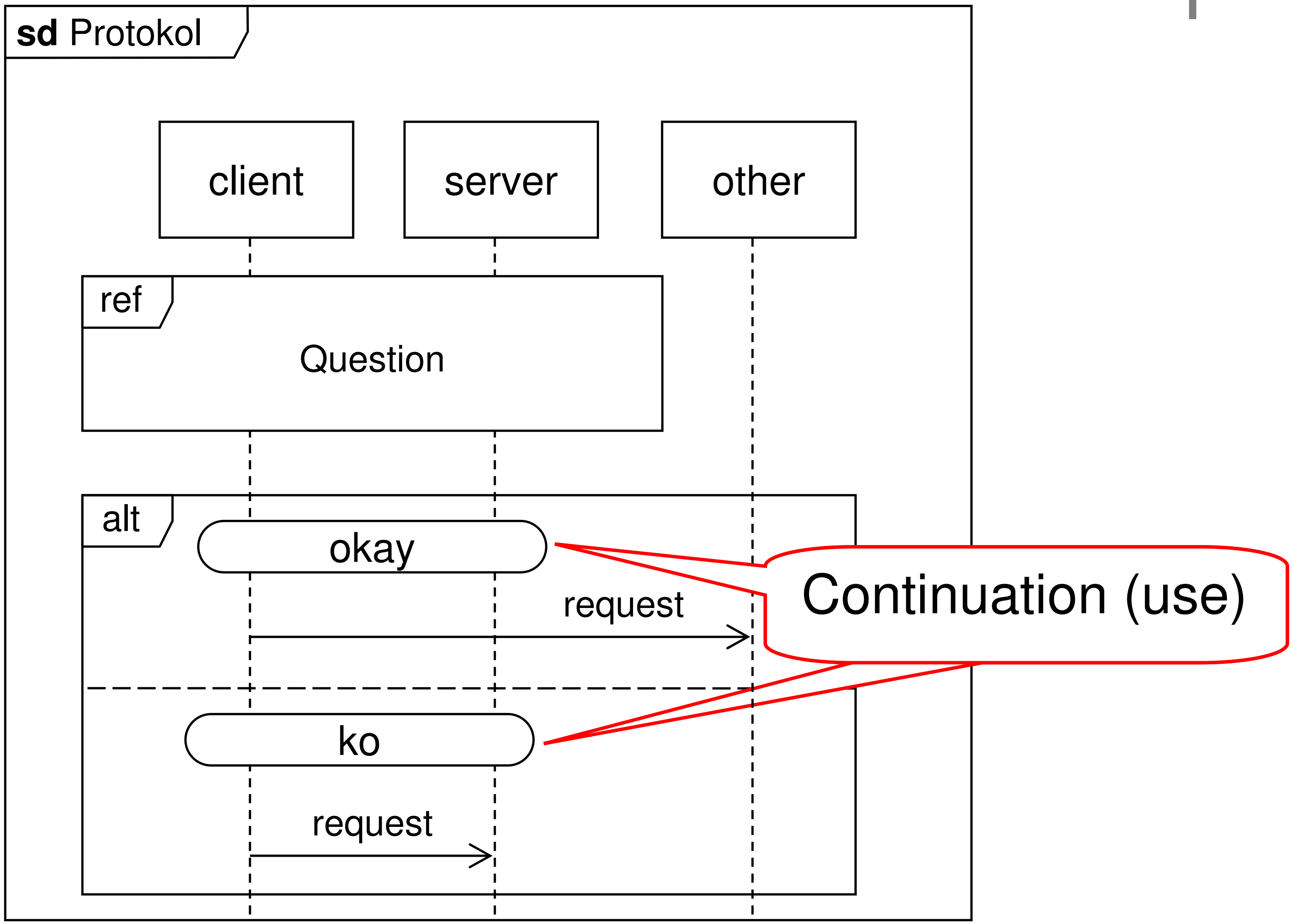
Interaction fragments have much modelling power, but tend to be “programming” and not what sequence diagrams were originally made for or really good at!

Example



Continuation
(definition);
use: see next slide

Example


sd Protokol
client
server
other
ref
Question
alt
okay
request
ko
request
Continuation (use)

UML Behaviour Diagrams

- Use Cases
- Interaction Diagrams
  - Sequence Diagrams
  - Communication Diagrams
- Activity Diagrams
- State Machines (StateCharts)