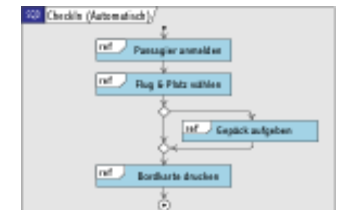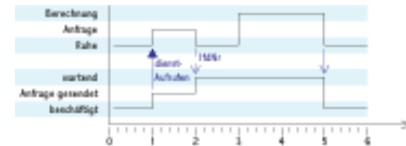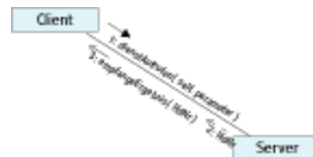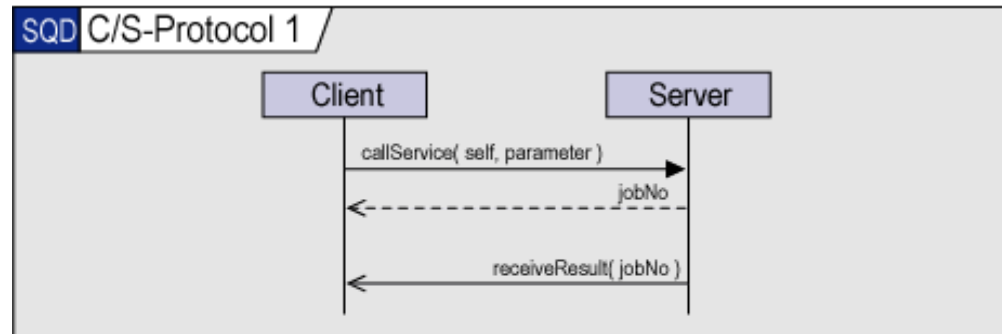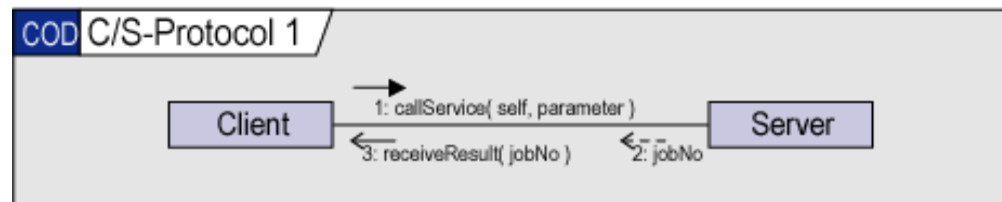# Unified Modeling Language 2
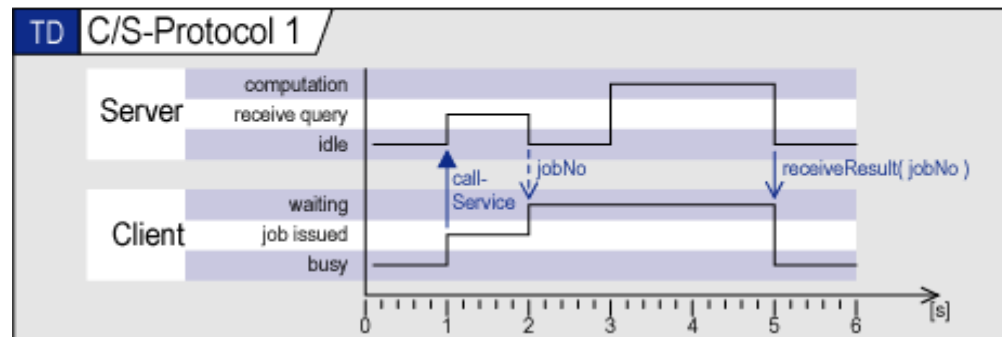
Interactions

A first glimpse

sequence diagram

communication diagram

all three are semantically equivalent

timing diagram

History and predecessors

- Simple sequence diagrams
  - 1990's
    - Message Sequence Charts (MSCs) used in TelCo-industry
    - several OO-methods use sequence diagrams

- Complex sequence diagrams
  - 1996: Complex MSCs introduced in standard MSC96
  - 1999: Life Sequence Charts (LSCs)

- Communication diagrams
  - 1991: used in Booch method
  - 1994: used in Cook/Daniels: Syntropy

- Timing diagrams
  - traditionally used in electrical engineering
  - 1991: used in Booch method
  - 1993: used in early MSCs

- Interaction overview
  - 1996: high-level MSCs (graphs of MSCs as notational alternative)

Usage scenarios

- **Class/object interactions**
  - design or document message exchange between objects
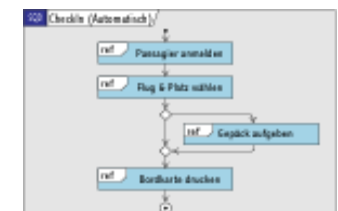 - express synchronous/asynchronous messages, signals and calls, activation, timing constraints

- **Use case scenarios**
  - illustrate a use case by concrete scenario
  - useful in design/documentation of business processes (i.e. analysis phase and reengineering)

- **Test cases**
  - describe test cases on all abstraction levels

- **Timing specification/documentation**

- **Interaction overview**
  - organize a large number of interactions in a more visual style
  - defined as equivalent to using interaction operators
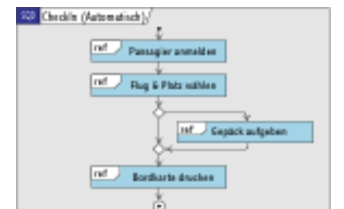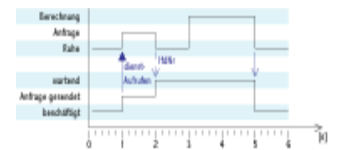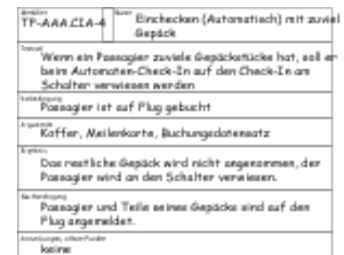
Syntactical variants

- Sequence diagram
  - traditional sequence diagrams + interaction operators
  - focuses on exchanging many messages in complex patterns among few interaction partners
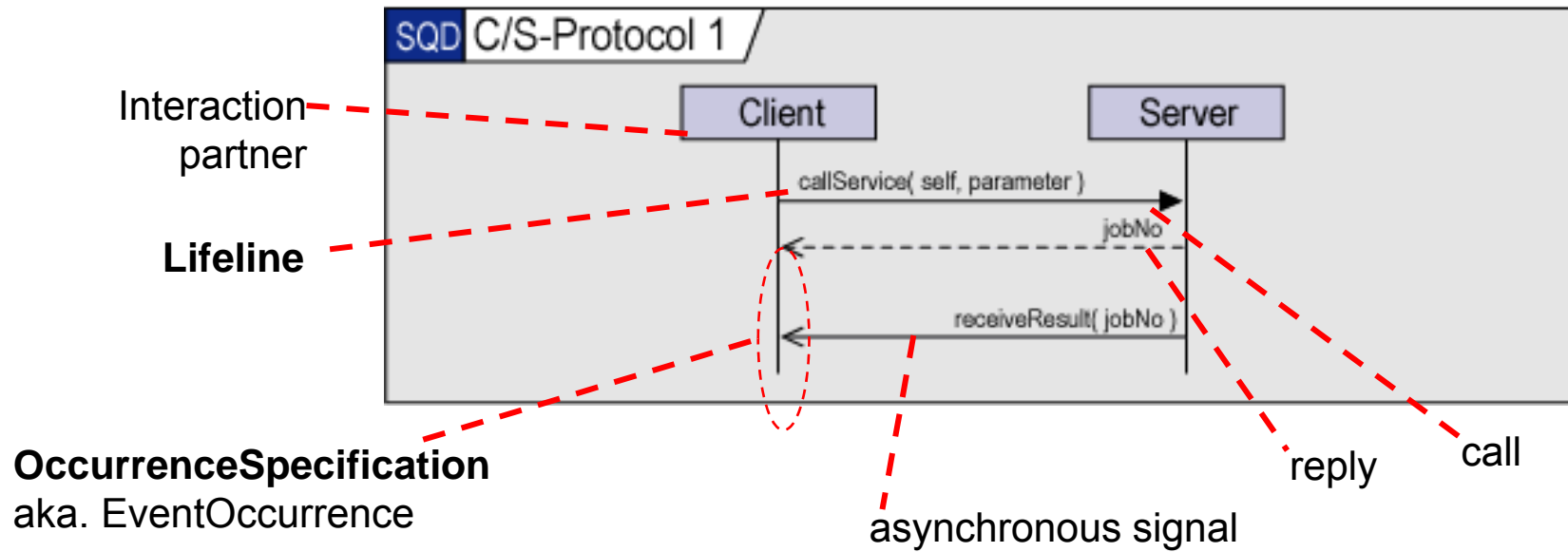

- Communication diagram
  - "collaboration diagram" in UML 1.x
  - focuses on exchanging few messages between (many) interaction partners in complex configuration

- Timing diagram
  - new in UML 2.0, oscilloscope-type representation, not necessarily metric time
  - focuses on (real) time and coordinated state change of interaction partners over time

- Interaction overview diagram
  - looks like restricted activity diagram, but isn't
  - arrange elementary interactions to highlight their interaction
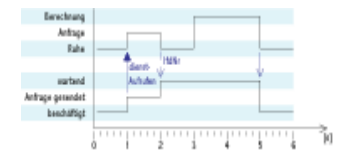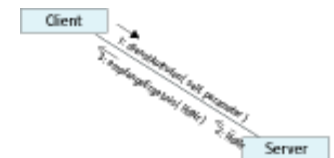
Main concepts

Message types

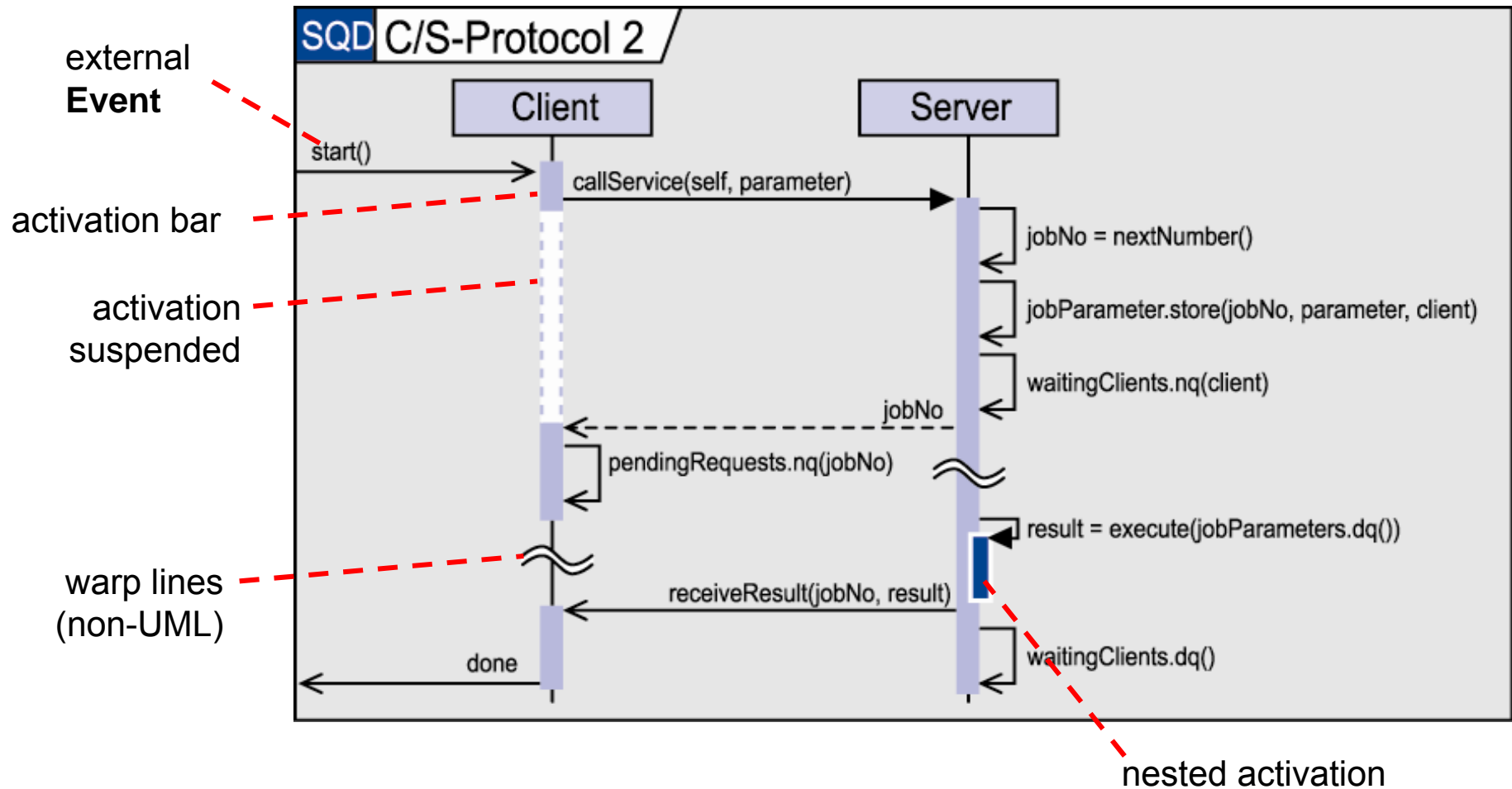instantiation **Event**

lost & found **Message**s
(i.e.: very slow messages)

termination **Event**

non-instantaneous
Message

Activation

external **Event**

activation bar

activation suspended

warp lines (non-UML)

nested activation

SQD C/S-Protocol 2

Client

Server

start()

callService(self, parameter)

jobNo = nextNumber()

jobParameter.store(jobNo, parameter, client)

waitingClients.nq(client)

jobNo

pendingRequests.nq(jobNo)

result = execute(jobParameters.dq())

receiveResult(jobNo, result)

waitingClients.dq()

done

Usage: Use case scenarios

- Interaction **participants** are actors and systems rather than classes and objects.

- May be **refined** successively.

- Useful also for specifying high-level non-functional requirements such as response times.

- All kinds of interaction diagrams may be applied, depending on the circumstances.
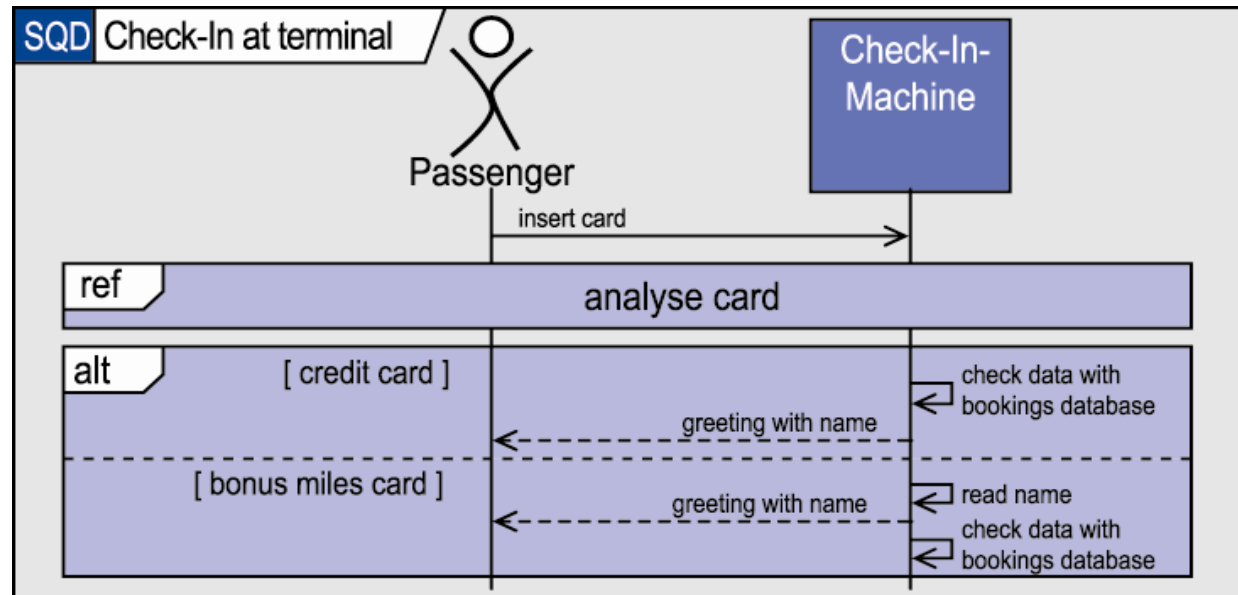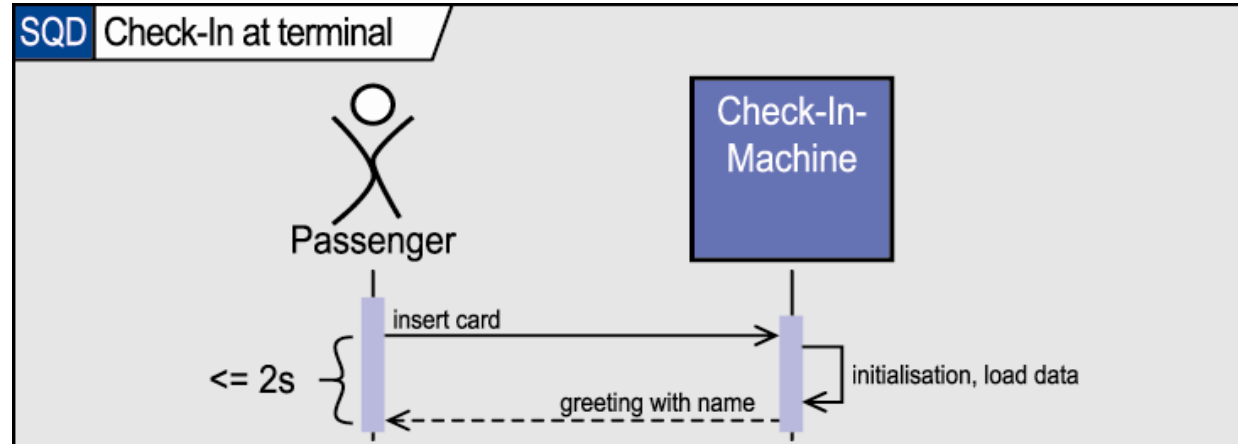
Usage: Class interactions

- Interaction **participants** are classes and objects rather than actors and systems.

- Again, successive **refinement** may be applied in different styles:
  - break down processing of messages
  - break down structure of interaction participants.

- All kinds of interaction diagrams may be applied, depending on the circumstances.