BRITISH
COLUMBIA

ACTIVITY DIAGRAM MODELING

STANDARDS AND GUIDELINES

VERSION 1.0

DECEMBER 2, 2005

Information and Technology Management Branch

❖ **IM / IT Standards & Guidelines** ❖

Table of contents

Document Purpose

The purpose of this document is to provide a set of minimal acceptable standards and guidelines for modeling the *Activity Diagrams* based on UML 2.0. The document briefly describes the elements of a typical *Activity Diagram* followed by naming and modeling standards for the elements. The document assumes that the reader has prior knowledge of UML, Use case and Activity Diagrams.

Diagram Purpose

Activity Diagram is typically used for modeling the logic captured in a specific use case in a use case diagram. Activity diagram can also be used to model a specific Actor's workflow within the entire system. Activity diagram can also be used independent of use cases for other purposes such as to model business process of a system, to model detailed logic of business rules etc. Activity diagram shows all potential sequence flows in an activity.

Diagram elements

The Activity Diagram is comprised of the following model elements.

Initial State

An *initial state* is a model element that explicitly shows the beginning of a workflow on an activity diagram. It is the point at which reading of the activity diagram begins. Because activity diagram shows a sequence of actions, it must indicate the starting point of the sequence using the *initial state* element.

The *initial state* is drawn as a solid circle with an optional name or label : ● **Begin Loan Application Processing**

Action

An *Action* is a model element that represents the performance of a task in a workflow or operation. Each *action* must occur before the workflow continues to the next element on an activity diagram. An action is a non-interruptible task that takes place during an activity. *Action* may be nested - i.e. action may have sub-actions.

Action is drawn as a capsule shaped rounded rectangle with a name or description : **Loan Applicant submits online loan application**

Transition

"T*ransition*" element connects the various elements of the activity diagram. Typically the transition element represents the workflow between two or more actions or other elements of the activity diagram.

It is drawn as a solid line with open arrowhead : ⟶

"*Transition*" element can have an optional label enclosed in square brackets called "*guard condition*" or simply "*guard*", as shown below :
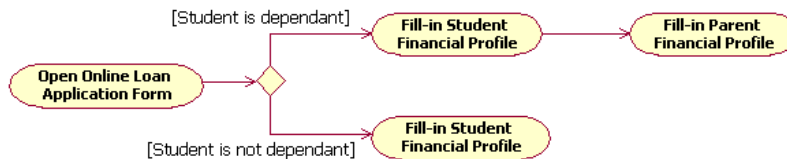

Guard conditions are used to define the conditional logic that controls the flow of control, and the specific criteria that must be met for a transition to occur.

Decision point

A *Decision* is a model element that typically has one incoming transition and two or more outgoing transitions based upon the outcome of guard conditions from the previous element.

Decision is drawn as a diamond shape usually without a name or label (no need for name/label because the guard conditions usually imply the reason for the decision) : ◇

Transitions that leave a *decision* model element often have guard conditions as shown in the example below.
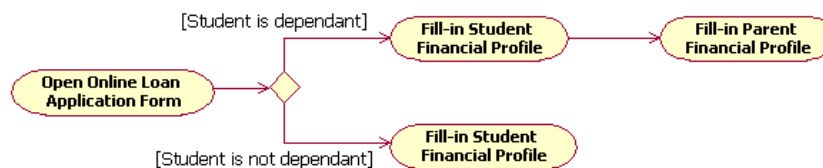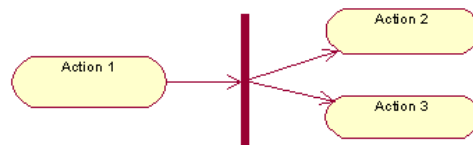

Synchronization

Synchronization is a model element that allows modeling of simultaneous workflow in an activity diagram - i.e. parallel activities. Synchronizations visually define forks and joins that represent parallel workflow or execution.

Synchronization is drawn as a horizontal or vertical bar with no name or label :

or

Fork

Fork is a kind of synchronization model element that facilitates the modeling of simultaneous workflow in an activity. A fork identifies where a single flow of control divides into two or more separate, but simultaneous flows. *Fork* is drawn as a bar with one transition going into it and two or more transitions leaving it:

Join

Join is another kind of synchronization model element that facilitates the modeling of simultaneous workflow in an activity. A join identifies where two or more simultaneous flows of control unite into a single flow of control. *Join* is drawn as a bar with two or more transitions going into it and one transition leaving it :
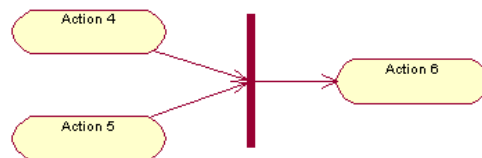
Swimlane

A *swimlane* is a model element that can represent a user, an organizational unit, or a role in an activity diagram. *Swimlanes* depict who or what is responsible for carrying out specific activities. Swimlane enables grouping of actions on the activity diagram performed by the same actor or by a single thread. Action transition can take place between two swimlanes.

Swimlanes are drawn as vertical columns or horizontal rows with a name associated with each swimlane, as shown in the following examples :

| [vertical swimlanes example] | [horizontal swimlanes example] |
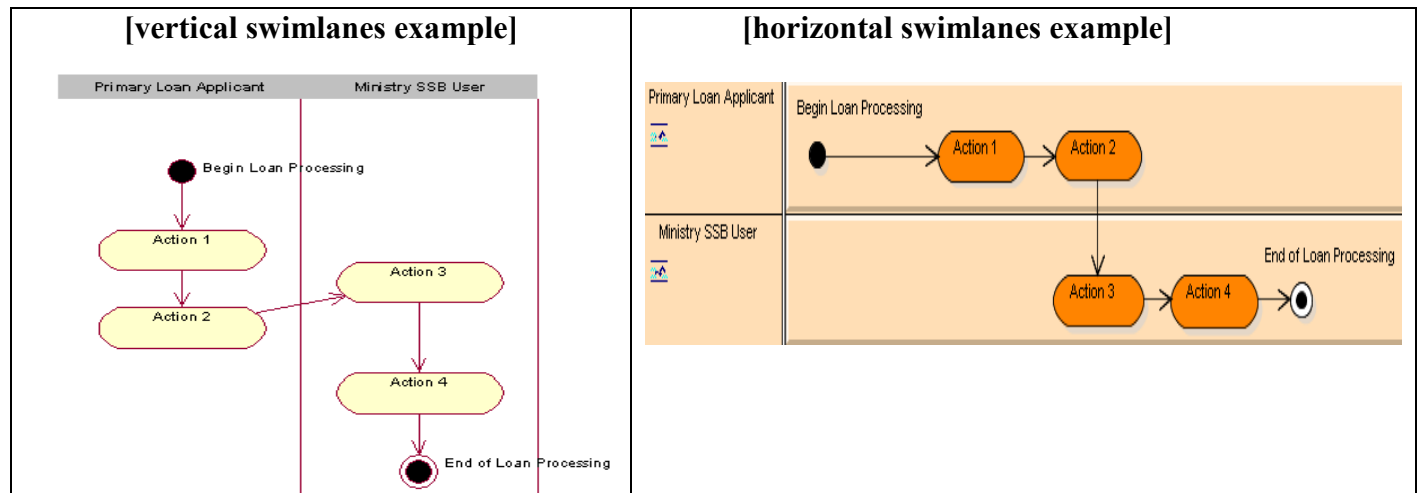
Final state

A *final state* is a model element that explicitly shows the end of a workflow on an activity diagram. Unlike initial state, there can be multiple final states in an activity diagram to indicate termination of specific branches of workflow.

Final state is drawn as a filled circle inside a larger unfilled circle with an optional name or label :

Naming standards

Initial State Naming

- It is a good practice to name the initial state although it is not mandatory.
- The name of the initial state should describe its purpose.
- Always use verbs to name the initial state such as "begin pre-screening", "start loan application processing".

Action Naming

- The name of the *Action* should describe its purpose.
- It is a good practice to name the *Action* starting with a verb and indicating the activity that takes place in the *action* such as "Fill-in Parent Financial profile" as shown in the earlier examples.

Modeling standards

Activity Diagram modeling

- Always model Activity diagrams per Use case and do not mix two use cases' flow into a single activity diagram. An exception to this is when the Activity diagram is drawn for an actor or system as a whole.
- Always draw separate activity diagram for basic flow and alternate flows of a use case. Do not mix the two.
- Ensure that the activity diagram of a use case reflects business flow and business terminology rather than system flow and technical terminology.

Initial State modeling

- There must be only *one initial state* element drawn in an Activity diagram.
- If *swimlanes* are used, it is customary to place the *initial state* element in the first swim lane.
- In case multiple nested Activity diagrams are drawn, each activity diagram can have its own new initial state or the same common initial state, depending on the context.
- The initial state should be connected only to the "*Action*" element of the activity diagram and not to any other element.
- The initial state must be connected to only one *action* element and not to multiple action elements.

Transition modeling

- Transition element has no name.
- Guard condition can be specified as a mathematical/logical expression (such as "$x>=0$ *and y=true*") or as a simple sentence that specifies the condition (such as "*loan applicant is a dependant*").
- Ensure that guard condition always evaluates to "true" or "false" value.
- Do not specify any guard condition for transition that departs (leaves) from *initial state*.
- Each transition leaving a decision point must have a guard condition.

- Ensure that guard conditions do not overlap as in the following example, where it is not clear what should happen when x=0.

$$[x <= 0 \text{ and } x >= 0]$$

- Specify an "else" guard condition for fail-through logic of *decision points*.

Action modeling

- Avoid modeling of nested actions (action within action) if it is not adding value to the Activity diagram.
- Avoid modeling "black box" *Actions* - i.e. the actions that have transitions into them but no transitions out of them.
- Avoid "miracle" *Actions* - i.e. the actions that have transitions out of them but no transitions into them.

Decision modeling

- *Decision* element should reflect the previous *action*.
- Avoid Superfluous Decision Points.

Fork and Join modeling

- For every *Fork* there must be a *Join*.
- Ensure that Forks have only one entering transition and two or more leaving transitions.
- Ensure that Joins have two or more entering transitions and only one leaving transition.
- Avoid Superfluous Forks and Joins so as not to clutter the diagram.

Swimlane modeling

- Arrange swimlanes in a logical order although order is not important.
- As a thumb rule, apply swimlanes only to linear processes.
- As a thumb rule, avoid using more than five swimlanes in a single activity diagram.

Final State modeling

- Always provide a name/label for the final state.
- The Name/label of the *final state* should reflect the end of the workflow such as "End of loan processing".

Specification standards

Unlike Use case diagram, the Activity diagram has no specification or template explicitly for documentation purposes. As such, there is no need for documenting the Activity diagrams beyond what is done in the diagram itself. However, most UML tools provide in-built documentation capturing and printing capabilities for the Activity diagram and its elements. Use this facility to document the activity diagram, if needed.