Modelling III

Sequence Diagrams

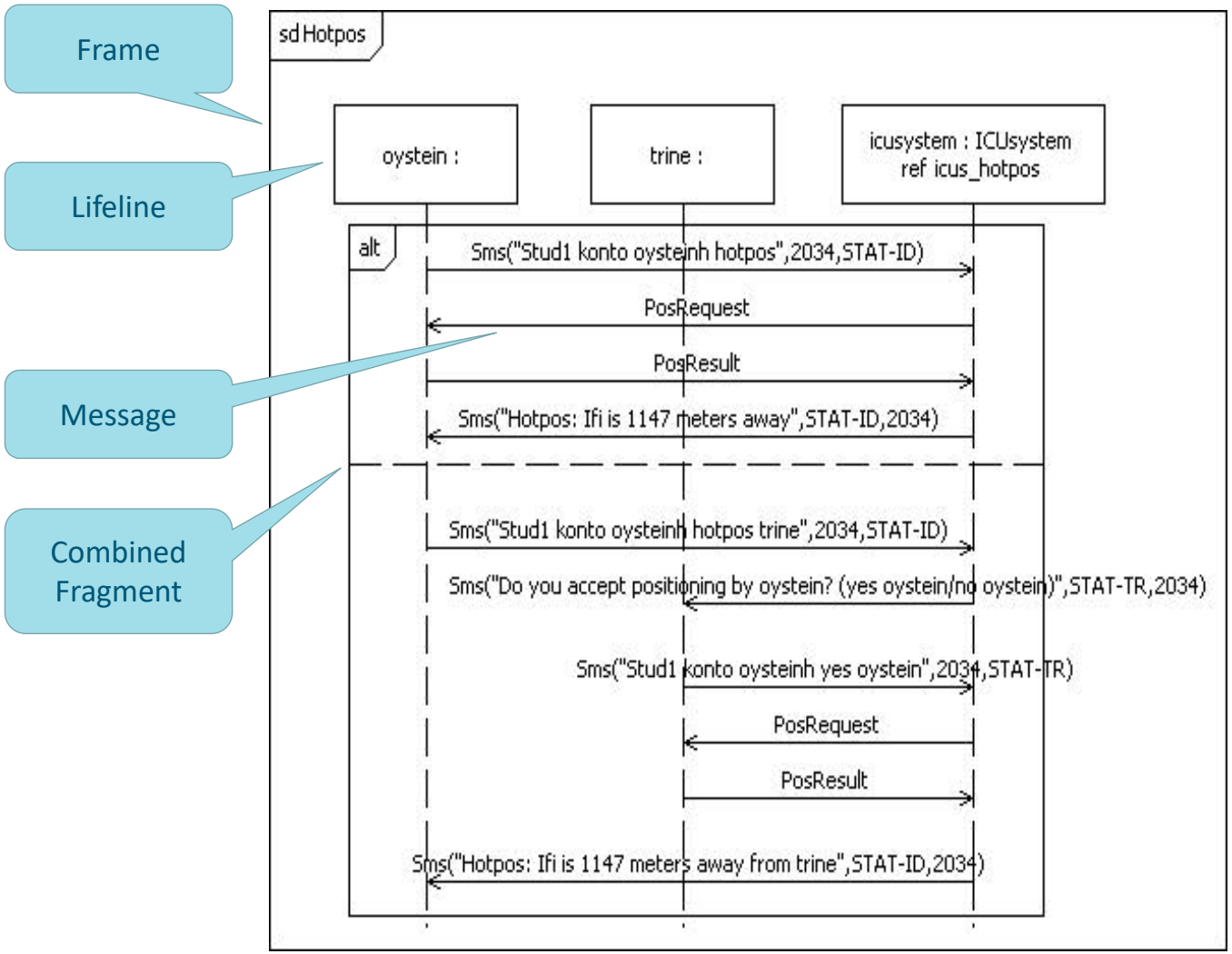
Ketil Stølen

Partly based on slides prepared by Prof. Øystein Haugen, HiØ & SINTEF

Overview of lecture

- Sequence Diagrams
  - What are they for?
  - What is their role in software engineering?
- Basic sequence diagrams
- High level sequence diagrams
  - Structuring mechanisms

This is a Sequence Diagram



Sequence Diagrams in a nutshell

- Sequence Diagrams are
  - simple
  - powerful
  - readable
- Emphasizes the interaction between objects
- Describes often only a small portion of the total behavior
  - improves the individual understanding of an interaction problem

Sequence Diagrams are used to ...

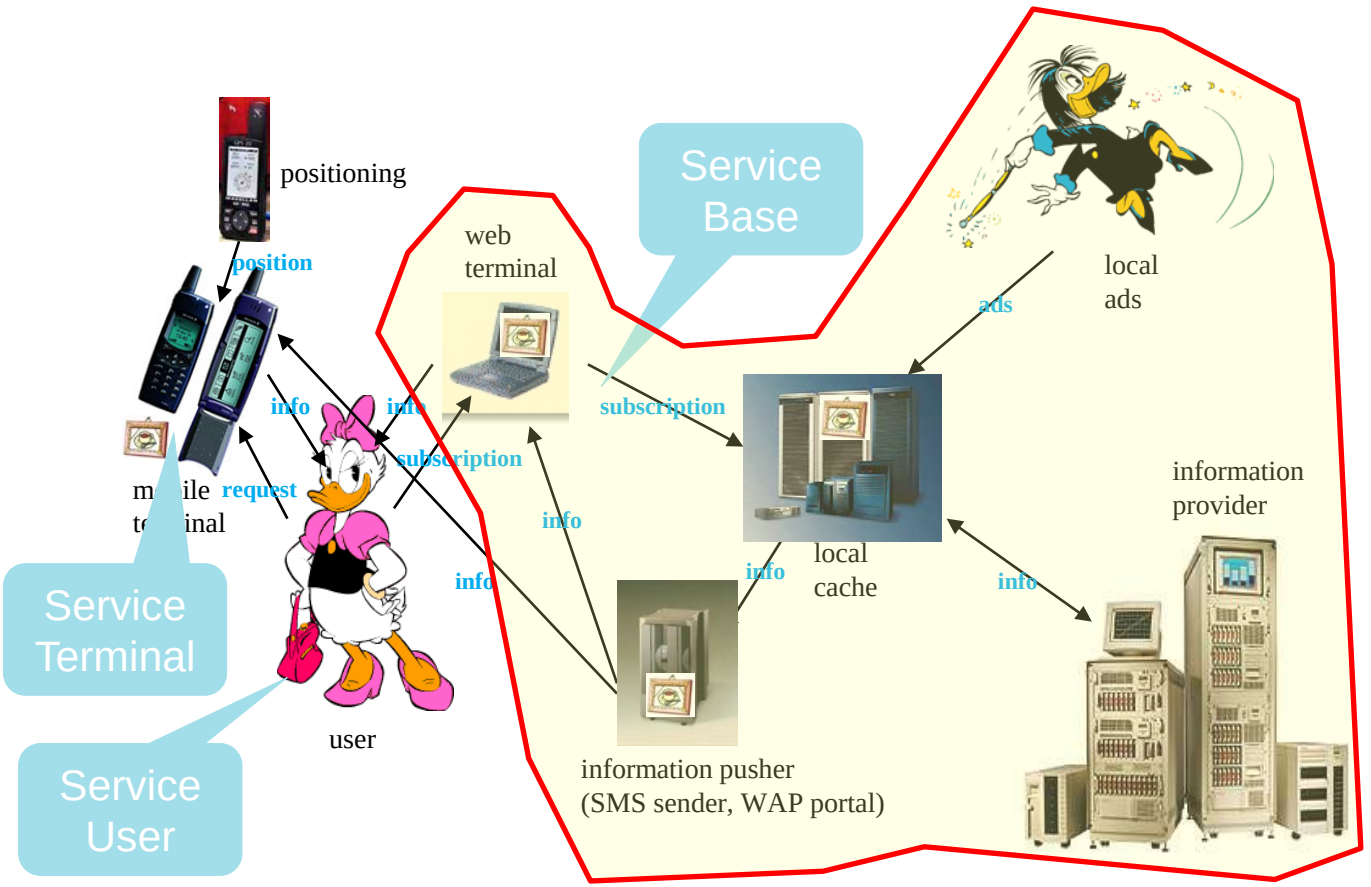
- document protocol situations
- exemplify behavior situations
- verify interaction properties relative to a specification
- describe test cases
- document simulation traces

The example context: Dolly Goes To Town

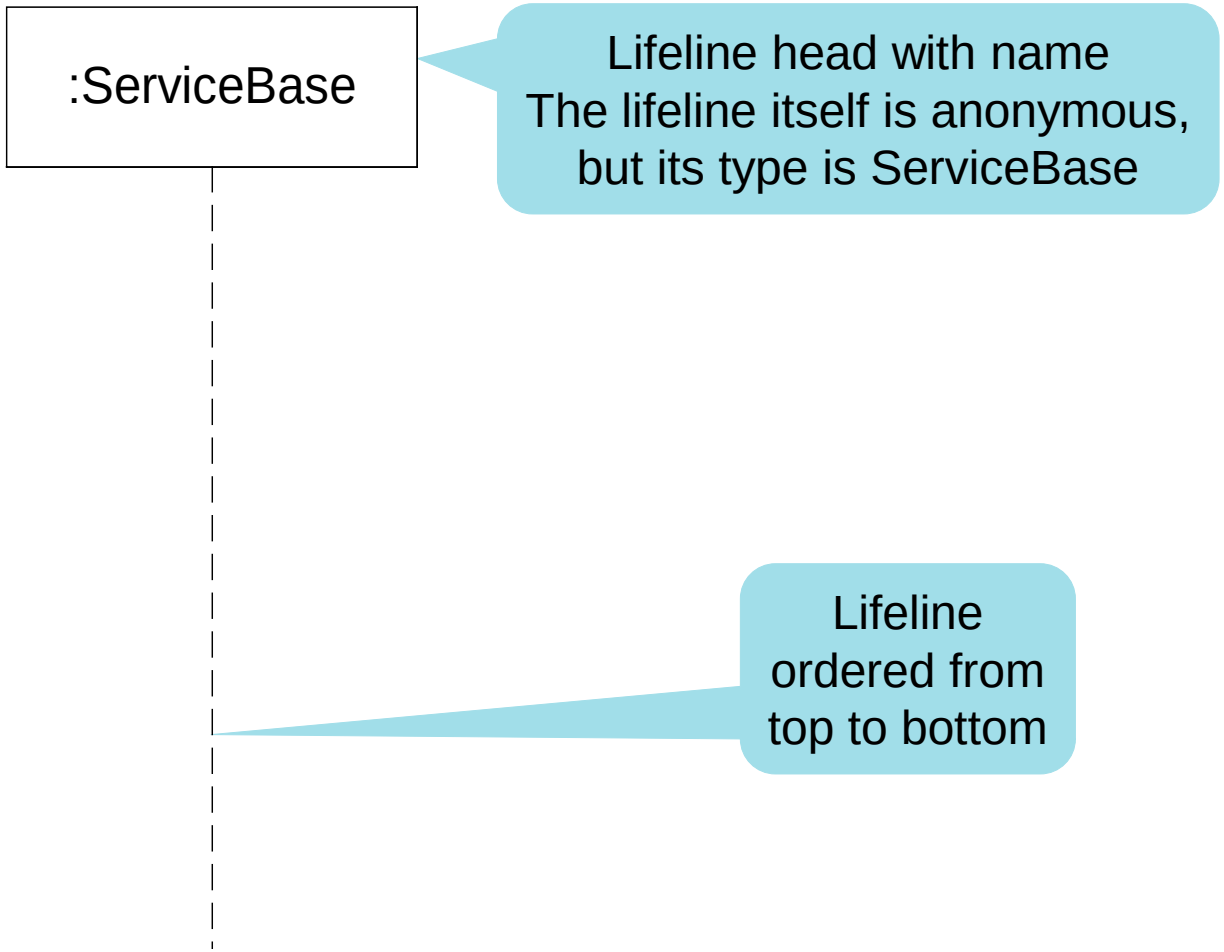
- Dolly is going to town and
  - wants to subscribe for bus schedules back home
  - given her current position, and
  - the time of day

- The service should not come in effect until a given time in the evening

The informal architecture



Lifeline – the “doer”

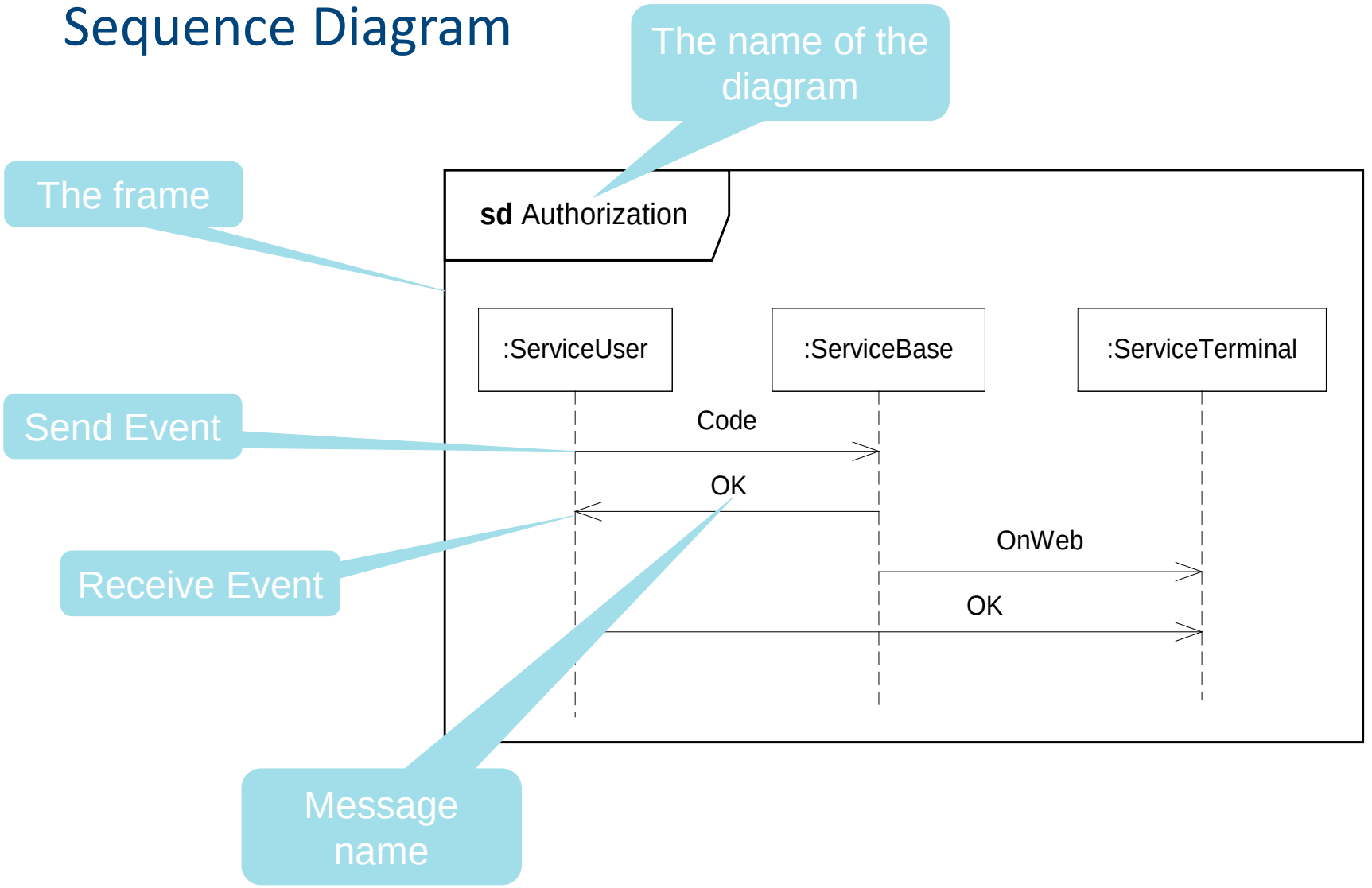


Messages

- Lifelines communicate messages
- A message has one send event and one receive event
- A message is always sent before it can be received
- Events are strictly ordered in time along a lifeline from top to bottom

- For any message name M we use !M to denote its send event and ?M to denote its receive event

Sequence Diagram



Trace

- A trace is a sequence of events ordered in time
- A trace represents a possible execution of a sequence diagram
- A trace may be finite or infinite
- A sequence diagram describes at least one trace
- A sequence diagram may decribe infinitely many traces

Two rules

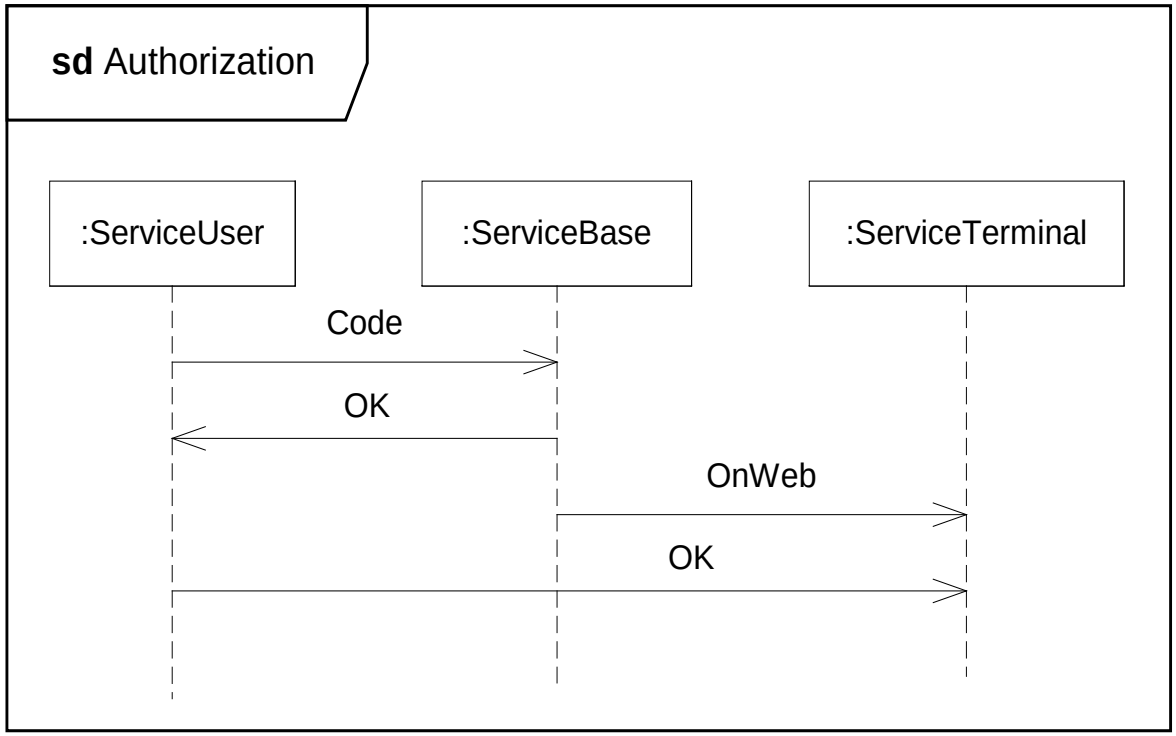
Causality

- a message can never be received before it has been transmitted
- the transmission event for a message is therefore always ordered before the reception event for the same message

Weak sequencing

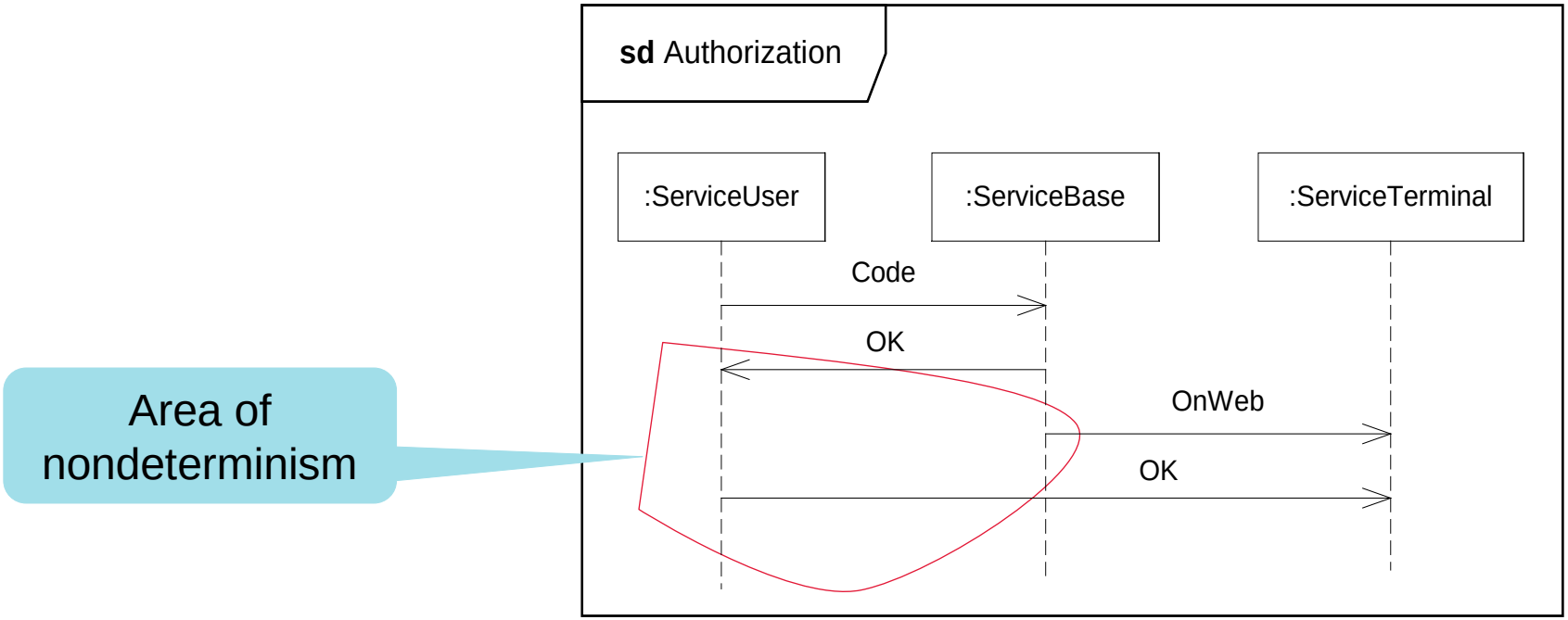
- events from the same lifeline are ordered in the trace in the same order as on the lifeline (from top to bottom)

One possible trace of Authorization

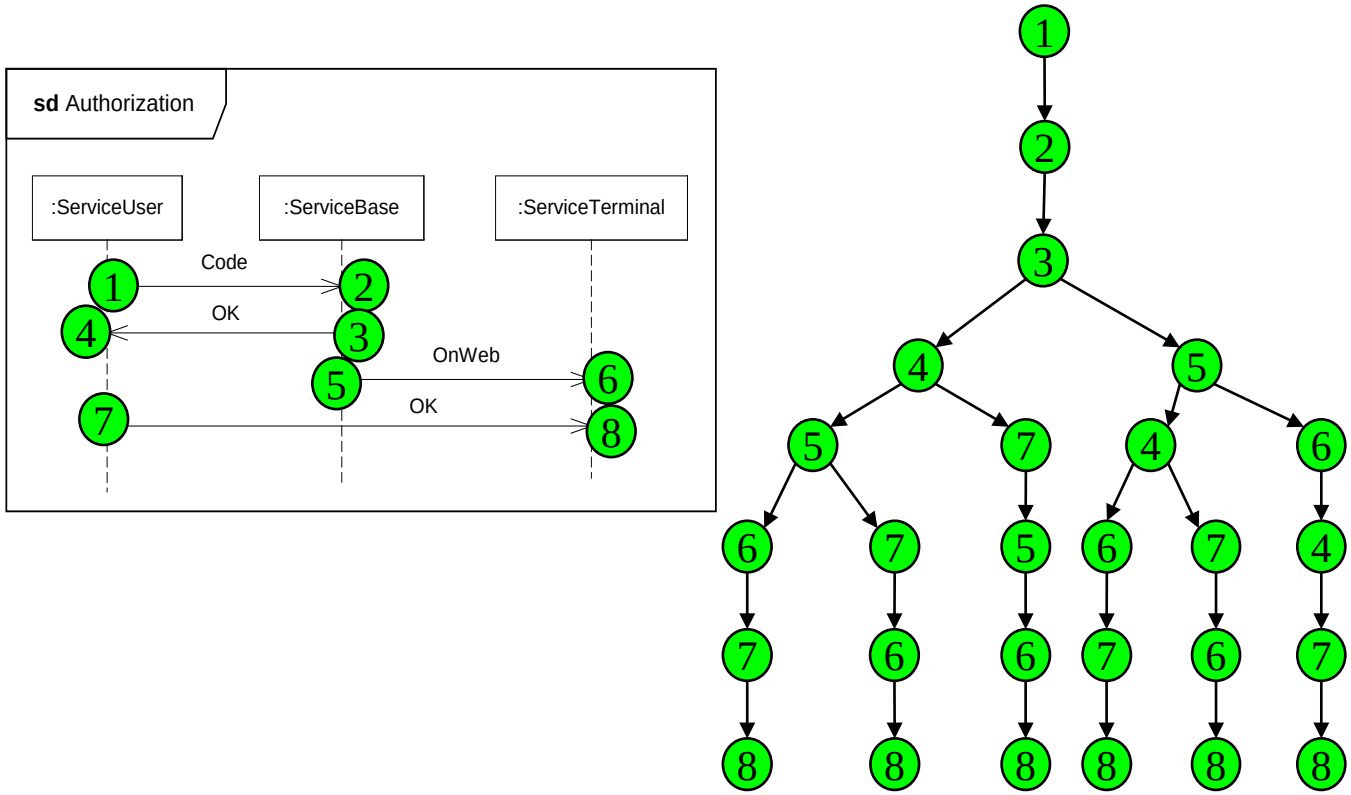


< !Code, ?Code, !OK, ?OK, !OnWeb, ?OnWeb, !OK, ?OK >

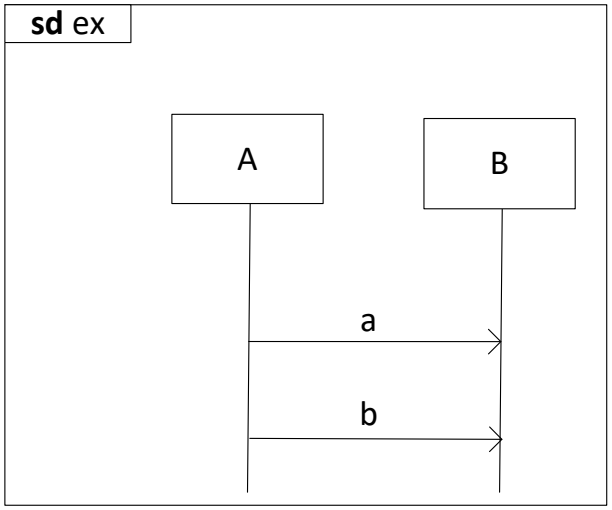
But there is more than one

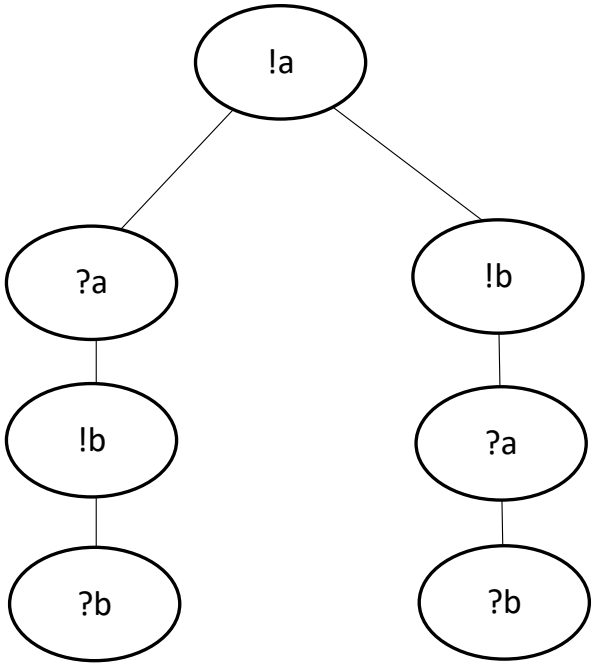


In fact, there are six traces



Another example

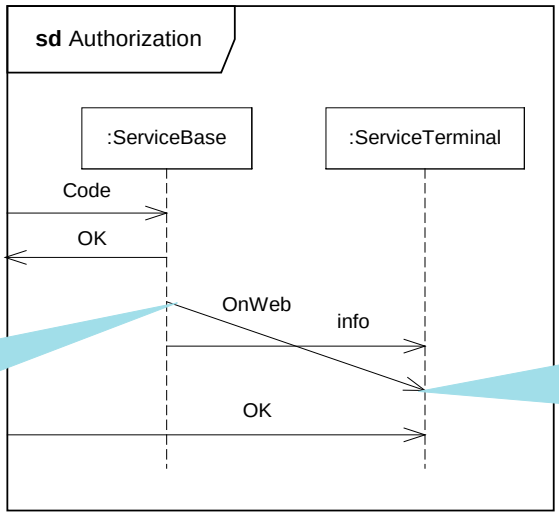




Asynchronous communication

- asynchronous communication = the sender does not wait for the reply to the message sent
- reception is normally interpreted as consumption of the message
- when messages are asynchronous, it is important to be able to describe message overtaking

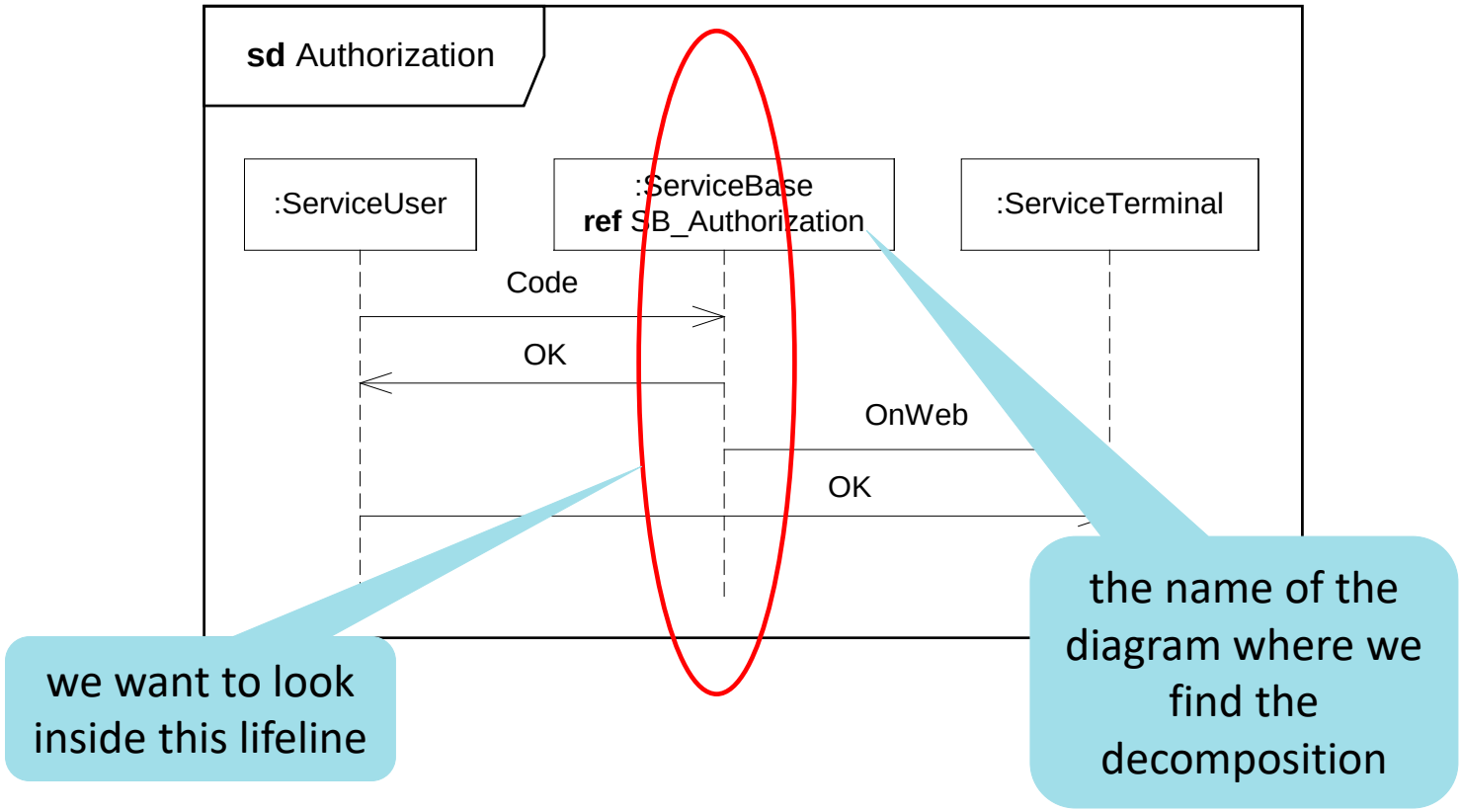
notice message to/from environ-ment



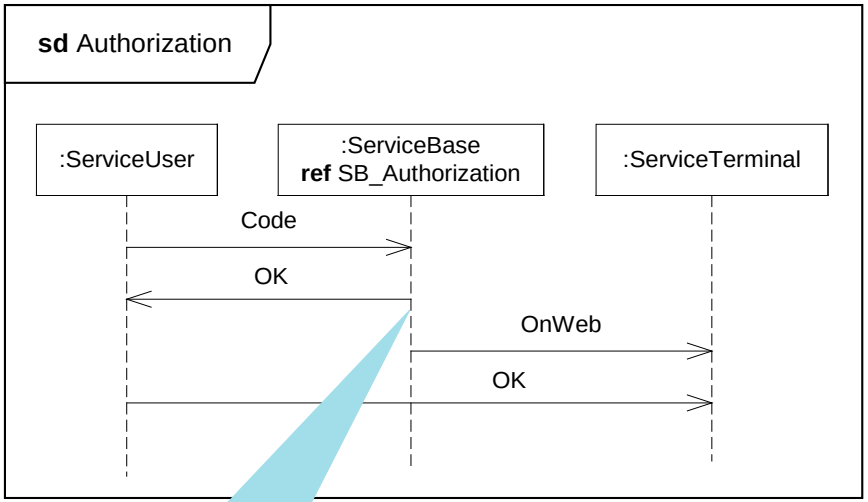
sending OnWeb before sending Info

receiving OnWeb after receiving Info

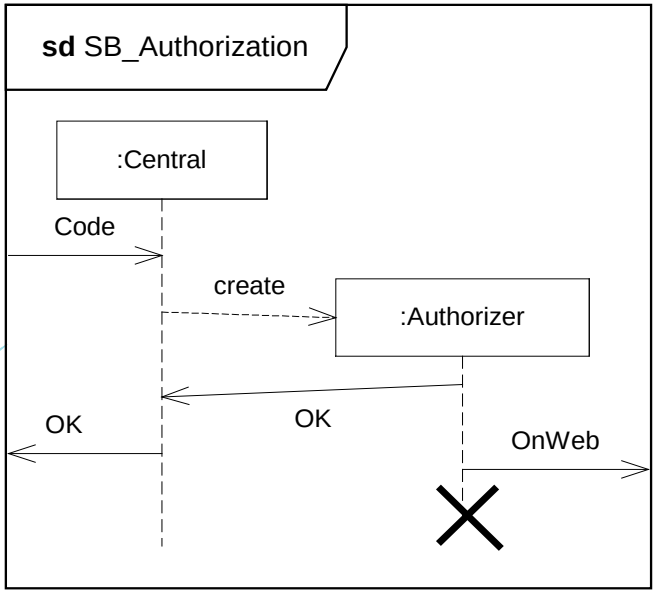
Decomposing a lifeline



The decomposition



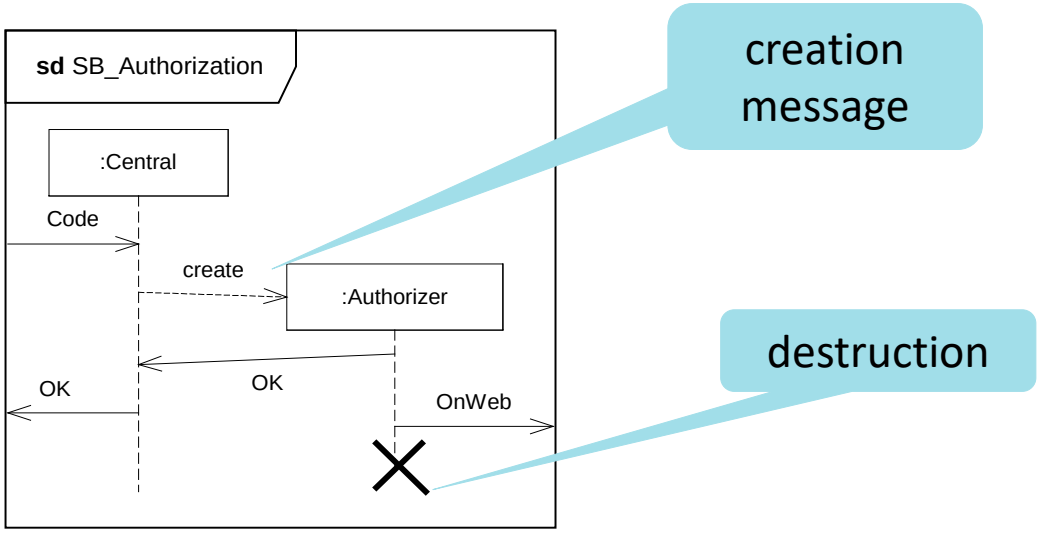
notice the *event* correspondence!



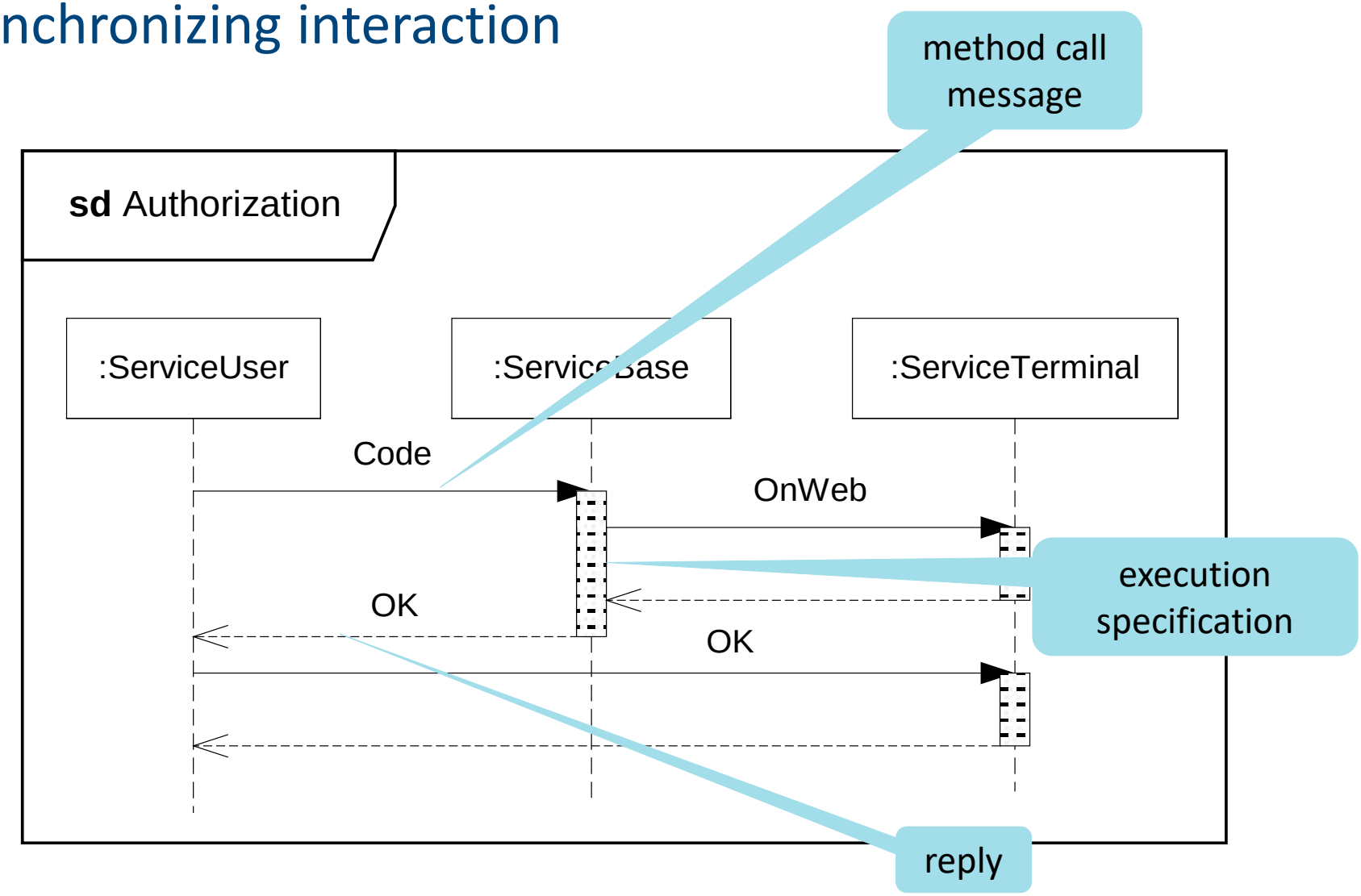
notice the *gate* correspondence!

Lifeline creation and destruction

The idea here (though rather far fetched) is that the ServiceBase needs to create a new process in the big mainframe computer to perform the task of authorizing the received Code



Synchronizing interaction



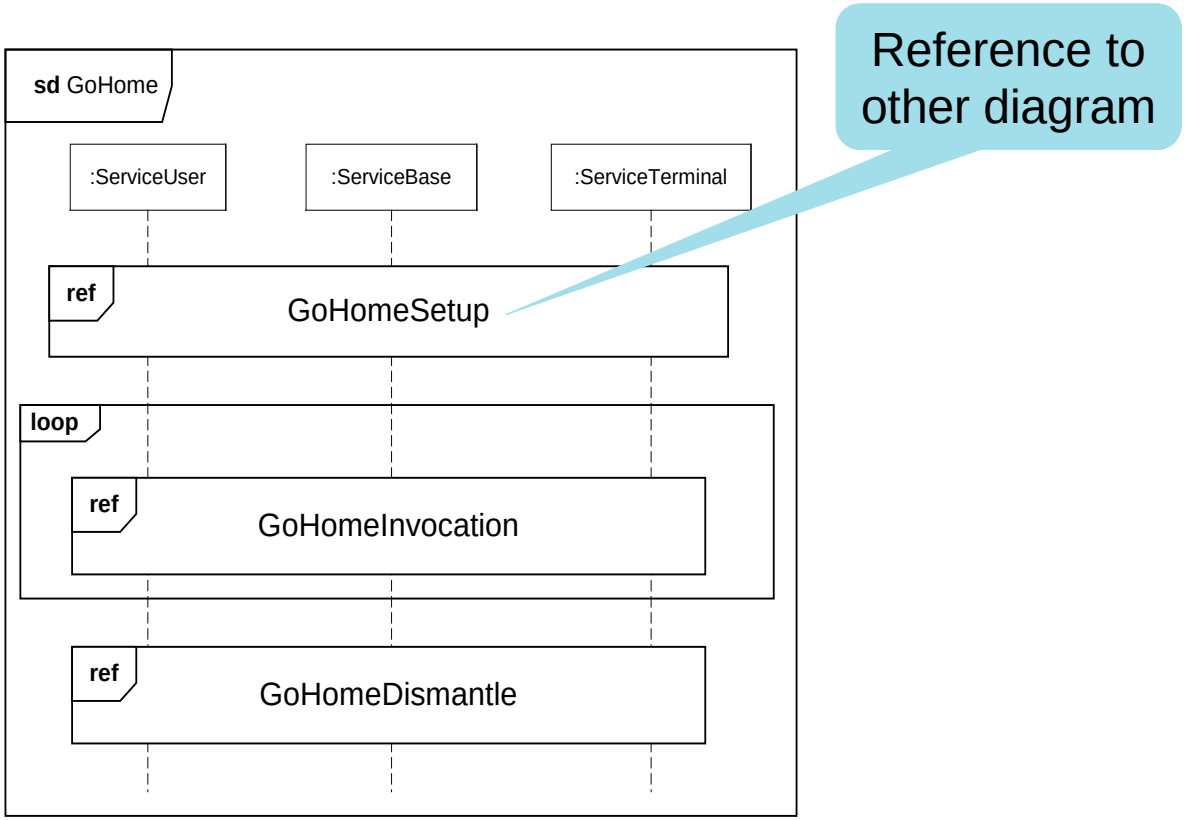
Summary of basic sequence diagrams

- We consider mostly messages that are asynchronous
  - the sending of one message must come before the corresponding reception
- The events on a lifeline are strictly ordered
- The distance between events is not significant.
- A lifeline (within an interaction) may be detailed in a decomposition
- We may dynamically create and destruction of lifelines

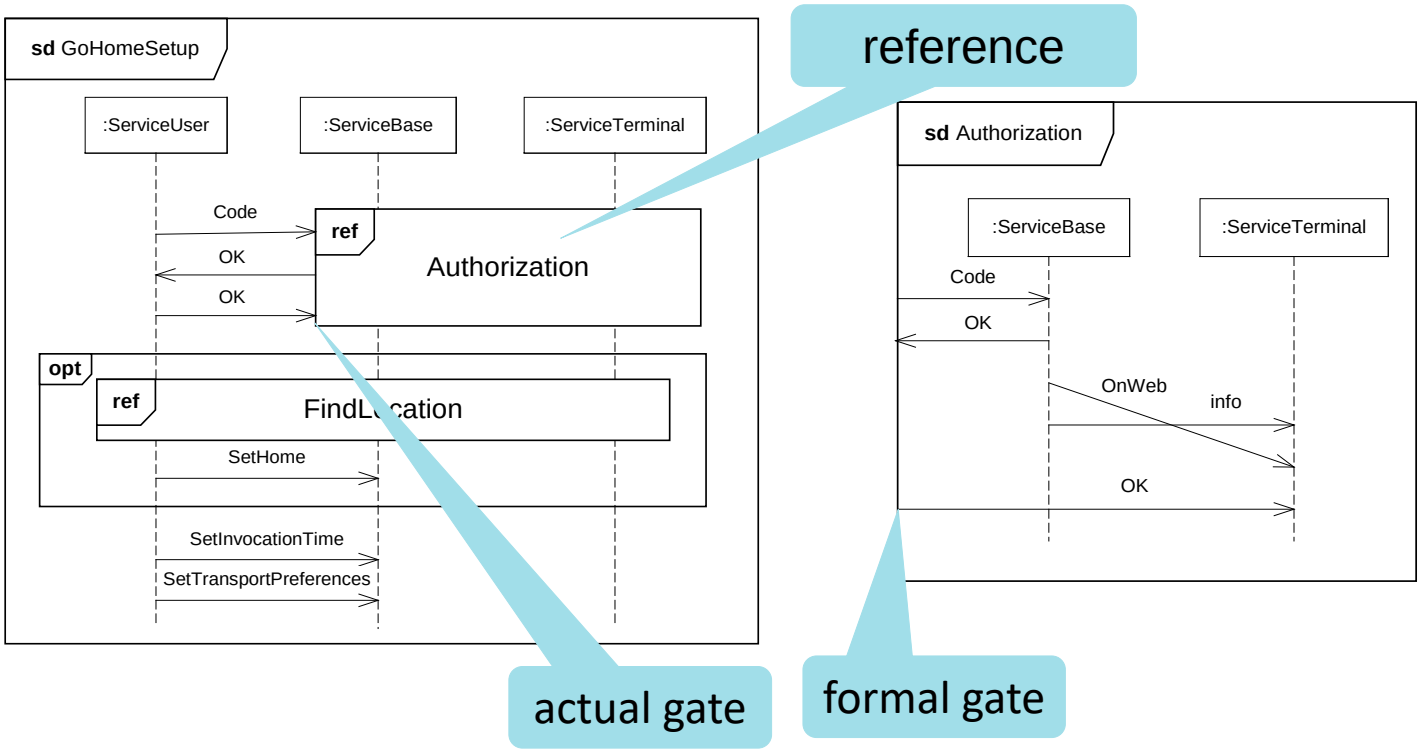
High-level sequence diagrams

- better overview of combinations – hiding lifelines and individual messages
- references – diagrams may be referenced within other diagrams
- gates – connection points between references and their surroundings
- operators – combining fragments to express choice, interleaving and loops
- negative behaviour – describes what is not allowed

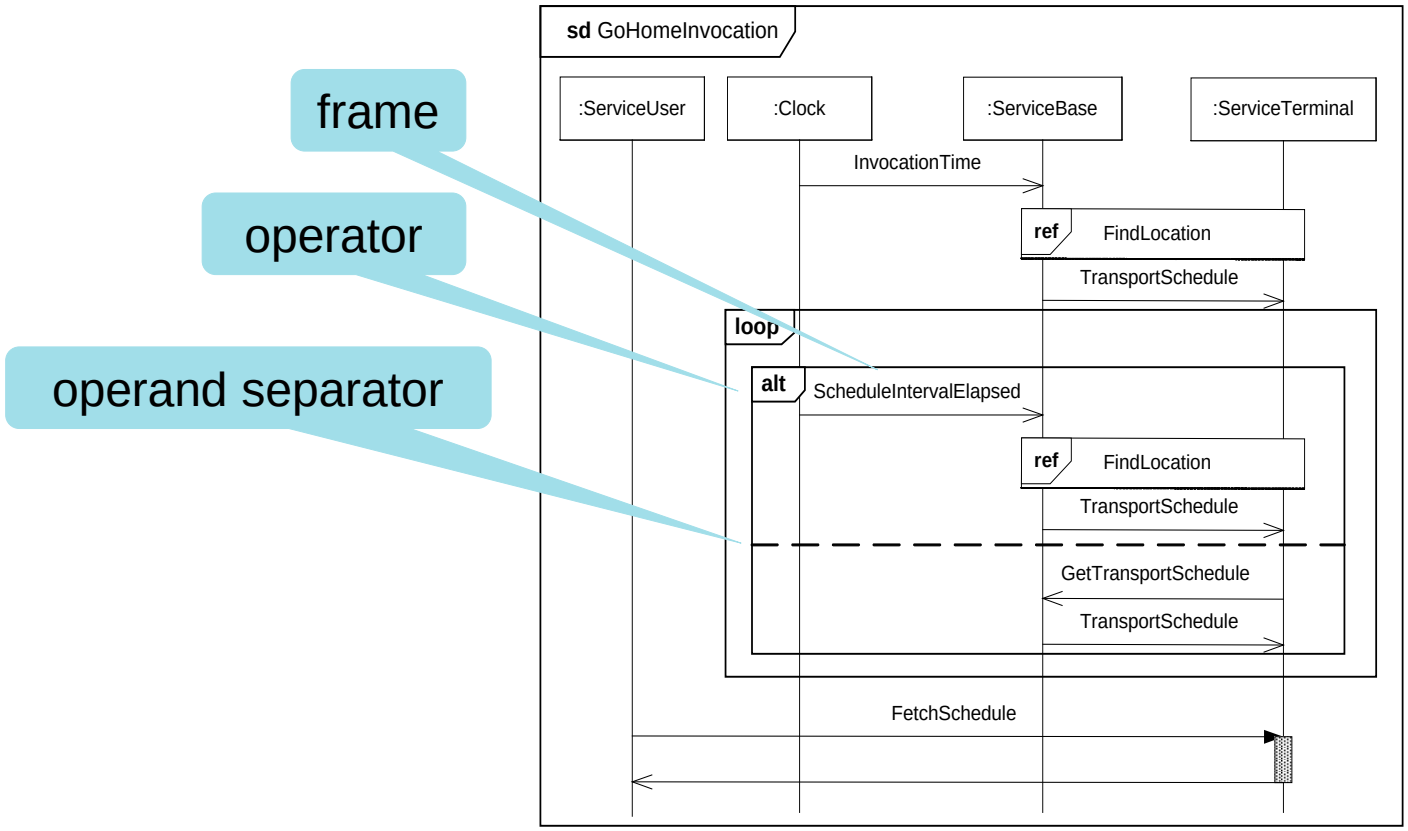
References


sd GoHome
:ServiceUser
:ServiceBase
:ServiceTerminal
ref
GoHomeSetup
loop
ref
GoHomeInvocation
ref
GoHomeDismantle
Reference to other diagram

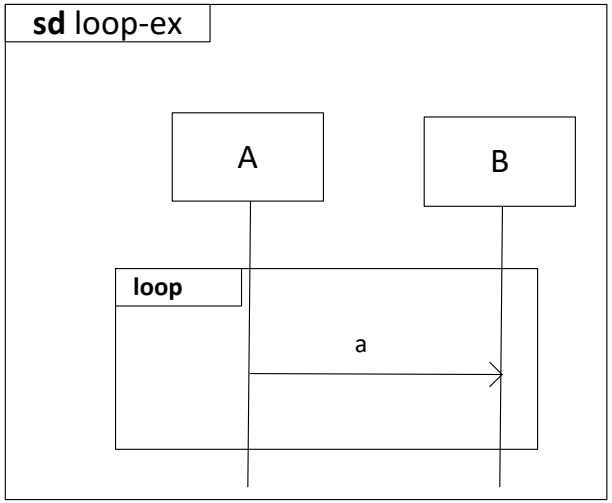
Gates and opt operator



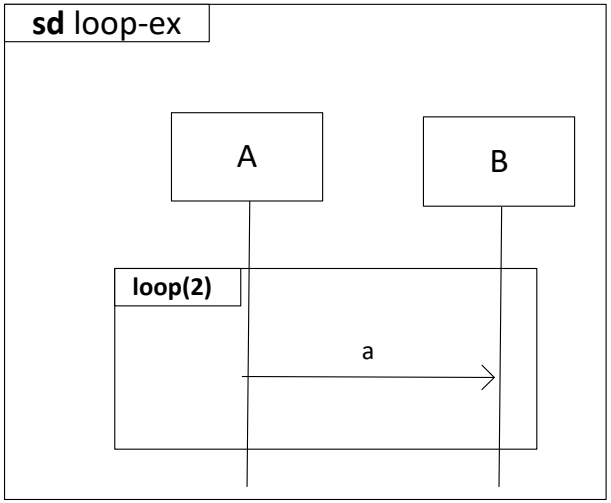
Operators alt and loop



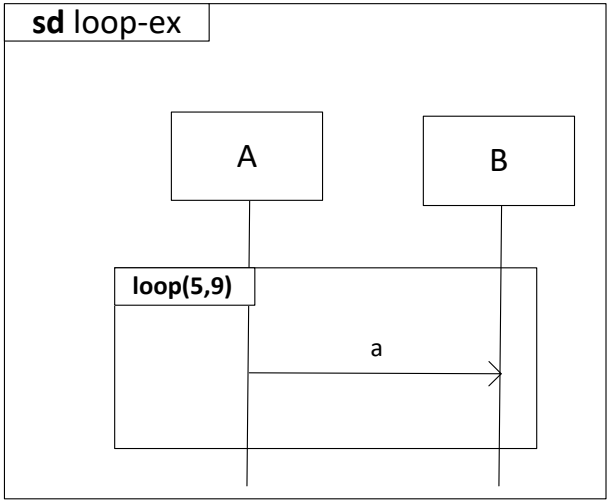
Various example usages of loop operator



Loops an arbitrary number of times between zero and infinite

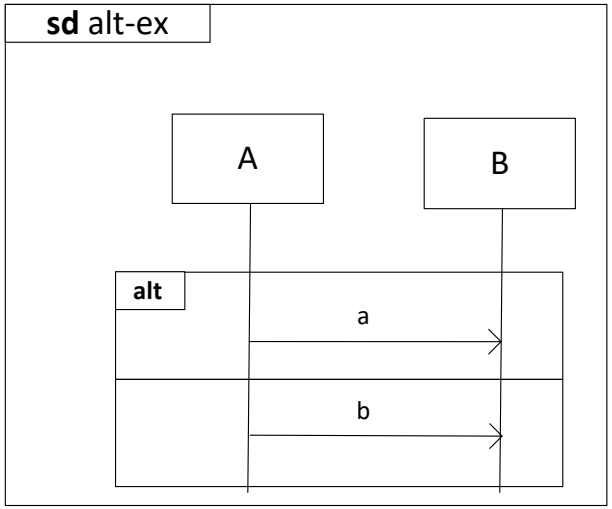


Loops two times

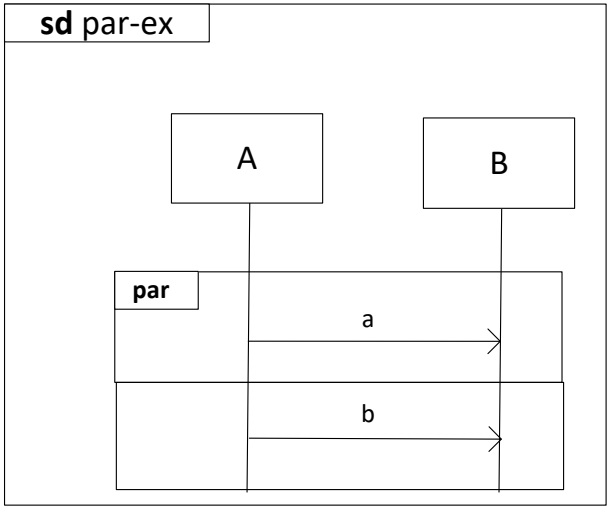


Loops an arbitrary number of times between 5 and 9

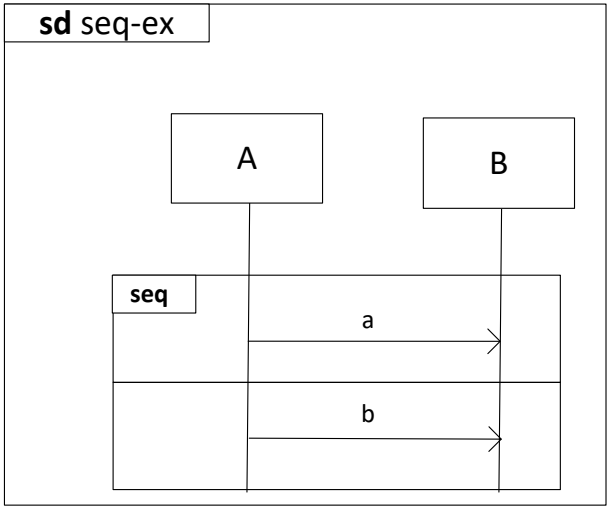
Overview of operators



arbitrary choice between operands



arbitrary interleaving of the traces from the two operands

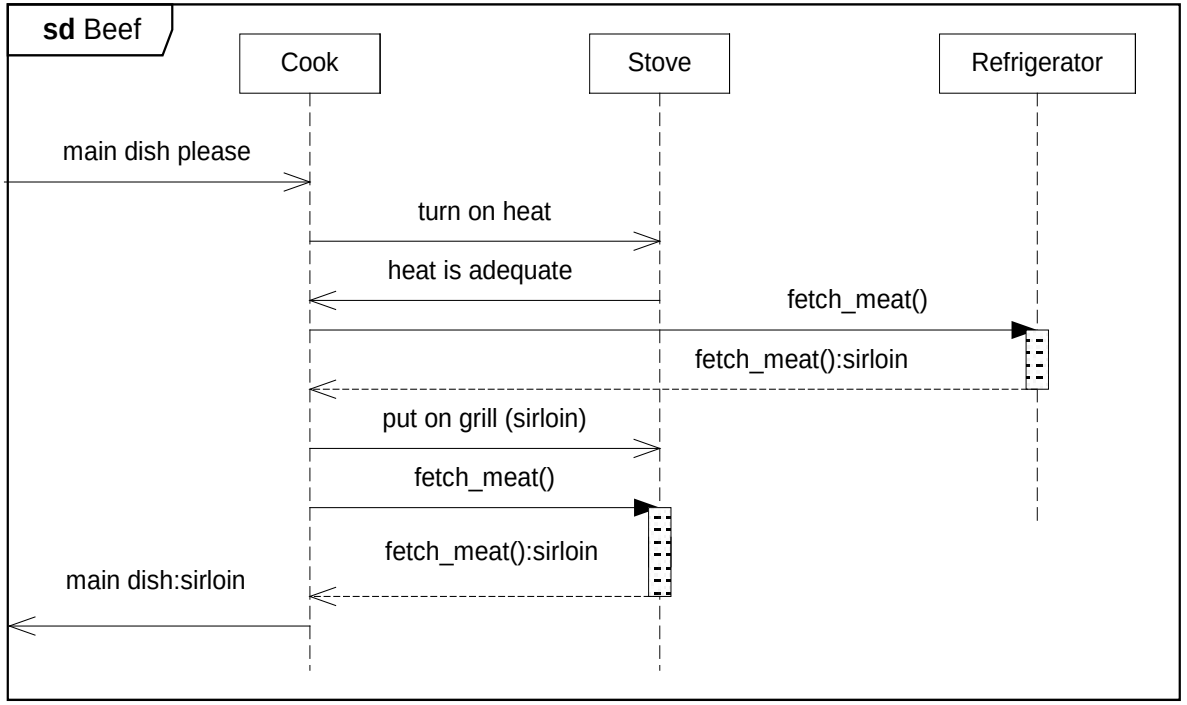


If we remove the frame but keep the messages the semantics is unchanges

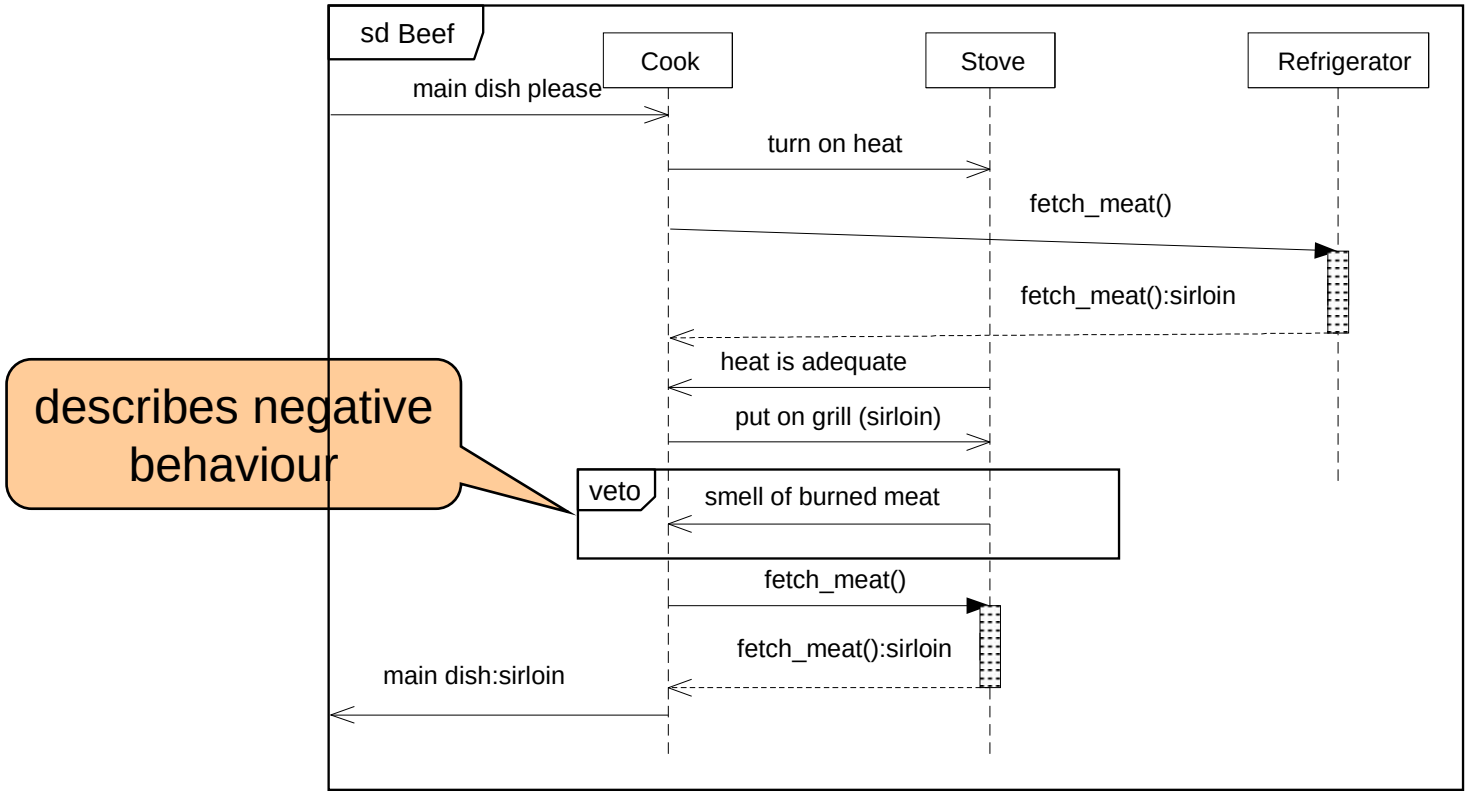
Positive versus negative behaviour

- Positive behaviour
  - the executions that are allowed
  - referred to as the positive traces
- Negative behaviour
  - the executions that are not allowed
  - referred to as the negative traces

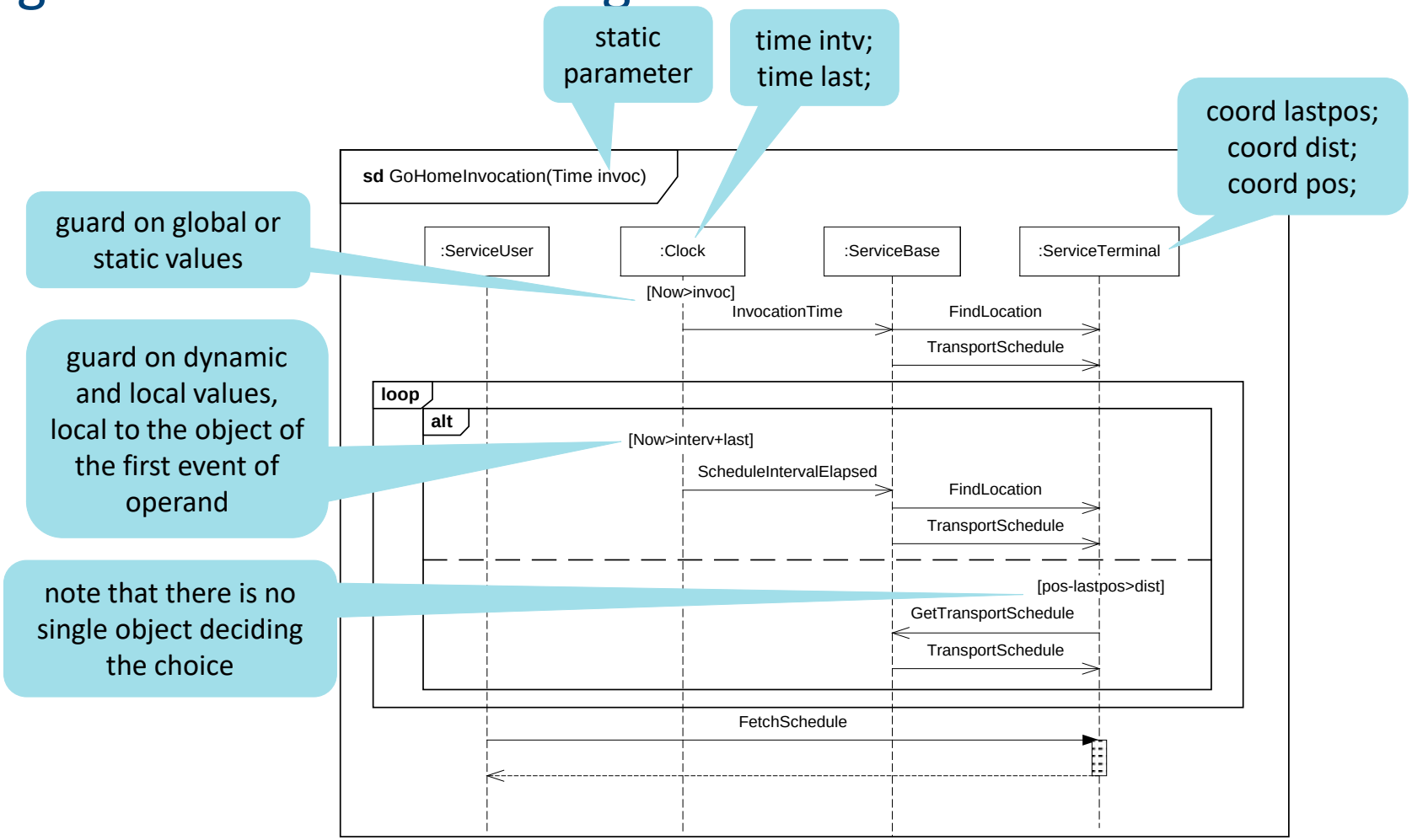
Ordering Beef



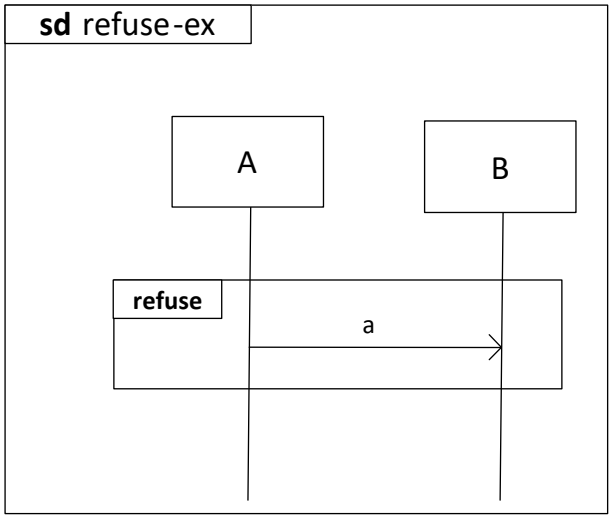
Ordering Beef also including negative behavior

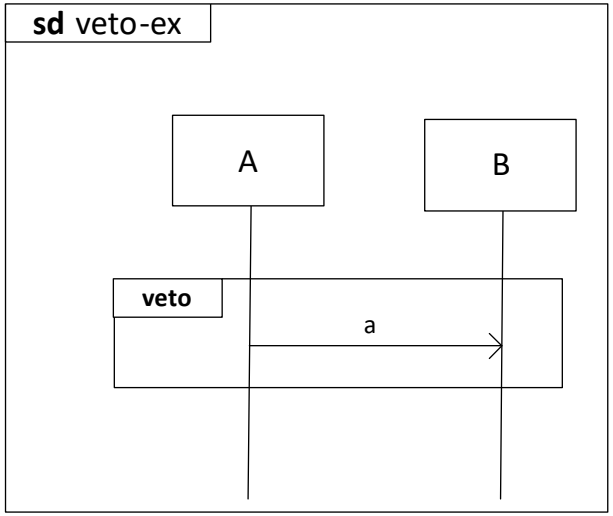


Negative behavior due to guards



veto and **refuse**





Describes one negative trace <!a,?a>, but no positive trace

Describes one negative trace <!a,?a> and one positive trace <> (the empty trace)