Graphical Systems Modeling with UML / SysML

Sequence diagrams

© Piotr Ciskowski
**Perspectives - views:**

- **architecture**
  - class diagram
  - object diagram
  - composite structure diagram
  - package diagram

- **behavior**
  - sequence diagram
  - activity diagram
  - state machine diagram
  - interaction overview diagram
  - communication diagram
  - timing diagram
  - package diagram

- **system scope & functionality**
  - use case diagram
  - package diagram

- **software**
  - component diagram
  - package diagram

- **hardware**
  - deployment diagram
  - package diagram

- **Logical view**

- **Dynamic View**

- **Use case view**

- **Implementation View**
Sequence diagram

- describes interactions between instances as sequences of messages passed between them

- documents functionality of a use case

- shows:
  - flow of control
  - order and location of performed operations
  - exchange of messages between objects - chronologically
  - logic of methods and services

- close to code

- horizontally - classifiers’ instances - statics
- vertically - time - dynamics
**Sequence diagram**

- **types of diagrams - levels of detail / abstraction:**
  - **conceptual** - general scope
    - evident interactions
  - **implementational** - very precise
    - all types of elements
    - main flow and all alternative flows
    - complete specification
    - for programmers / CASE tools
  - **instance** - implementational for a certain scenario
SEQUENCE DIAGRAM BASIC ELEMENTS
Basic elements:

- objects, actors
  (classifiers' instances)

- lifelines

- messages

- activation
  - focus of control
  - object active / inactive
SEQUENCE DIAGRAM ADVANCED ELEMENTS
Advanced elements

- on detailed implementation diagrams:
  - different types of messages
  - creating and destroying objects
  - conditions
  - messages to self (self messages)
  - duration and time constraints
  - combined fragments
  - interaction uses
  - gates
Advanced elements

MESSAGES
Messages

- synchronous call
- asynchronous call
- asynchronous signal
- reply

- self

- create
- delete

- lost
- found
Advanced elements

OTHER ELEMENTS
Other elements

- conditions
- iterations
- forks
- constraints
  (time & duration)
Advanced elements

**COMBINED FRAGMENTS**
Combined fragments

- **combined fragment** - part of sequence diagram
  enclosed in a frame
  executed under specific circumstances

  - adds procedural logic

- **operator** - how to interpret

- **operand** - what to do
Combined fragments

- `alt`
- alternative
- if-then-else
- switch

Image: www.uml-diagrams.org
Combined fragments

- `opt`
- option
- `alt` with only one operand
Combined fragments

- **break**
- breaking or exceptional scenario
- performed instead of other interactions
Combined fragments

- **loop**

  loop (n)
  loop (min, max)
  loop

- iteration

images: www.uml-diagrams.org
Combined fragments

- `neg`
- invalid series of messages
Combined fragments

- **par**

- parallel, concurrent processing

[Diagram of a UML diagram showing the parallel processing of `search_google()`, `search_bing()`, and `search_ask()`]
Combined fragments

- par

- simplified notation

(image: www.uml-diagrams.org)
Combined fragments

- **critical**
  - critical section
  - highest priority
  - object „reservation“
Combined fragments

- **consider** - **ignore**

```java
consider { <messages> }
ignore { <messages> }
```

---

images: www.uml-diagrams.org
Combined fragments

- **strict**
- strict order (sequencing)

![Diagram showing strict and strict order (sequencing) with search_google(), search_bing(), and search_yahoo() functions]
Combined fragments

- *seq*
- weak sequencing

![UML Diagram](www.uml-diagrams.org)
Combined fragments

- ref

- other interaction use

images: www.uml-diagrams.org
Combined fragments:

- **alt**
- **opt**
- **break**
- **loop**
- **neg**
- **par**
- **critical**
- **consider, ignore**
- **strict, seq**
- **ref**

- alternative
- option
- breaking or exceptional scenario
- iteration
- invalid actions
- parallel processing
- critical section, highest priority
- strict order, weak sequencing
- reference to other interaction
SUMMARY
Sequence diagram

Steps:

1. analyze use case scenario
2. identify classifiers to take part in the interaction
3. create conceptual diagram:
   - classifiers’ instances - horizontally
   - ordered messages - vertically
   - execution boxes
4. create implementation diagram:
   - all other details, advanced elements and full specifications needed
5. create instance diagram