BPMI.org

BPMI Notation Working Group

Execution Level Notation Concepts

February 12, 2002

BPMI Document Number: NWG-2002-02-02
Revision 2

Notation Concepts

1.0 The Execution Level notation will depict, through a series of connected lines, the simple flow of control of a business process, from its start to its end, without any disconnected elements. The connected lines will be called control flow links. *<<This may be modified upon vote of the Working Group>>*

1.1 There will be only 1 (one) Control Flow Link connecting to a given Activity.

1.2 There will be only 1 (one) Control Flow Link connecting from a given Activity.

1.3 Control Flow Links cannot cross the boundary of an Activity Block (as defined in item 3.0). They can connect to an Anchor of an Activity Block (boundary).

2.0 All atomic activities will have the same basic shape and that shape will be easily distinguishable from activity blocks.

2.1 Activity blocks *will not* graphically contain other BPML Activities within its borders.

2.2 Different atomic activities will have a graphical mechanism (such as a marker), which doesn't dramatically alter the basic shape, but allows the reader of the model to be able to distinguish them.

3.0 All activity blocks will have the same basic shape and that shape will be easily distinguishable from atomic activities.

3.1 Activity blocks *will* graphically contain other BPML elements within its boundaries. The internal elements will be connected by control flow links.

3.1.1 Some of the internal control flow links will be specialized so that they provide additional flow information, such as the name or expression of a BPML switch case condition.

3.2 Activity blocks will have a Start Anchor and an End Anchor to show that boundaries of the block.

3.2.1 A Control Flow Link will connect to a Start Anchor to show the flow to a block.

3.2.2 One or more Control Flow Links will connect from a Start Anchor to show the performance of the Activities within the block

3.2.3 One or more Control Flow Links will connect to an End Anchor to show the conclusion of the Activities within the block.

3.2.3.1 The same number of Control Flow Links that leave a Start Anchor must enter the corresponding End Anchor for a given block.

3.2.4 A Control Flow Link will connect from an End Anchor to show the flow from a block.

3.3 Activity blocks will *optionally* have a graphical perimeter that surrounds its boundaries *<<this may be modified upon vote of the Working Group>>*.

3.3.1 If a perimeter is not shown, then the model can be more free-form with the start and end anchors showing the boundaries of the block. However, the same semantics must apply such that illegal connections between Activities should not be allowed. E.g., an Inner Activity Block must be completely closed before an outer Activity Block can be closed. Such semantics are obvious with visible

perimeters around the Activity Blocks, but are not as obvious when the perimeters are not visible.

3.4 Different activity blocks will use variations of the shape of the Anchors to allow the reader of the model to be able to distinguish them.

3.4.1 ¿Should the Start Anchors be graphically distinguished from the End Anchors? I.e., if the same shape is used for both the Start and End Anchors (e.g., a circle), should there a way to differentiate them, especially if the Activity Block perimeter is not visible?

4.0 If an ActivitySet uses a Context for Event Handling, then the graphic perimeter of the Context *must* be visible.

4.1 If the Context is in an Activity Block that also is displaying its perimeter, then the perimeter of the Activity Block and the Context will exactly overlap.

4.2 The Event Anchor(s) will be attached to the Context perimeter in a location that is offset from any Start or End Anchors that are present.

4.3 An Event Flow Link, which is outside the normal Control Flow, will connect an Event Anchor of a Context to an Event Handler element for that Context.

4.3.1 Event Flow Links cannot cross the boundary of an Activity Block (as defined in item 3.0). They can connect to an Anchor of an Activity Block (boundary).

5.0 If a Context is defined as being a Transaction, then the graphic perimeter for the Transaction will be visible and will not exactly overlap with the perimeter of its Context.

5.1 The Transaction Anchor, if used, will be attached to the perimeter of the Context for the Transaction in a location that is offset from any Start or End Anchors that are present.

5.2 Atomic Transactions should be distinguished from Open Transactions through some graphic marker or text.

5.3 A Transaction Failure Link, which is outside the normal Control Flow, will connect the Transaction Anchor of a block to the Compensation element for that block.

5.3.1 Transaction Failure Links cannot cross the boundary of an Activity Block (as defined in item 3.0). They can connect to an Anchor of an Activity Block (boundary).

6.0 External “participants” can be depicted. Internal “participants” will not be depicted.

7.0 Message Flow to external “participants” can be depicted. Messages to internal “participants” will not be depicted.

7.1 The direction of message flow will be shown with a Message Link.

7.1.1 This type of link will be distinguishable from the types of links.

7.1.2 The contents of the message will not be depicted as a separate object, but will be associated with the Message Link.

7.1.3 There will be a maximum of 1 (one) Message Flow Link connecting from a given Action.

7.1.4 There will be a maximum of 1 (one) Message Flow Link connecting to a given Action.

7.1.5 There can be multiple Message Flow Links connecting to a Participant.

7.1.6 There can be multiple Message Flow Links connecting from a Participant.

7.1.7 Message Links can cross the boundary of an Activity Block.

8.0 Nested processes, although disconnected from the main flow of control, should be represented and should be easily distinguishable from activity blocks and atomic activities.

Metamodel Conventions

1.0 Classes that have a stereotype and a color represent elements that will be graphically depicted in the notation. The other classes are used to help define and organize the elements that will be depicted.

1.1 The stereotypes represent general types of graphics. There may be variations of a general type. For example, the DirectedLine stereotype will have different variations of a line with an arrowhead for control flow, message flow, etc.

2.0 Some classes have properties that will be displayed in the notation. For example, elements that have names that can appear on the model will have a Name property. Displayed properties will be defined as public properties (shown with a "+" in front of the property name). Other properties of a class may not be displayed. These will be defined as private properties (shown with a "-" in front of the property name).

2.1 Private properties will generally be available for modification through a modeling tool. All relevant Properties should be included in the metamodel.

2.2 ¿How can we show that some public properties are optionally depicted and some are mandatory?