Software Engineering 1
Special lecture: Modelling Behaviour

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

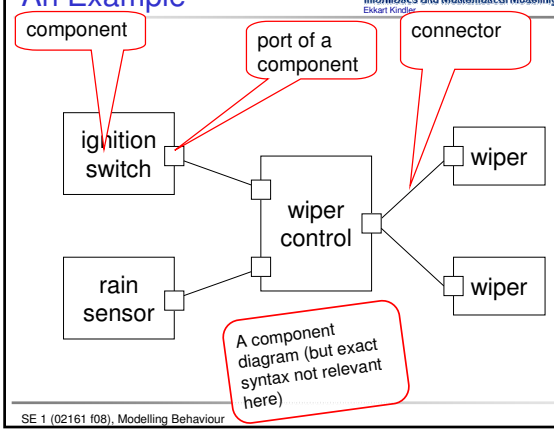
1. Motivation and Idea

Motivation

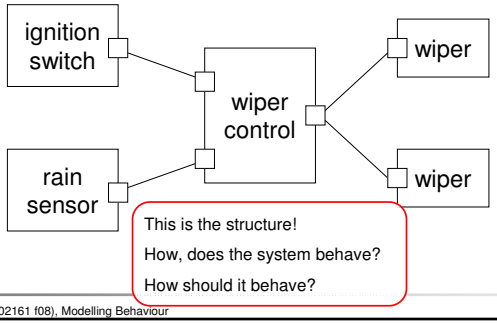
- In this course **up to now**:
  Mainly structural models: class diagrams, object diagrams, package diagrams of software.
- **Now**: Modelling what the software actually should do: its functionality and behaviour.

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

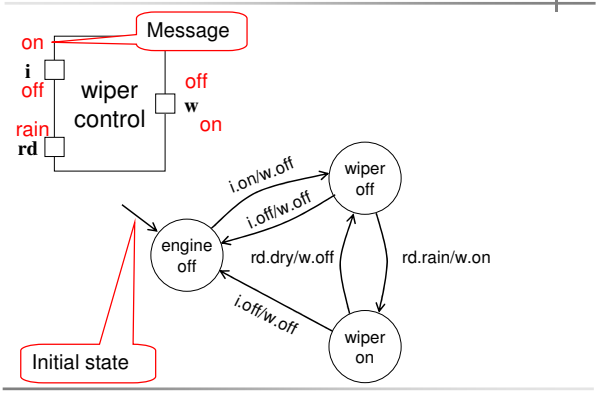
An Example



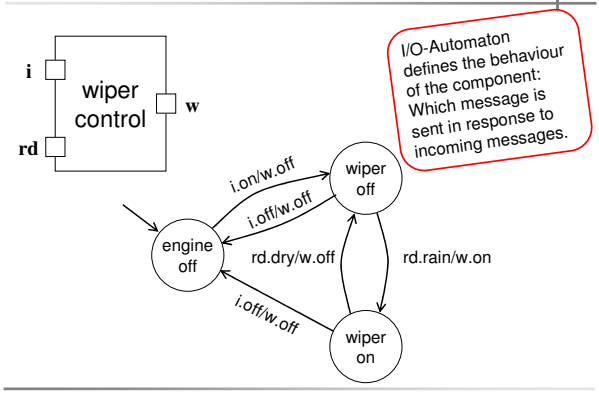
An Example



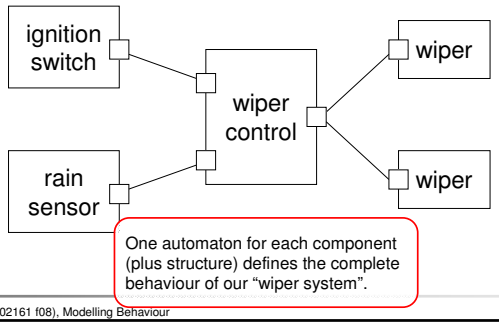
Behaviour of wiper control



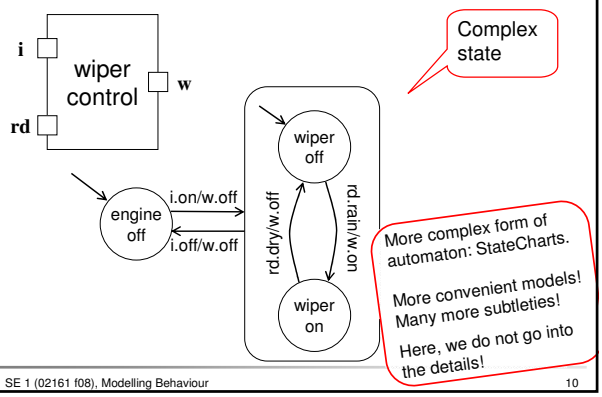
Behaviour of wiper control



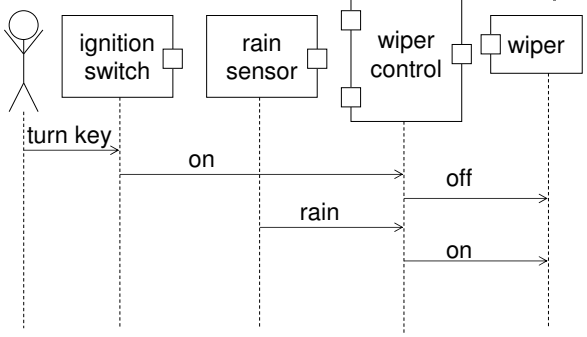
An Example



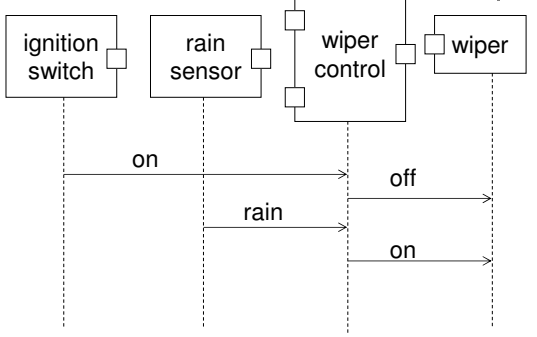
Behaviour of wiper control



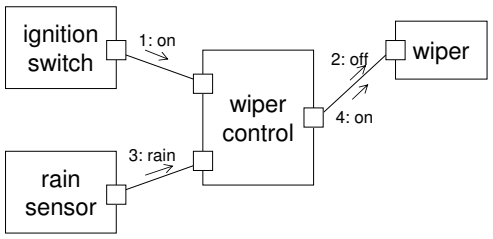
Sequence Diagram



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

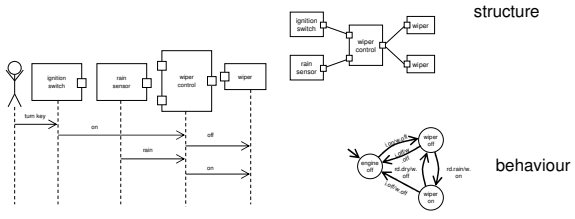
Observations

- 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



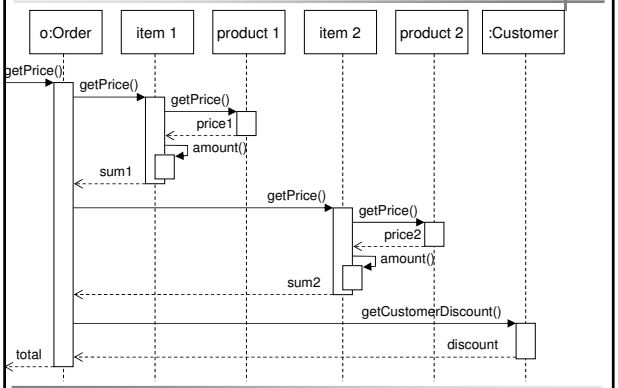
2. Sequence Diagrams

In the rest of today's lecture:
UML 2.0 interaction diagrams (in "sequence diagram notation")

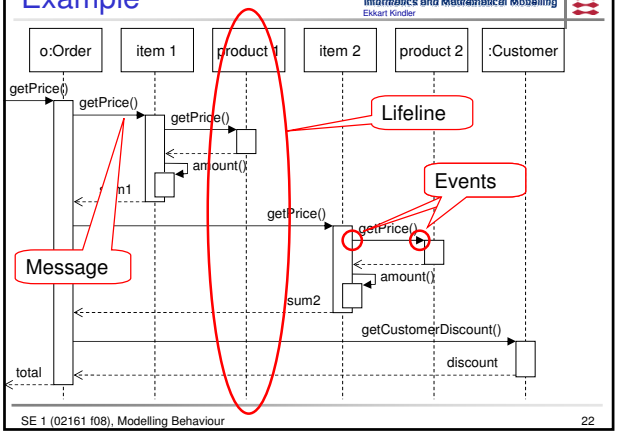
Concepts

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

Example



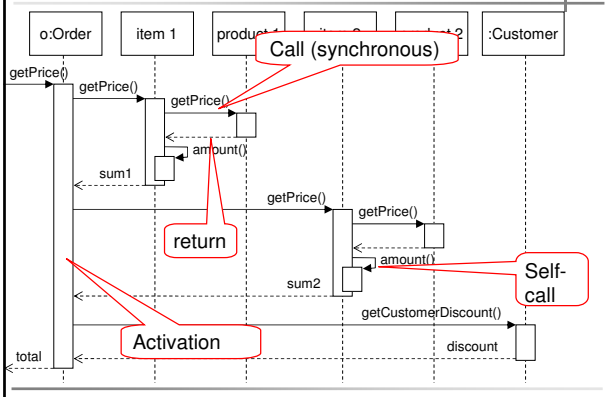
Example



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)

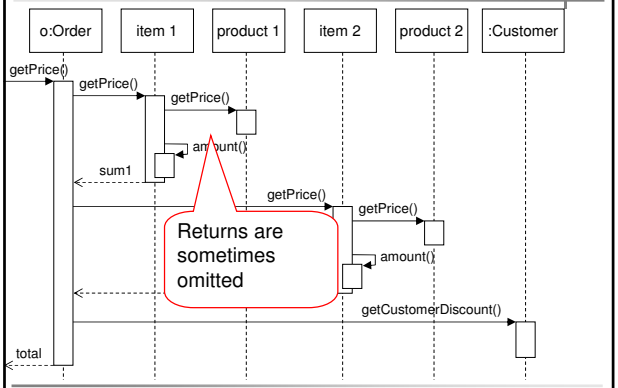
Example



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”

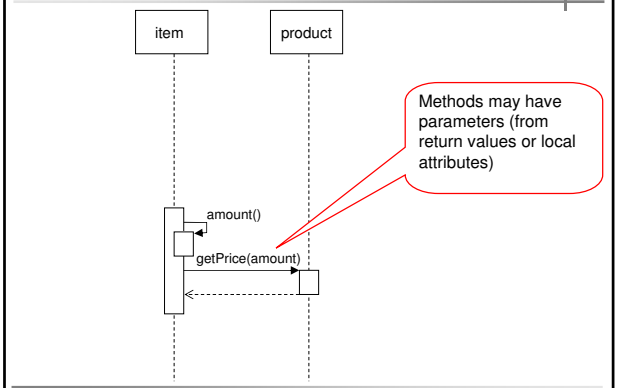
Example



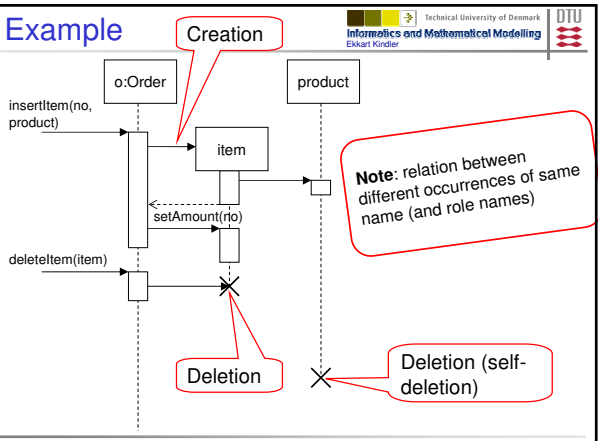
Concepts

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

Example



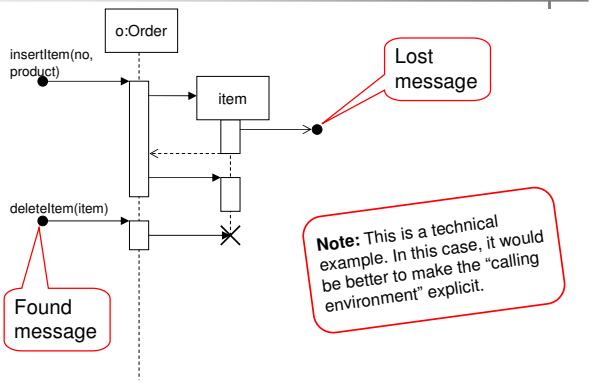
Example



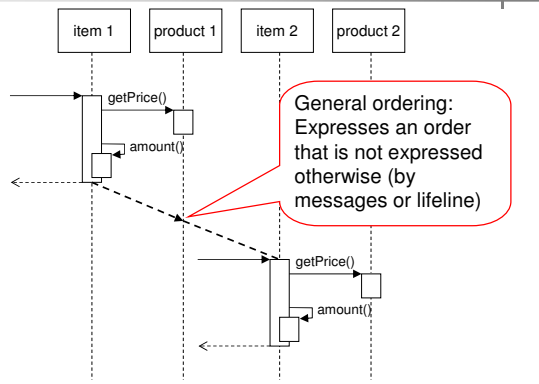
Concepts

- 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



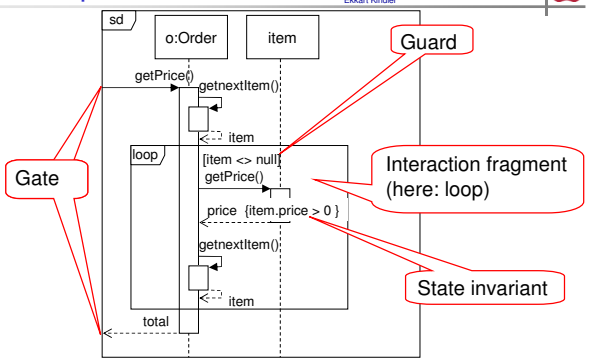
Example



Concepts

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

Example

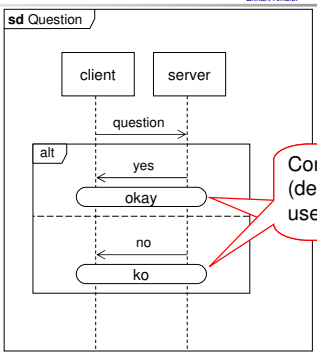


Concepts

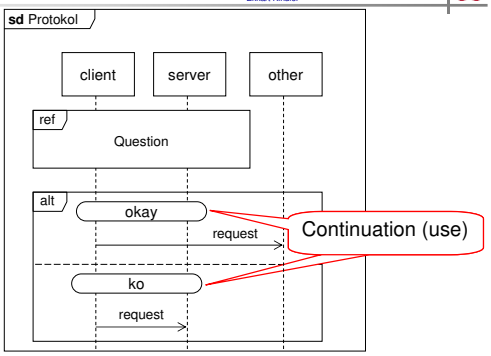
- Interaction fragments
  - sd (surrounds the complete sequence diagram)
  - Loop (iteration)
  - alt (choice / if then else)
  - opt (optional / if then)
  - par (fragment operands run in parallel)
  - ref (reference to another definition)
  - …

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



Example



UML Behaviour Diagrams

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