Activity Diagrams

Activity diagram is UML **behavior diagram** which shows **flow of control** or **object flow** with emphasis on the sequence and conditions of the flow. The actions coordinated by activity models can be initiated because other actions finish executing, because objects and data become available, or because some events external to the flow occur.

The following nodes and edges are typically drawn on **UML activity diagrams**: **activity**, **partition**, **action**, **object**, **control**, **activity edge**.

You can find some **activity diagram examples** here:

- **Online Shopping**
- Business Flow - **Process Order**
- Business Flow - **Document Management Process**
- Software Design - **Resolve Issue**
- **Sentinel HASP SL - Manual Activation of Trial Product**
- **Single Sign-On for Google Apps**

Activity

Activity is a parameterized **behavior** represented as coordinated flow of **actions**.

The **flow of execution** is modeled as activity nodes connected by activity edges. A node can be the execution of a subordinate behavior, such as an arithmetic computation, a call to an operation, or manipulation of object contents. Activity nodes also include flow of control constructs, such as synchronization, decision, and concurrency control. Activities may form invocation hierarchies invoking other activities, ultimately resolving to individual actions. In an object-oriented model, activities are usually invoked indirectly as methods bound to operations that are directly invoked.

Activity contains **activity nodes** which could be:

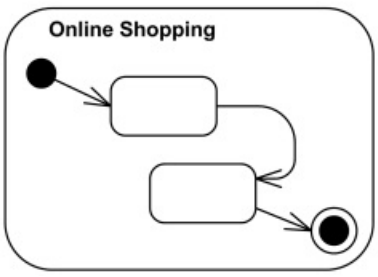
- **action**
- **object**
- **control**

Activities may contain **actions** of various kinds:

- Occurrences of primitive functions, such as arithmetic functions.
- Invocations of behavior, such as activities.
- Communication actions, such as sending of signals.
- Manipulations of objects, such as reading or writing attributes or associations.

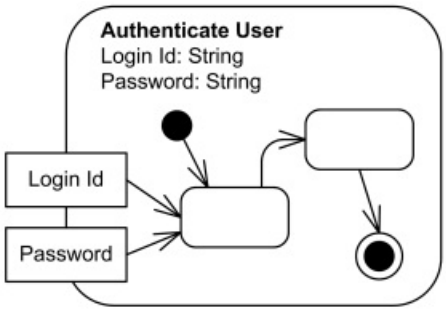
There are actions that invoke activities - either directly using **call behavior action** or indirectly with **call operation action**.

Activity could be rendered as round-cornered rectangle with activity name in the upper left corner and nodes and edges of the activity inside the border. UML 2.4 specification examples show activity name in bold.



Online Shopping activity.

Activity parameters are displayed on the border and listed below the activity name as:
parameter-name: ***parameter-type***.

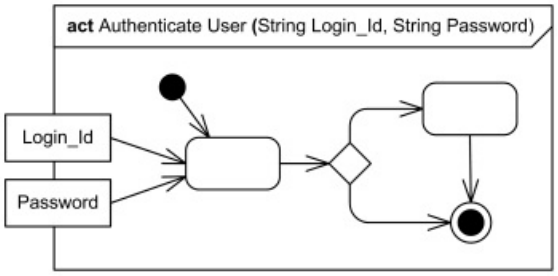


Authenticate User activity with two parameters - Login Id and Password.

As a **behavior** activity could have pre- and post-condition constraints. If present, these are shown with the keywords «precondition» and «postcondition», respectively.

The keyword «singleExecution» is used for activities that execute as a single shared execution (singleton), otherwise, each invocation executes in its own space.

The round-cornered activity border may be replaced with the frame notation for diagrams. The kind of the frame in this case is **activity** or **act** in short form. Activity parameters if any are displayed on the frame.



Authenticate User activity frame with two parameters - Login Id and Password.

The notation for classes with the keyword «activity» can be used to show the features of a reflective activity, to indicate it is an activity class. Association and state machine notation can also be used as necessary.

UML allows behaviors to produce tokens that are **activities** and which can in turn be executed at the runtime.

Activity Partition

An activity **partition** is **activity group** for actions that have some common characteristic.

Partitions often correspond to **organizational units** or **business actors** in a **business model**.

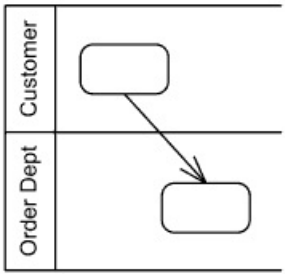
Partitions provide a constrained view on the behaviors invoked in activities. Constraints could be selected according to the type of the element that the partition represents. The following constraints are normative (standard) in UML 2.4:

- **classifier**
- **instance**
- **part**
- **attribute** and value

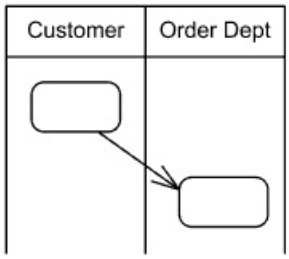
For example, partitions could represent specific **classifiers**. In this case actions in each partition should be operations or signals targeting objects that are instances of the corresponding classifier.

A partition may represent some **attribute** and its subpartitions - specific values of that attribute. For example, a partition may represent the location at which a behavior is carried out, and the subpartitions would represent specific values for that attribute, such as New York.

Activity **partition** may be shown using a **swimlane** notation - with two, usually parallel lines, either horizontal or vertical, and a name labeling the partition in a box at one end. Any activity nodes, e.g. actions and edges placed between these lines are considered to be contained within the partition.

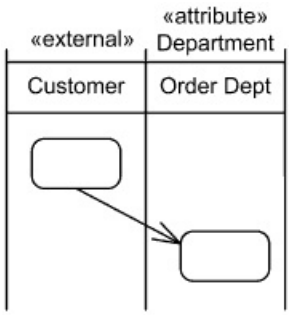


Activity partitions Customer and Order Dept as horizontal swimlanes



Activity partitions Customer and Order Dept as vertical swimlanes

Hierarchical partitioning is represented using swimlanes for subpartitions as illustrated below.



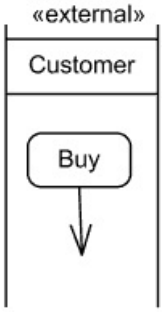
Hierarchical partitioning with subpartitions

A partition may be marked as a **dimension** for its subpartitions to contain (group) those subpartitions along dimension. For example, an activity may have one dimension of partitions for location at which the contained behaviors are carried out, and another for the cost of performing them. **Dimension partitions** cannot be contained by any other partition.

Diagrams can also be partitioned multidimensionally, where each swim cell is an intersection of multiple partitions. The partitions within each dimension may be grouped into an enclosing activity partition with isDimension=true, whose name is the dimension name. Rather than being shown as a partition itself, however, the dimension is indicated by placing its name along side the set of partitions in the dimension.

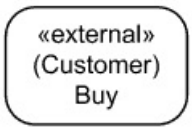
Partition could represent an **external entity** to which the partitioning structure does not apply. **External partitions** are intentional exceptions to the rules for partition structure. For example, a dimension may have partitions showing parts of a **structured classifier**. It can have an **external partition** that does not represent one of the parts, but a completely separate classifier. In **business modeling**, external partitions can be used to model entities outside a business.

When activities are considered to occur outside the domain of a particular model, the partition can be labeled with the keyword **«external»**. Whenever an activity in a swimlane is marked «external», this overrides the swimlane and dimension designation.



*Buy **action** occurs in external **partition** Customer*

In the situations when swimlanes can't be used to show partitions, alternate text notation with qualified action name could be used instead. In this case **partition** name is placed in parenthesis above the **action** name. A comma-delimited list of partition names means that the node is contained in more than one partition. A double colon within a partition name indicates that the partition is nested, with the larger partitions coming earlier in the name.

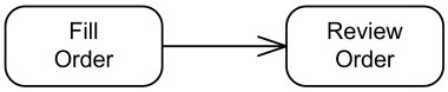


*Buy **action** occurs in the external **partition** Customer*

Activity Edge

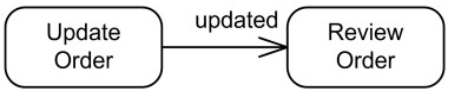
Activity Edge is an abstract class for the directed connections along which **tokens** or **data objects** flow between **activity nodes**. It includes **control edges** and **object flow edges**. The source and target of an edge must be in the same activity as the edge.

Activity edge is notated by an open arrowhead line connecting two activity nodes.



***Activity edge** connects Fill Order and Review Order.*

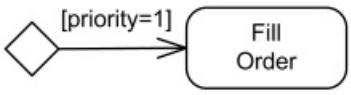
Edges can be named, however, edges are not required to have unique names within an activity. If the edge has a name, it is notated near the arrow.



***Activity edge** "updated" connects two nodes.*

Activity edge can have a **guard** - specification evaluated at runtime to determine if the edge can be traversed. The guard must evaluate to true for every token that is offered to pass along the edge.

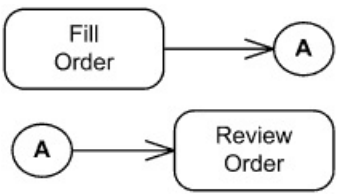
The **guard** of the activity edge is shown in square brackets that contain the guard.



Fill Order when priority is 1

An activity edge can be notated using a **connector**, which is a small circle with a **name** inside. Though UML 2.4 specification calls it **name of the edge**, provided connector notation and examples suggest that connector has its own name (also called **label**).

Connectors are generally used to avoid drawing a long edge. This is purely notational. It does not affect the underlying model. The circles and lines involved map to a **single activity edge** in the model. Every connector with a given label must be paired with exactly one other with the same label on the same activity diagram. One connector must have exactly one incoming edge and the other exactly one outgoing edge, each with the same type of flow, object or control.

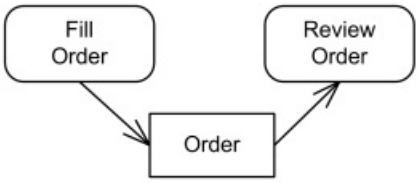


Connector A connects two edges between Fill Order and Review Order.

Object Flow Edge

Object flow edges are **activity edges** used to show **data flow** of **object** and data tokens between action nodes.

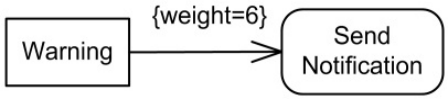
An object flow is notated by an arrowed line.



***Object flow** of **Orders** between Fill Order and Review Order actions*

Any number of tokens can pass along the edge, in groups at one time, or individually at different times. The **weight** attribute dictates the **minimum number** of tokens that must traverse the edge at the same time. When the minimum number of tokens are offered, all the tokens at the source are offered to the target all at once.

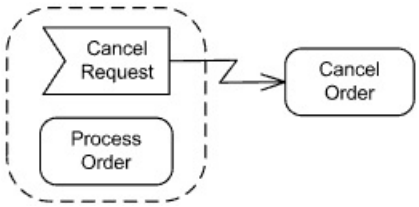
The **weight** of the edge may be shown in curly braces that contain the weight. The weight is a value specification, which may be a constant, that evaluates to a non-zero unlimited natural value. An **unlimited weight** is notated as "*".



Send Notification when number of Warnings reaches 6.

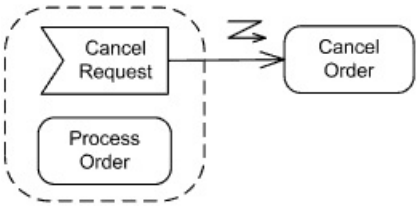
Interrupting Edge

Interrupting edge is activity edge expressing interruption for regions having interruptions. It is rendered as a lightning-bolt.



Cancel Request signal causes interruption resulting in Cancel Order.

An option for notating an interrupting edge is a zig zag adornment on a straight line.



Cancel Request signal causes interruption resulting in Cancel Order.

Noticed a spelling error? Select the text using the mouse and press Ctrl + Enter.

This document describes UML versions up to ***UML 2.5*** and is based on the corresponding **OMG™ Unified Modeling Language™ (OMG UML®)** specifications. UML diagrams were created in **Microsoft® Visio®** 2007-2016 using ***UML 2.x Visio Stencils***. ***Lucidchart*** is a nice, free UML tool that I recommend for students.
You can send your comments and suggestions to webmaster at **webmaster@uml-diagrams.org**.