

Octopus/UML Facility Reference

Version 1.3

Domiczi Endre, Farfarakis Rallis, Ziegler Jürgen
Nokia Research Center

| | | |
|------|--|----|
| 1. | FACILITIES AT A GLANCE..... | 1 |
| 2. | DESCRIPTION OF FACILITIES | 2 |
| 2.1 | Event Diagram ^(A6) | 3 |
| 2.2 | Event List | 4 |
| 2.3 | Inter-Process Message List ^(D34) | 5 |
| 2.4 | Shared Object Table | 6 |
| 2.5 | Statechart List..... | 6 |
| 2.6 | Subsystem Inter-Process Communication Diagram ^(D28) | 7 |
| 2.7 | Subsystem Inter-Process Scenario ^(D32) | 8 |
| 2.8 | Subsystem Operation Diagram ^(D35) | 8 |
| 2.9 | Subsystem Scenario ^(A8) | 10 |
| 2.10 | System Scenario ^(D23) | 11 |
| 2.11 | Use Case Diagram | 12 |

1. FACILITIES AT A GLANCE

This chapter lists the facilities^(C26) grouped according to the phases in the system development sequence and defines the notation that each facility makes use of. For each phase a table is presented. The superscripts found in the following tables refer to the IDs defined in the Chapter Release of Octopus/UML. The notation is explained in the Chapter Octopus/UML Notation Summary.

Table 1-1. Facilities and Notation for System Requirements Specification Phase

| Facility | Notation |
|------------------|------------------|
| Use Case Diagram | Class Diagram |
| System Scenario | Sequence Diagram |

Table 1-2. Facilities and Notation for Subsystem Analysis Phase

| Facility | Notation |
|--|------------------|
| Event List | Text Table |
| Event Diagram ^(A6) | Class Diagram |
| Subsystem Operation Diagram ^(D35) | Class Diagram |
| Subsystem Scenario ^(A8) | Sequence Diagram |
| Statechart List | Text Table |

Table 1-3. Facilities and Notation for Subsystem Design Phase

| Facility | Notation |
|---|-----------------------|
| Inter-Process Message List ^(D34) | Text Table |
| Shared Object Table | Text Table |
| Subsystem Inter-Process Scenario ^(D32) | Sequence Diagram |
| Subsystem Inter-Process Communication Diagram | Collaboration Diagram |

2. DESCRIPTION OF FACILITIES

This section gives a summary of each facility. The facilities are arranged in alphabetical order. The phase in which the facility belongs, and the notation used to express the facility is given.

In Octopus/UML, the same notation may be used for various facilities. Each facility can make use of the notation in a way that suites best for its purpose and context. This mapping of notation and facility is stated.

Moreover, a template is given for each facility. The templates of the graphical facilities are generic examples. The templates of the text tables are in their final form and layout, but empty. So, a short explanation is given in each type of field.

2.1 EVENT DIAGRAM^(A6)

The purpose of the Event Diagram is to show the hierarchy of the input events of the Event List. Usually these events can be grouped relatively easily showing that the events of a group are in some way, preferably closely, related to each other. For instance, they may have the same properties or they may be dealt with in a similar way, even if they trigger separate responses.

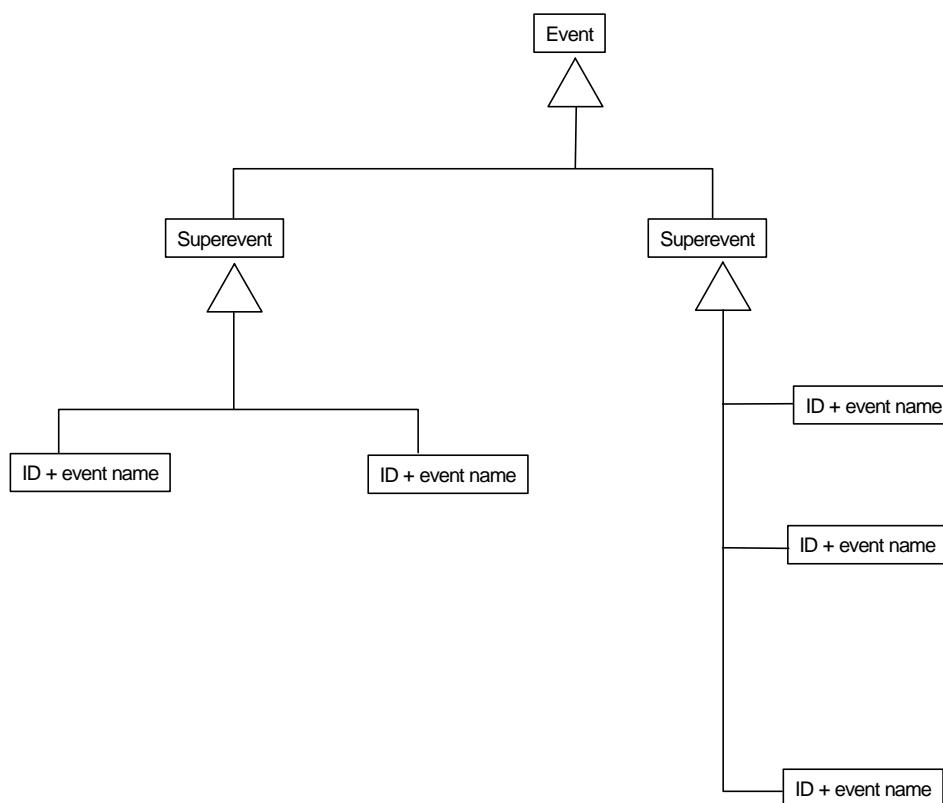
For each group of events a pseudo-superevent is introduced and the events of the group become subevents of this superevent. The superevent can be conveniently used in descriptions or discussions whenever you want to refer to the group as a whole or an arbitrary member. For instance, you may produce an event sheet for the superevent, which covers the common properties of all members. Superevents can also constitute higher order groups.

Phase: subsystem analysis

Notation: class diagram notation.

Mapping: input events as classes; grouping of events as inheritance tree.

Template:



2.2 EVENT LIST

The purpose of the Event List is to show all input and output events^(C21) that the subsystem under consideration receives or sends respectively. These events have been defined and agreed by the subsystem development team.

An input event^(C23) has its origin in the environment of the subsystem and crosses the subsystem boundary from outside to inside. The subsystem class diagram must show an environment class related to the origin of each input event.

Any event entered into the Event List as an input event, must also be considered in its role as an output event^(C23). Thus, the entries in the Event List of the receiving subsystems must be consistent with the corresponding entries in the Event List of the originating subsystems.

An output event has its target in the environment of the subsystem and crosses the subsystem boundary from inside to outside. The subsystem class diagram must show an environment class related to the target of each output event.

Any event entered into the Event List as an output event, must also be considered in its role as an input event. Thus, the entries in the Event List of the originating subsystems must be consistent with corresponding entries in the Event List of the receiving subsystems.

If the subsystem under consideration is an environment wrapper^(C22) subsystem, the Event List includes primitive events^(C25) as input events, and primary events^(C24) as output events. Only an environment wrapper subsystem can originate primary events.

If the subsystem under consideration is an application domain subsystem, all output events of the Event List are secondary events^(C24). The Event List may include primitive events as input events.

Phase: subsystem analysis

Notation: text table notation

Mapping: three columns; floating number of rows; column headers in first row; one row for each event; two parts, one for input, one for output events

Template:

| Event | Description | Originator/Target Subsystem(s) |
|-----------------|---|--------------------------------|
| Input Events | | |
| ID + event name | Short statement describing either what occurrence in the environment is announced by this event | originator subsystem |
| Output Events | | |
| ID + event name | The output produced by the subsystem, i.e. an announcement or a request for service | receiving subsystem(s) |

2.3 INTER-PROCESS MESSAGE LIST^(D34)

The purpose of the Inter-Process Message List is to define the inter-process messages created because of a decision to use an asynchronous mechanism for the implementation of a message flow.

Three types of inter-process messages usually are identified and they should be recorded in the following order: input events, output events and messages between processes of the same subsystem.

Phase: subsystem design

Notation: text table notation

Mapping: three columns; floating number of rows; column headers in top row

Template:

| Inter-process Message | Description | Reception |
|-------------------------------|---|--|
| ID:inter-process message name | Short statement describing the inter-process message; in particular refer to the event if the inter-process message is the implementation of an event; details about the data structure of the inter-process message are given in the inter-process message outline | State either primary wait point or internal wait point |
| ... | ... | |

2.4 SHARED OBJECT TABLE

The purpose of the Shared Object Table is to list all shared objects of the subsystem. If an object is shared by more than two groups, it appears for each pair of groups separately in the table. Each sharing situation must be investigated, case by case, if it is critical or not. A sharing situation found critical must be solved by the synchronisation means and recorded in the Shared Object Table.

Phase: subsystem design

Notation: text table notation

Mapping: three columns; floating number of rows; column headers on top

Template:

| Shared Object | Shared Between Groups | Solution |
|-------------------|-------------------------------|---|
| object identifier | (1) pair of groups (2) ... | State either uncritical or synchronisation mechanism used |
| | | |

2.5 STATECHART LIST

The purpose of the Statechart List is to identify the statecharts of the subsystem. It also assigns IDs to statecharts and states. Each statechart and each elementary state is briefly explained.

Phase: subsystem analysis

Notation: text table notation

Mapping: four columns; floating number of rows; column headers on top

Template:

| Statechart | Elementary State | Description | Related to |
|------------|------------------------|-------------------------|--|
| ID + name | ID + name ID + name | Briefly explain purpose | List classes or subsystem operations that make some use of this statechart |
| ... | | | |

2.6 SUBSYSTEM INTER-PROCESS COMMUNICATION DIAGRAM^(D28)

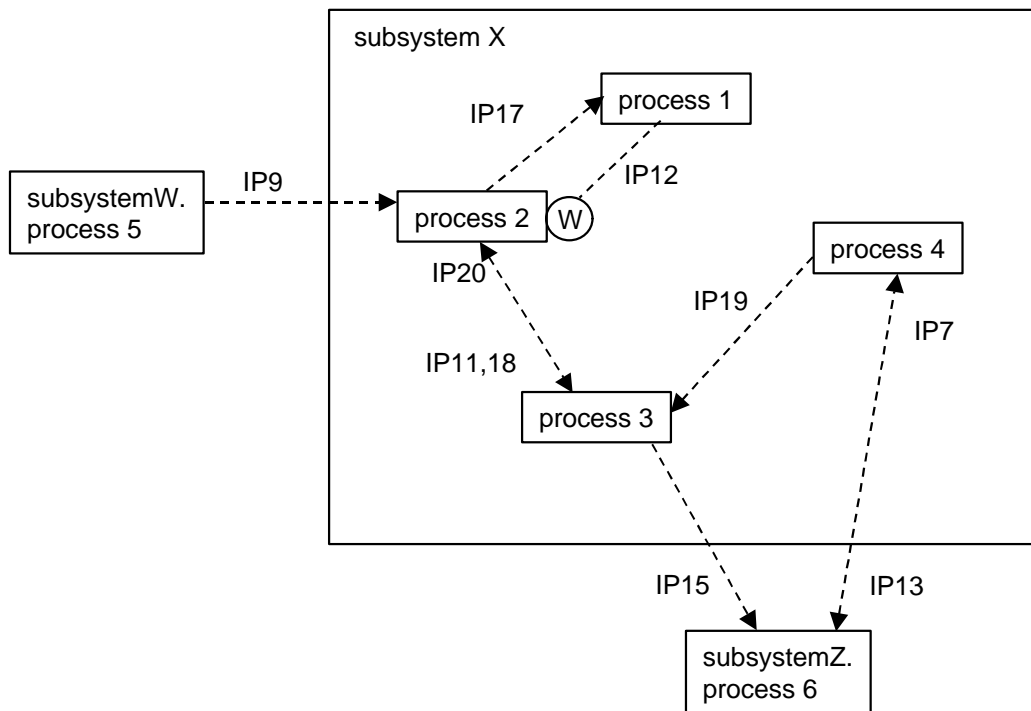
The purpose of the Subsystem Inter-Process Communication Diagram is to show the communication between the processes of the subsystem under consideration. It also shows inter-process messages received from and sent to other subsystems.

Phase: subsystem design

Notation: collaboration diagram notation

Mapping: processes as objects; subsystem as composite object

Template:



2.7 SUBSYSTEM INTER-PROCESS SCENARIO^(D32)

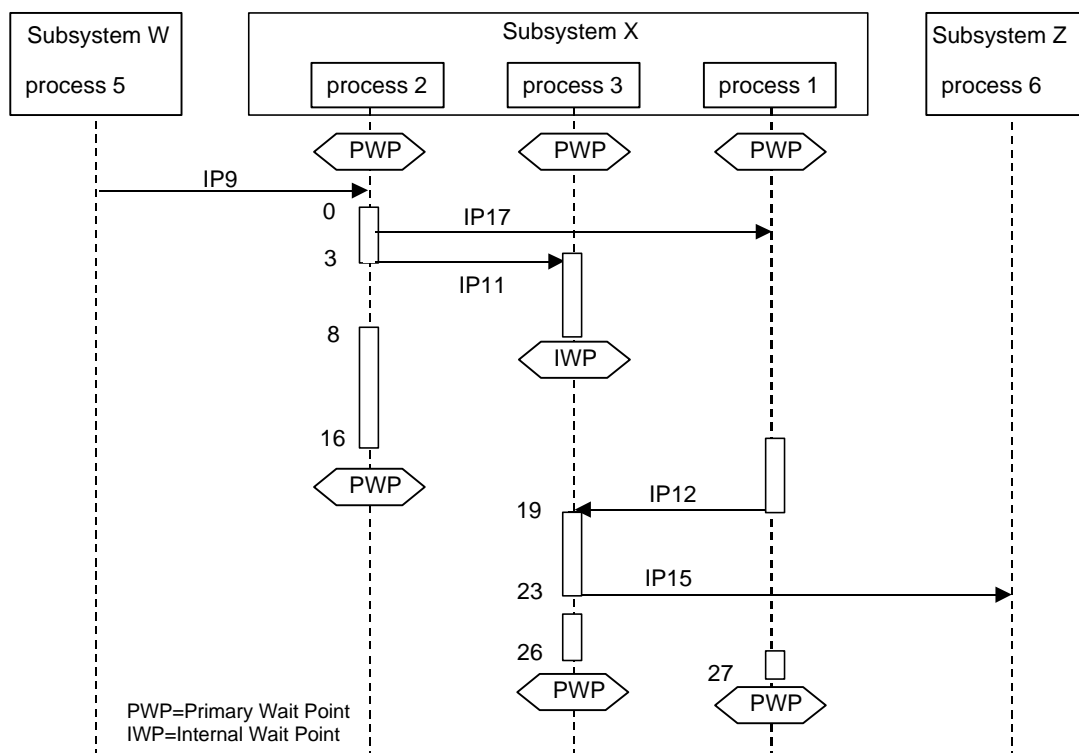
The purpose of the Subsystem Inter-Process Scenario is to show how the processes of a subsystem interleave with each other. This is based on the given process priorities and on the assumption that certain inter-process messages are coming from the outside. Moreover, the execution time estimates can be added, and thus the scenario will allow timeline validation.

Phase: subsystem design

Notation: sequence diagram notation

Mapping: processes as objects; inter-process messages between processes as messages; accumulated execution time as label at the lifeline; currently executing process as activation of object; primary and internal wait point as condition of object.

Template:



2.8 SUBSYSTEM OPERATION DIAGRAM^(D35)

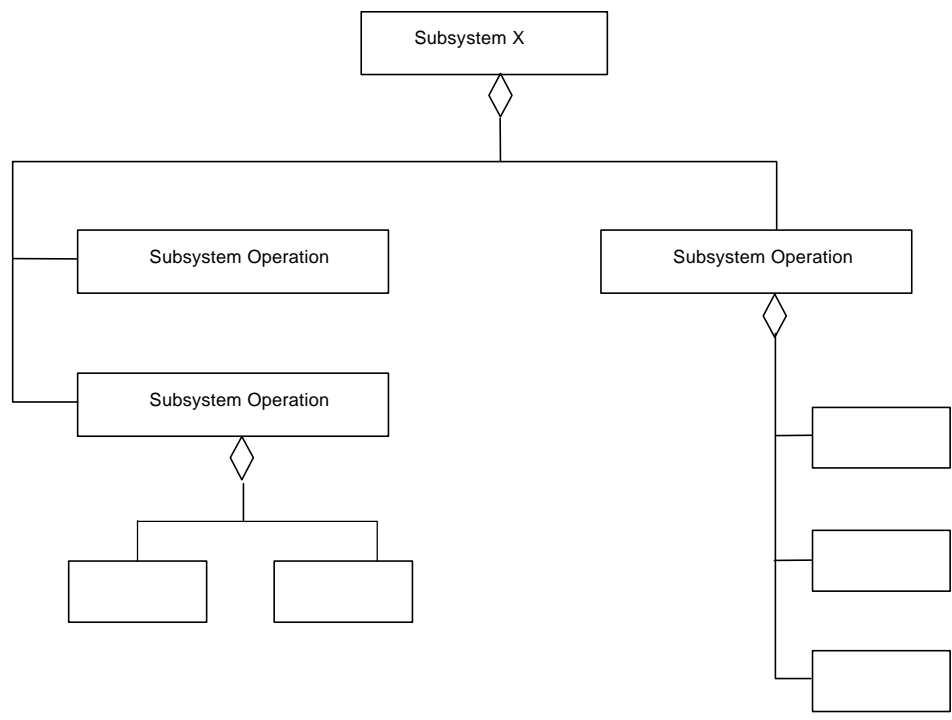
The purpose of Subsystem Operation Diagram is to provide a roadmap through the subsystem operations. The subsystem operations may relate with each other and form a hierarchy. The Subsystem Operation Diagram documents this hierarchy graphically using class diagram notation.

Phase: subsystem analysis

Notation: class diagram notation

Mapping: subsystem as root class of hierarchy; subsystem operation as class aggregated in the subsystem; sub-subsystem operation as class aggregated in the class expressing the super-subsystem operation;

Template:



2.9 SUBSYSTEM SCENARIO^(A8)

The purpose of the Subsystem Scenario is to demonstrate a single, explicit trace of interactions between the subsystem under consideration and the other subsystems and the actors. This trace is one of many ways demonstrating how the subsystem under consideration collaborates with the other subsystems or actors.

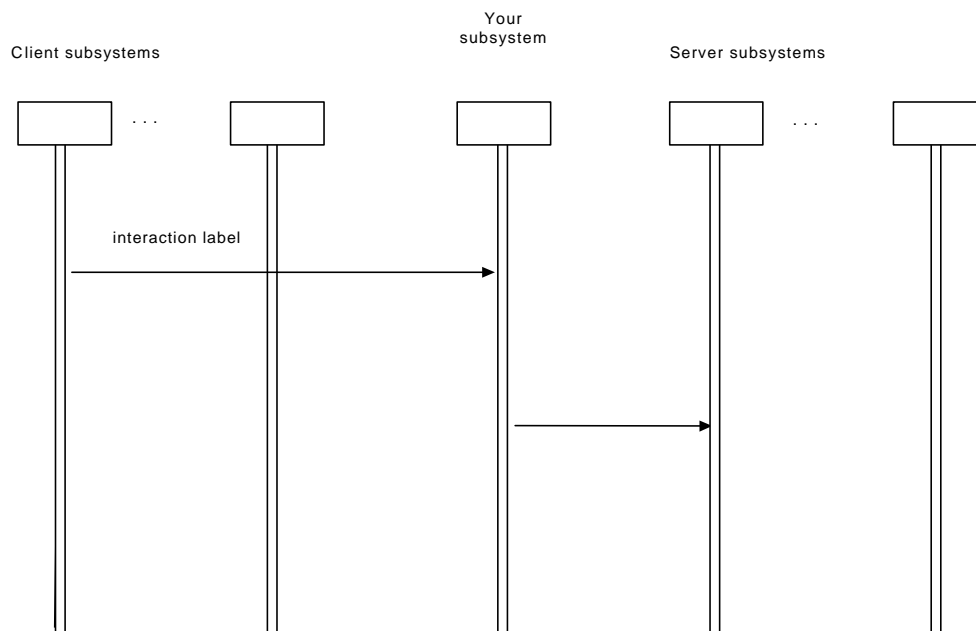
If Inter-Subsystem Scenarios already exist, Subsystem Scenarios may be extracted by placing the target subsystem into the center and removing all the interactions that do not concern the target subsystem.

Phase: subsystem analysis

Notation: sequence diagram notation

Mapping: subsystems as objects; actors as objects; interactions between actors and the subsystem as messages; all objects shown with concurrent activation along the whole lifeline.

Template:



2.10 SYSTEM SCENARIO^(D23)

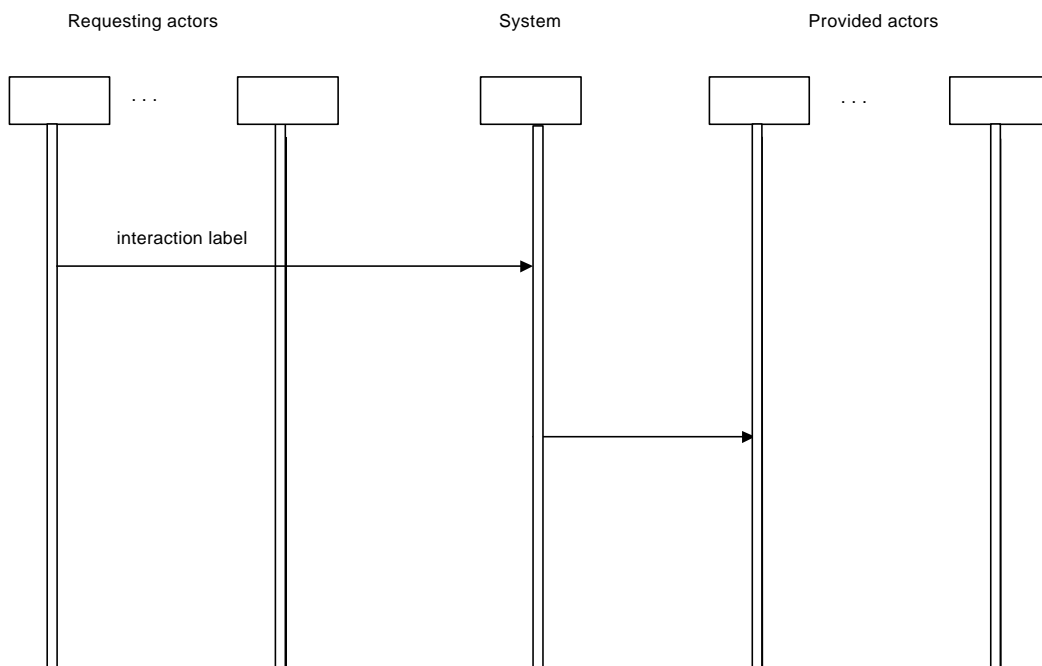
The purpose of the System Scenario is to show the system in action. Each scenario demonstrates a trace of interactions between the actors and the system. This trace is one of many ways which demonstrates how to use the system. It relates to a certain use case or a set of use cases.

Phase: system requirements specification

Notation: sequence diagram notation

Mapping: system as object; actors as objects; interactions between actors and the system as messages; all objects shown with concurrent activation along the whole lifeline

Template:



2.11 USE CASE DIAGRAM

The purpose of the Use Case Diagram is to show the decomposition of the system into branches of related use cases, and to highlight the relationships between the use cases. It serves as a roadmap for the set of Use Case Sheets. Optionally, the actors are added and relationships between the actors and use cases are shown.

In particular, the distinction between the use cases requested by an actor and the autonomously provided ones are shown as the two branches which the system consists of. If an autonomously provided use case has the lifetime of the system, a start relationship between the system and this use case is shown. Otherwise, explicit initiate/terminate relations are shown.

Phase: system requirements specification

Notation: class diagram notation

Mapping: system as root class of hierarchy; use case as class aggregated in the system; sub use case as class aggregated in the class expressing the super use case; actor as class not part of the system aggregation; relationship between use cases and/or actors as association between classes;

Template:

