**Collaboration Diagram (COD)**

**Purpose**

A collaboration diagram shows the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

**Compared with a sequence diagram**

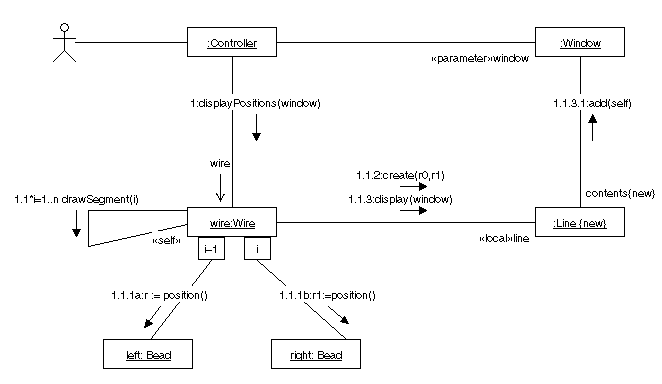
A sequence diagram shows the objects and messages involved in an interaction. It shows the timing of the messages, but not the relationships among the objects.

**As a decomposition diagram**

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram; it can be the decomposition of a use case, use case diagram, or part of a use case diagram.

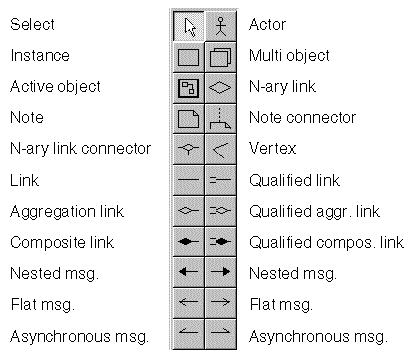
**Example COD**

A sample collaboration diagram.



**Control panel**

The control panel of the collaboration diagram.



**Actors, Instances, and Objects**

**Actor**

An actor in a collaboration diagram represents the person, software, hardware, or other agent external to the system that is interacting with the system.

http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/images/dgmsuml70.gif

**Label**

name

**Instance**

An instance in a collaboration diagram represents an instantiation of a class in a class diagram or a use case in a use case diagram

http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/images/dgmsuml64.gif

**Label**

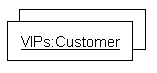
[instance-name][:instance-type]

* instance-name is the name of the instance
* instance-type is a class or a use case

**Note:** While both elements of the label are optional, you must provide one of them to avoid check errors.

**Multi object**

A multi object represents a set of instances at the many end of an association. If necessary, you can use the instance symbol to show one instance of the set and a composite link symbol to show that the instance is part of the multi object.



**Label**

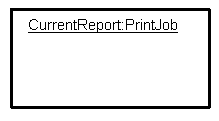
[name][:type]

* name is the name of the multi object
* type is a class or a use case

**Note:** While both elements of the label are optional, you must provide one of them to avoid check errors.

**Active object**

An active object is an instance that owns a thread of control and can initiate control activity. For example, processes and tasks are active objects. Active objects can contain other symbols and links among them.



**Label**

[name][:type]

* name is the name of the active object
* type is a class or a use case

**Note:** While both elements of the label are optional, you must provide one of them to avoid check errors.

**Links**

**Links in collaboration and class diagrams**

The links (associations, aggregations, and so on) in a collaboration diagram show the paths available for messages; they do not provide detailed information about the links. For example, the links in a collaboration diagram do not show the multiplicity of the links. The class diagram defines the links, providing detailed information about them.

**Descriptions of the links**

For more information about each type of link provided in the collaboration diagram, see the class diagram description.

|  |  |
| --- | --- |
| **Link Type** | **Description** |
| Aggregation link | [Aggregation](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "78309) |
| Link | [Association](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "1045101) |
| Composition link | [Composition](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "1079388) |
| N-ary Link | [N-ary associations](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "78238) |
| Qualified link | [Qualified association](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "1066435) |
| Qualified aggregation link | [Qualified aggregation](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "78344) |
| Qualified composition link | [Qualified composition](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml2.html" \l "1079664) |

|  |
| --- |
|  |

**Labels for links**

Each link in a collaboration diagram has two role labels and a name label:

«Link Start Stereotype»role name

«Link End Stereotype»role name

The role names in a collaboration diagram should be the same as the role names in a class diagram. Use the stereotype property of the role names to specify the link start and link end stereotypes.

**Messages**

**Messages**

A message flow carries a message from one object to another along any type of connector (link, aggregation link, and so on). Each message flow in a collaboration diagram is characterized by direction and type.

**Creating and moving messages**

You must create a connector before you can create a message. You can place message symbols on or near a connector, moving them if necessary to prevent them from overlapping the connector. When you move a connector, its message symbols move with it.

**Message direction**

There are two message directions:

* Backward Messages
* Forward Messages

The direction in which the link was drawn determines the direction of the message.

**Message type**

You can create these types of messages:

* [Nested message](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml3.html" \l "1055623)
* [Flat message](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml3.html#1040681)
* [Asynchronous message](http://ics.upjs.sk/%7Enovotnyr/home/skola/softverove_inzinierstvo/uml%20%28telelogic%29/dgmsuml3.html#1154039)

**Message syntax**

The default syntax for a message name is:

<predecessor> <guard-condition> <sequence-expression>

<return-value> [ = | := ] <message-name> (<argument-list>)

**<predecessor> <guard-condition>**

A comma-separated list of sequence numbers followed by a slash. The clause is omitted when the list is empty.

**<sequence-expression>**

A dot-separated list of numbers or names, optionally followed by a condition or an iteration (both enclosed in square brackets, the iteration preceded by an asterisk) and separated from the rest of the label by a colon.

**<return-value> [ = | := ]**

A data element with scope file. If the message does not return a value, the return value and the assignment operator are omitted.

**<message-name>**

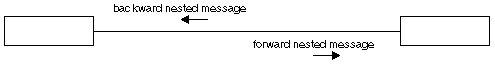
A process element with scope system.

**<argument-list>**

A comma-separated list of data elements with scope file. This list is enclosed in brackets and can be empty.

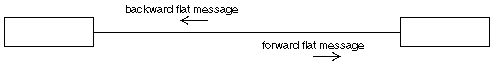
**Nested message**

The nested message represents a procedure call or other nested flow of control. The nested sequence is completed before the outer level sequence resumes. The nested message symbol is represented by a filled solid arrowhead.



**Flat message**

The flat message shows the progression to the next step in a sequence. The flat message symbol is represented by a stick arrowhead.



**Asynchronous message**

The asynchronous message symbol shows an asynchronous message between two objects. The asynchronous message symbol is represented by a half stick arrowhead.

