Activity Diagrams

MDT Papyrus Tutorial :
Using Activity Diagram

vincent.hemery@atos.net

Atos

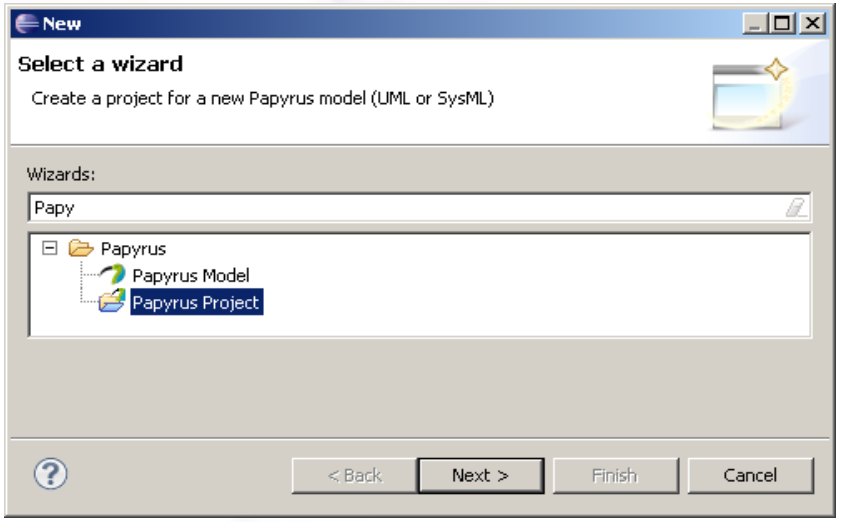
Agenda.

- This tutorial contains the following sections :
  - Create a Papyrus project with an Activity Diagram.
  - Create a Papyrus model with an Activity Diagram.
  - Create an Activity Diagram in an existing Papyrus model.
  - Activity's properties.
  - Create an activity node.
  - Edit an activity node's name.
  - Create an Activity Edge.
  - Activity edges.
  - Control nodes.
  - Object nodes.
  - Actions.
  - Activity groups.
  - Exception Handlers.
  - Comments.

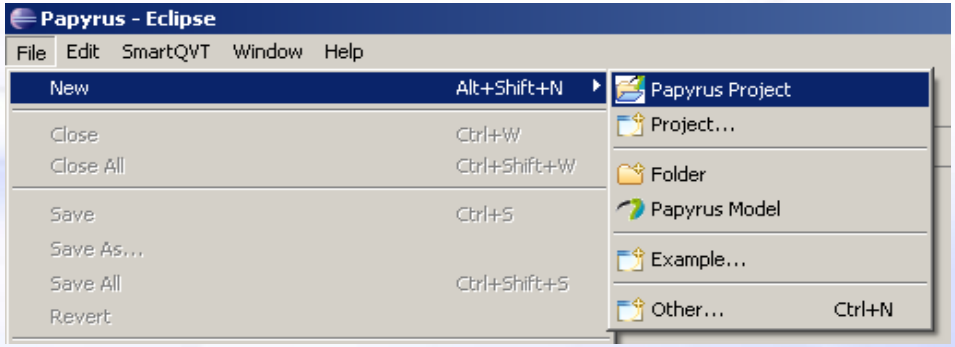
Atos

Create a Papyrus project with an Activity Diagram (1/3).

- Select the Papyrus perspective.
- Create a Papyrus project by :
  - Using the creation wizard :
    - Right click in Project Explorer view,
    - Select « New » > « Other ... »,
    - Choose « Papyrus Project »,

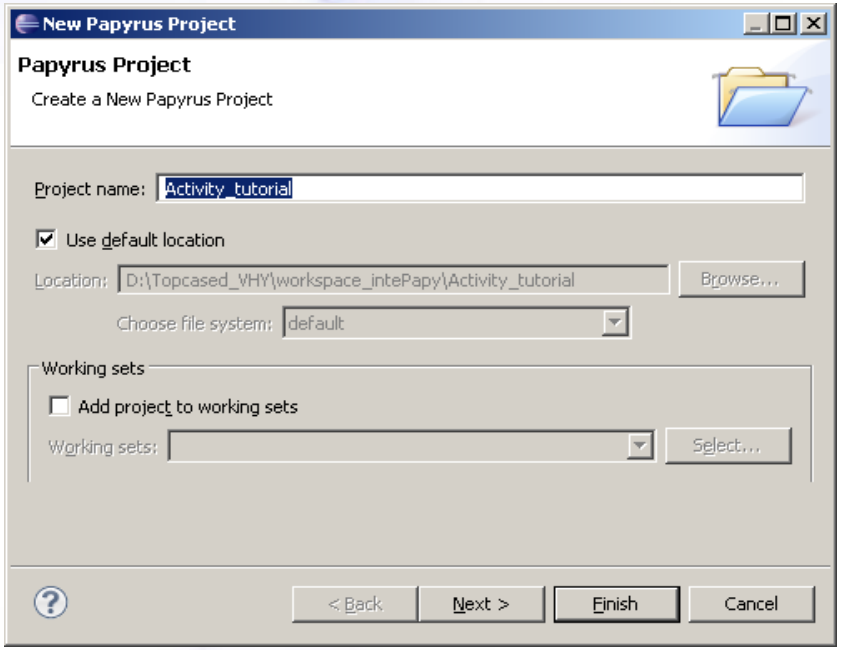


  - Or using the file menu action,

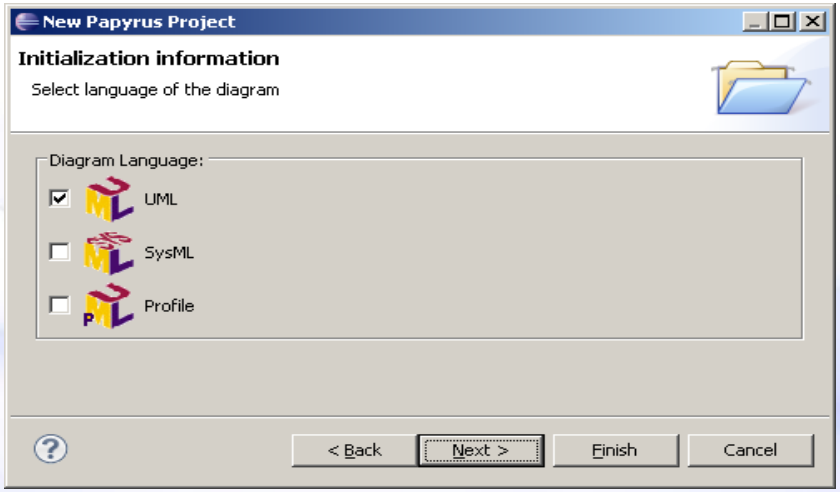


Create a Papyrus project with an Activity Diagram (2/3).

- Fill the project name,

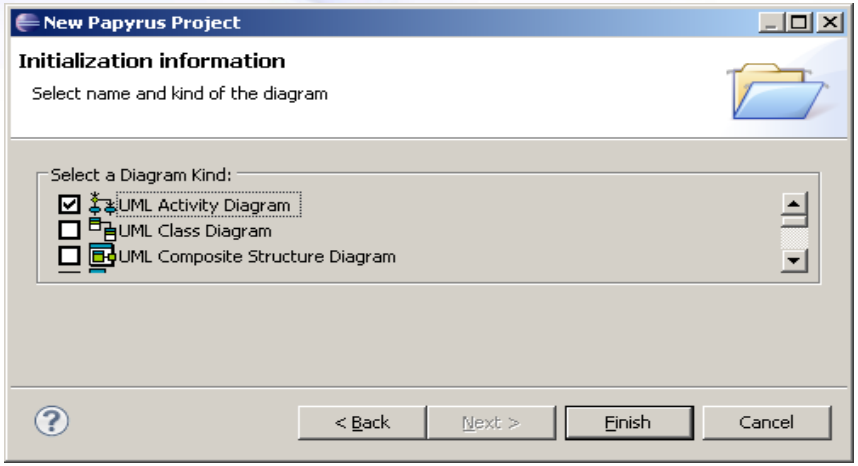


- Select UML language,

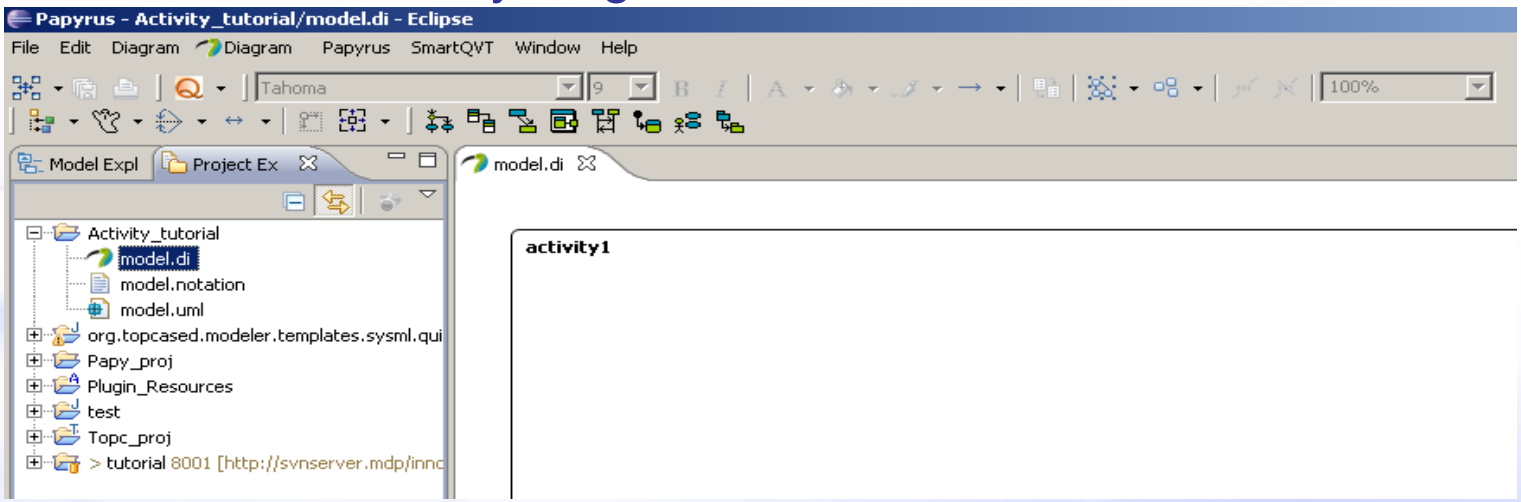


Create a Papyrus project with an Activity Diagram (3/3).

- Select the UML Activity Diagram.

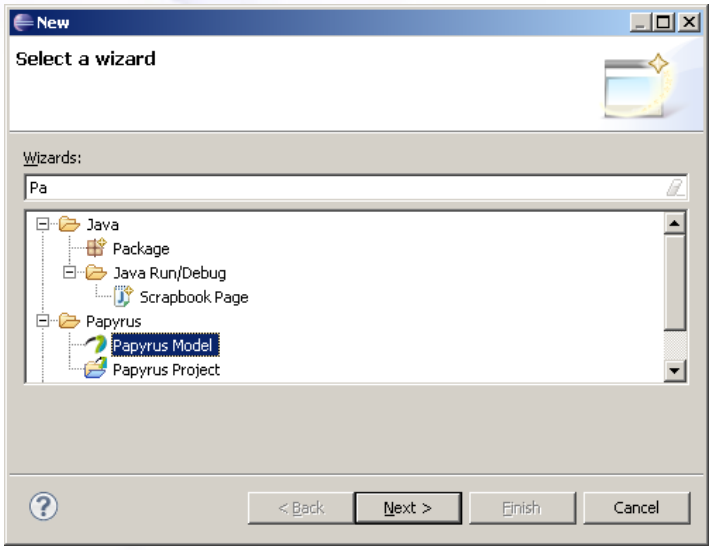


- A new project is initialized with a model and an Activity diagram.

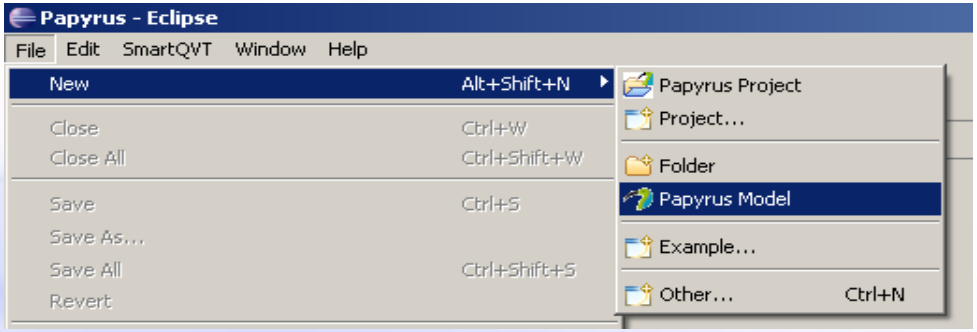


Create a Papyrus model with an Activity Diagram.

- You may also create a new Papyrus model in an existing project :
  - Using the creation wizard :
    - Right click in Project Explorer view,
    - Select « New » > « Other... »,
    - Choose « Papyrus Model »,



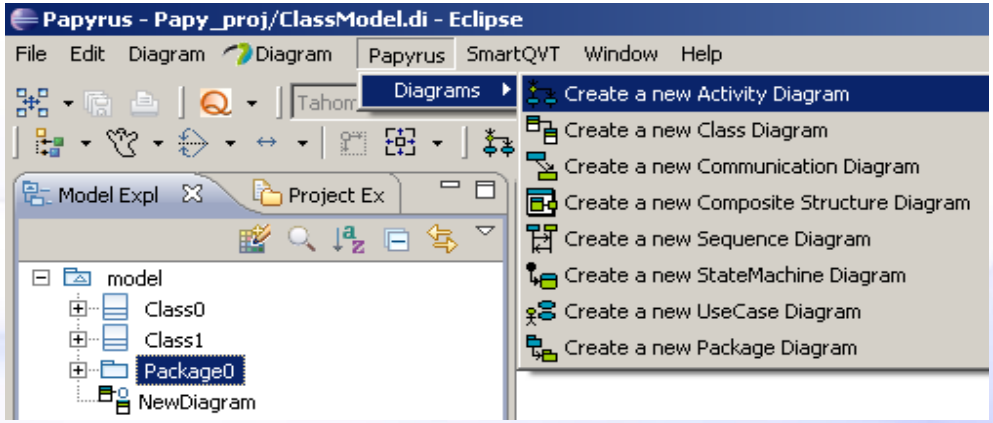
  - Or using the file menu action.

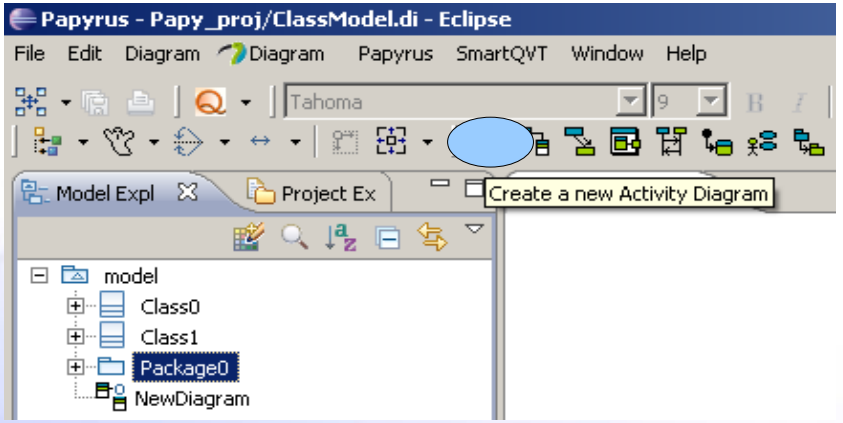


- Then, fill the model name,
- Then follow the same steps as in the project creation.

Create an Activity Diagram in an existing Papyrus model.

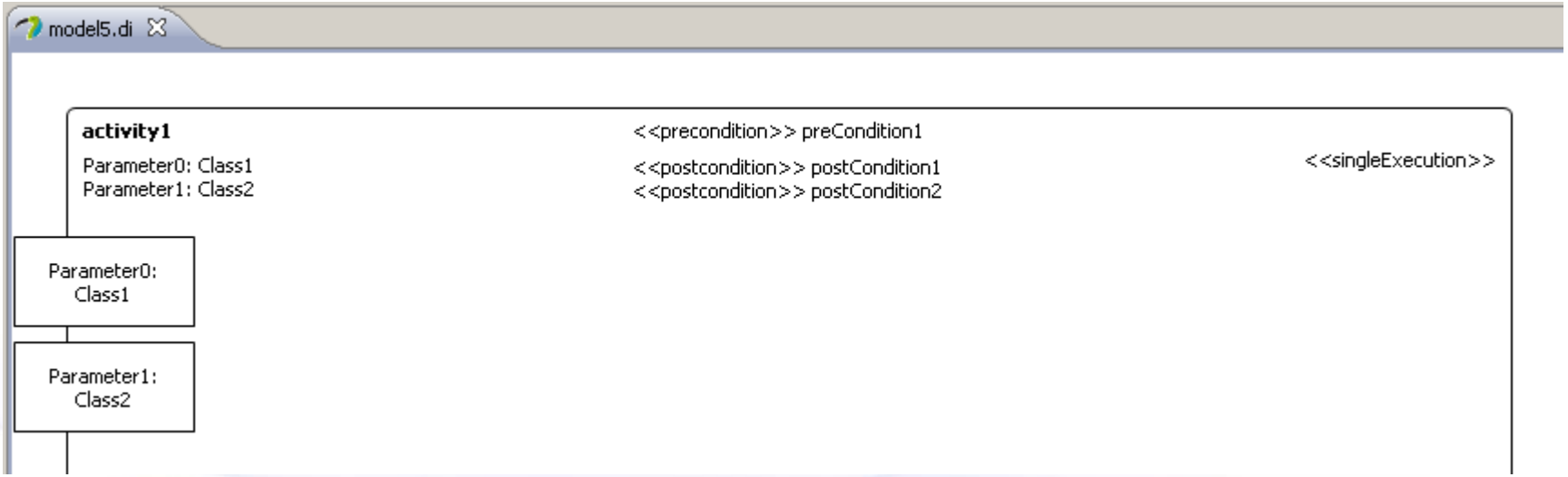
- You may also add an Activity Diagram to an existing model :
  - In the Model Explorer view, select the place where to add your diagram
  - Select the Activity Diagram menu or click on the corresponding toolbar action.





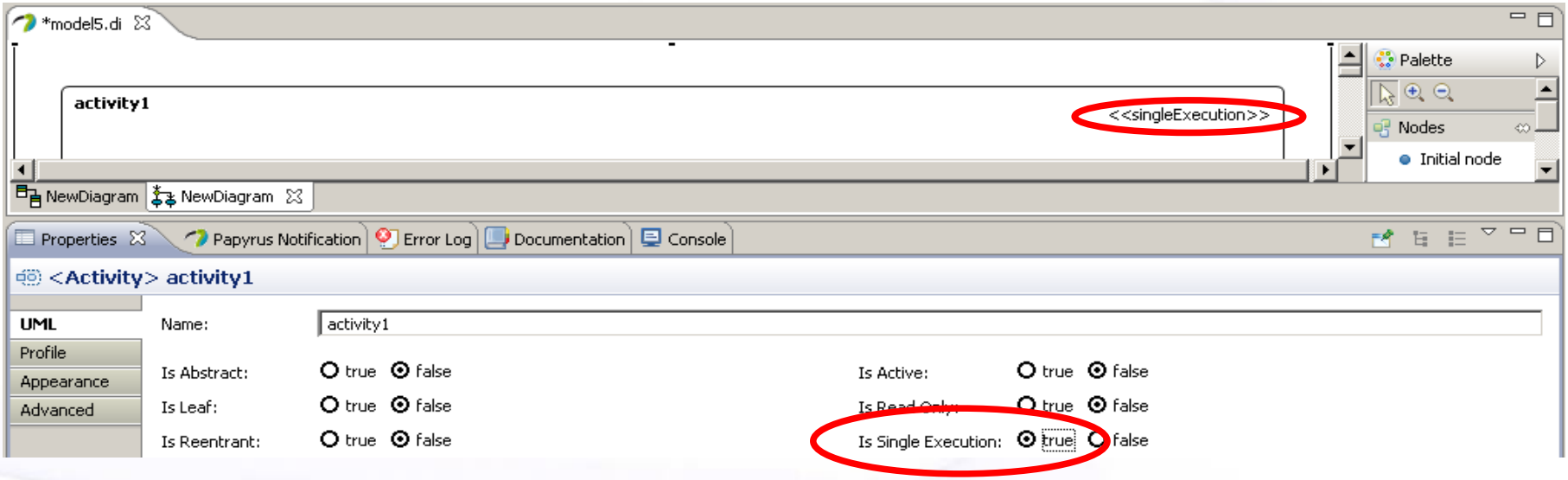
Activity's properties (1/4).

- An activity displays several properties in its heading part.
  Here is an example with all these properties displayed :



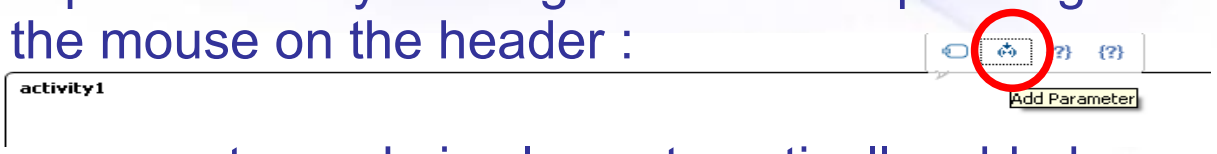
Activity's properties (2/4).

- The « singleExecution » label appears when the corresponding properties is activated in the Properties view.

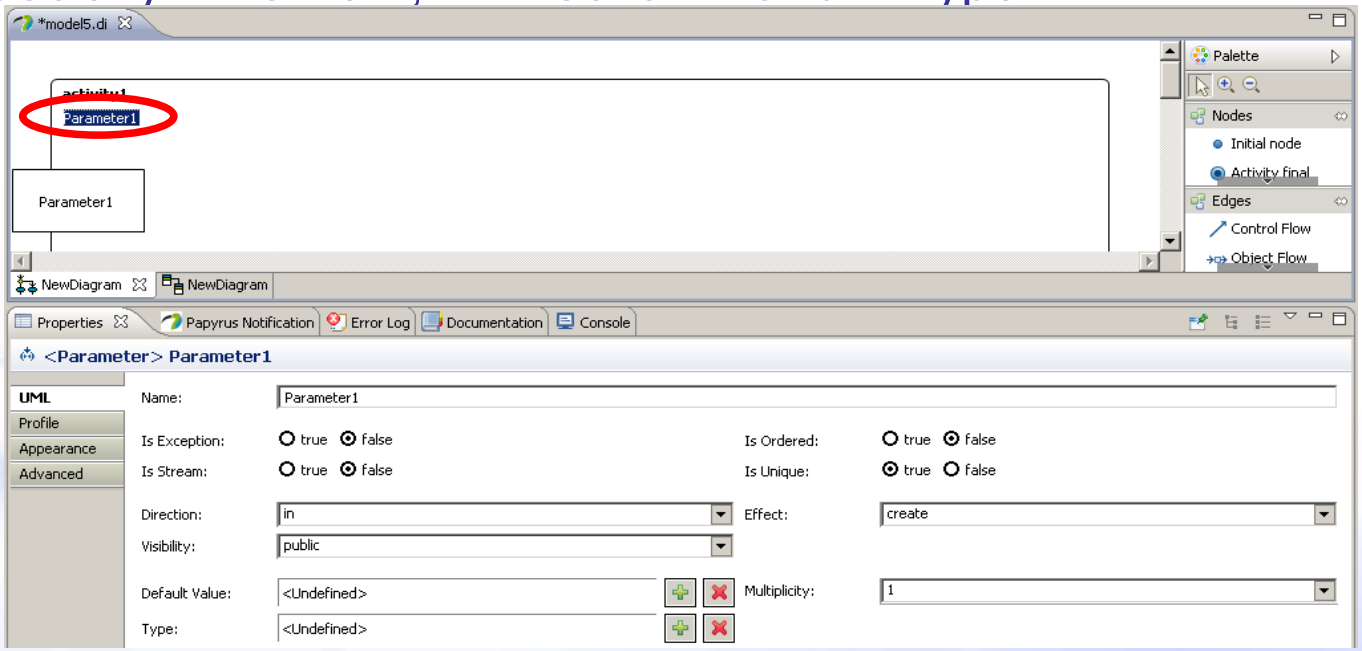


Activity's properties (3/4).

- You can add a parameter by clicking on the corresponding tooltip action, after leaving the mouse on the header :

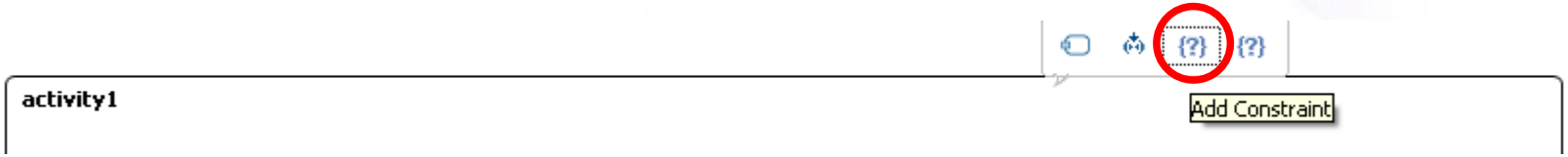


  - An activity parameter node is also automatically added.
  - Select the parameter and use the Properties view to edit its properties, especially « Name », « Direction » and « Type ».

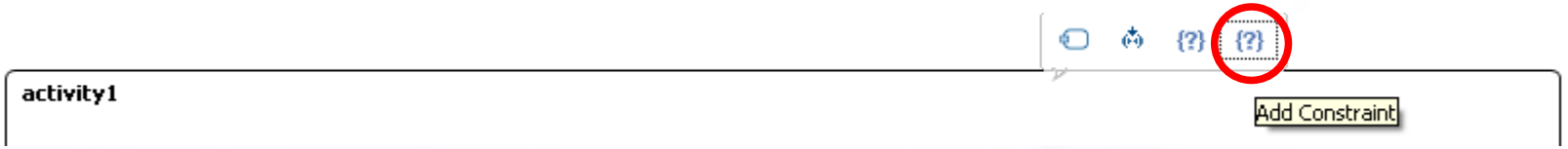


Activity's properties (4/4).

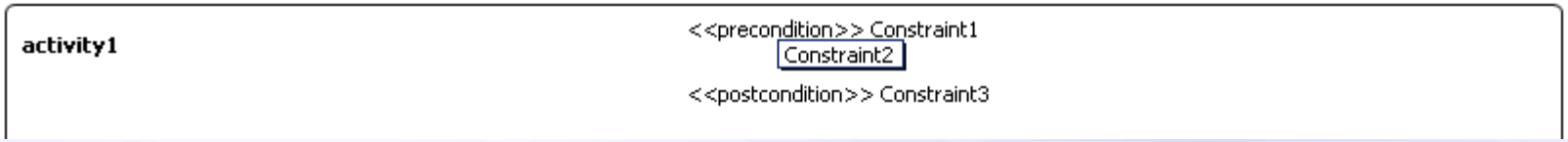
- You can add a precondition by clicking on the corresponding tooltip action, after leaving the mouse on the header :



- You can add a postcondition by clicking on the corresponding tooltip action, after leaving the mouse on the header :

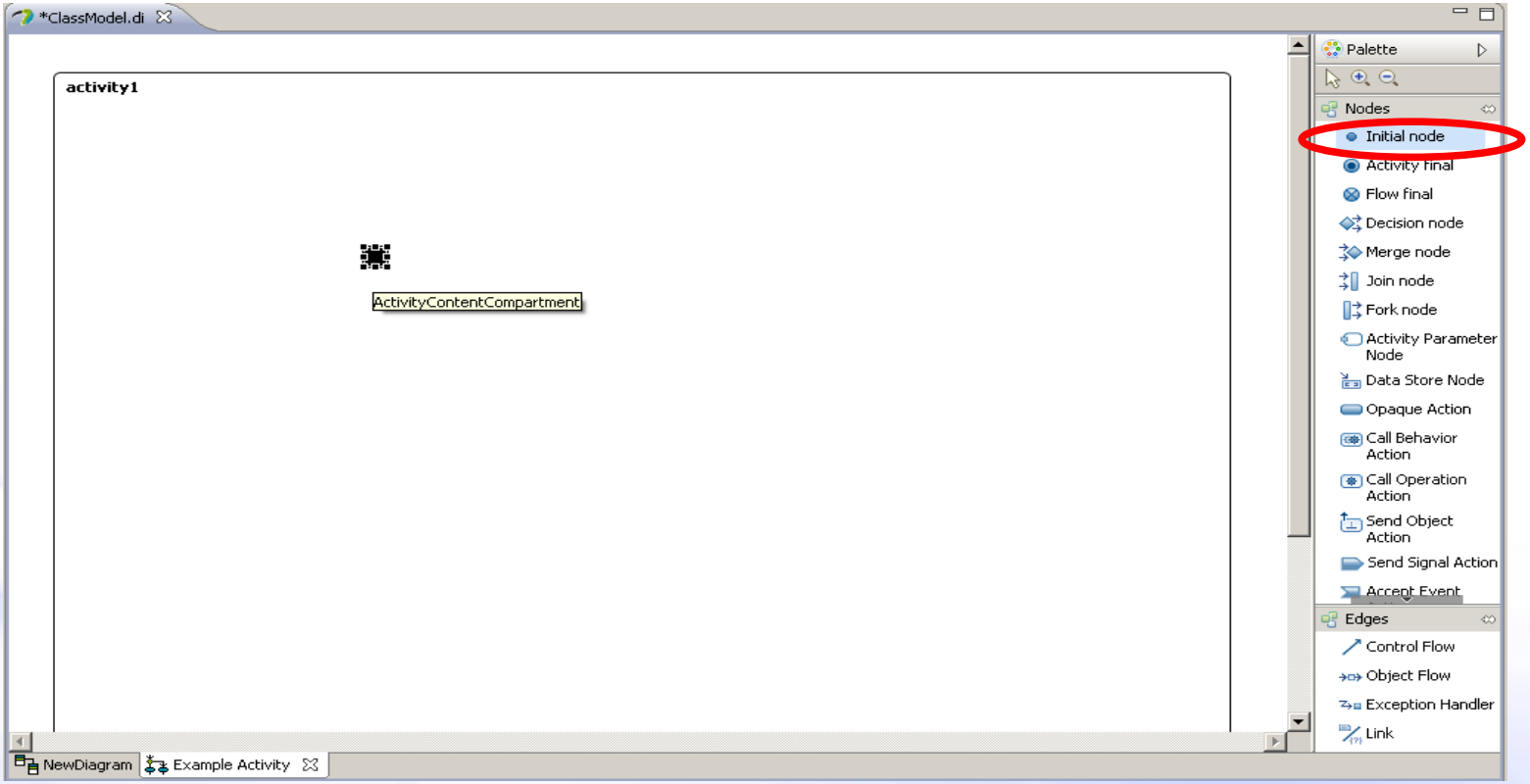


  - Select the added constraint label to edit its name.



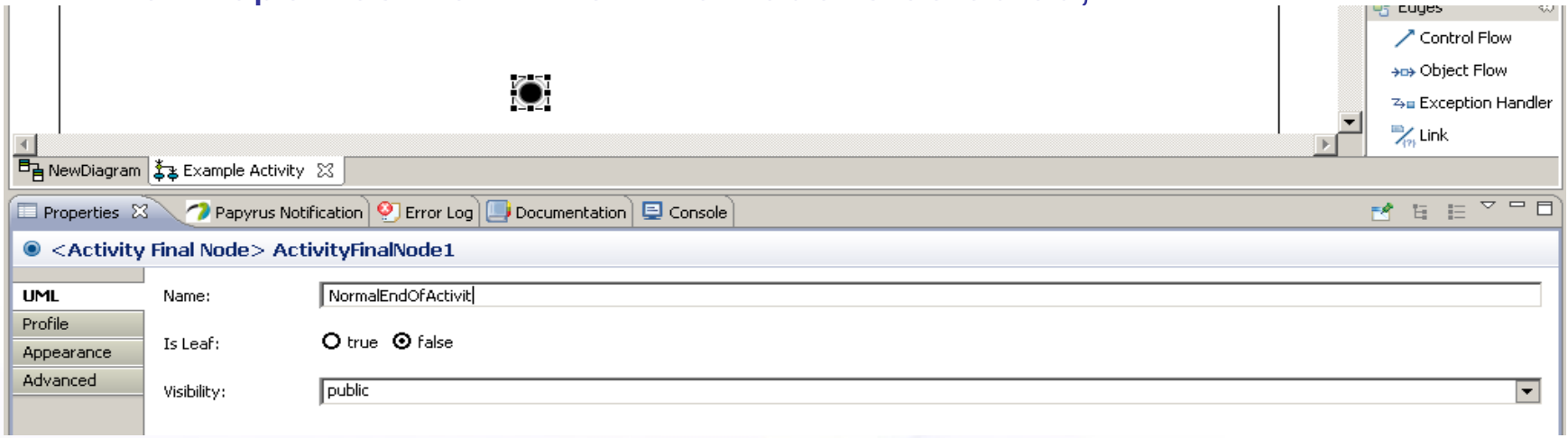
Create an activity node.

- You may create nodes in the activity :
  - Select the appropriated node in the Palette,
  - Click in the activity where you want to draw it.



Edit an activity node's name.

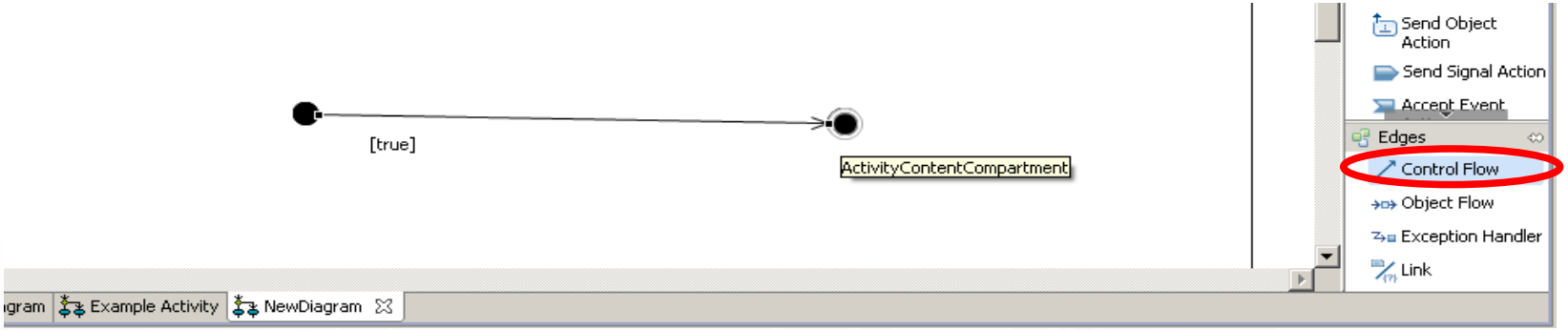
- You may edit the name of an activity node :
  - In the Properties view when the node is selected,



  - Directly by editing the label if it displays one.

Create an Activity Edge.

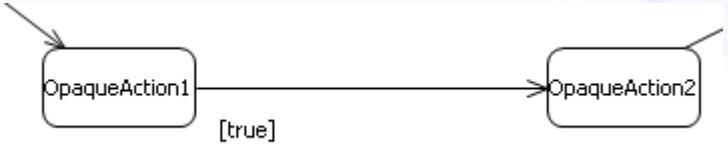
- You may create an activity edge between two activity nodes :
  - Select the appropriated edge in the Palette,
  - Click in the activity, from the source node to the target node, without releasing the mouse button.



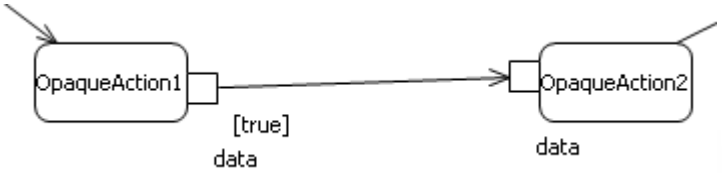
- Then, you may edit the edge's name in the properties view (see previous slide for activity nodes).

Activity edges.

- There are two kinds of activity edges :
  - Control flow :



    - This kind of edge passes control tokens which allow an activity node to start after another one. They can not carry objects nor data.
  - Object flow :

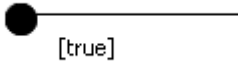


    - This kind of edge passes object and data tokens.
    - When it links two object nodes, these must have compatible types, in order that the tokens can pass from one to another through the object flow.
- Most activity nodes have restrictions on the kind of their incoming and outgoing edges. If you can not draw an edge between two nodes, check whether there are specific restrictions.

Atos

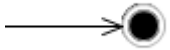
Control nodes (1/4).

- Control nodes allow to structure the activity. These are :
  - Initial node : allows to start the activity.



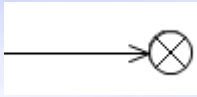
    - No edge can be drawn to an initial node, and only control flows can be drawn from an initial node.

  - Activity final node : allows to end immediately the activity.



    - No edge can be drawn from an activity final node.

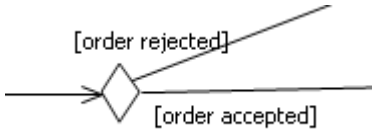
  - Flow final node : allows to destroy all incoming token (the activity is not ended if other tokens still exist in other flows)

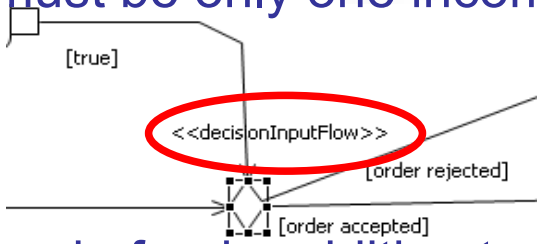


    - No edge can be drawn from a flow final node.

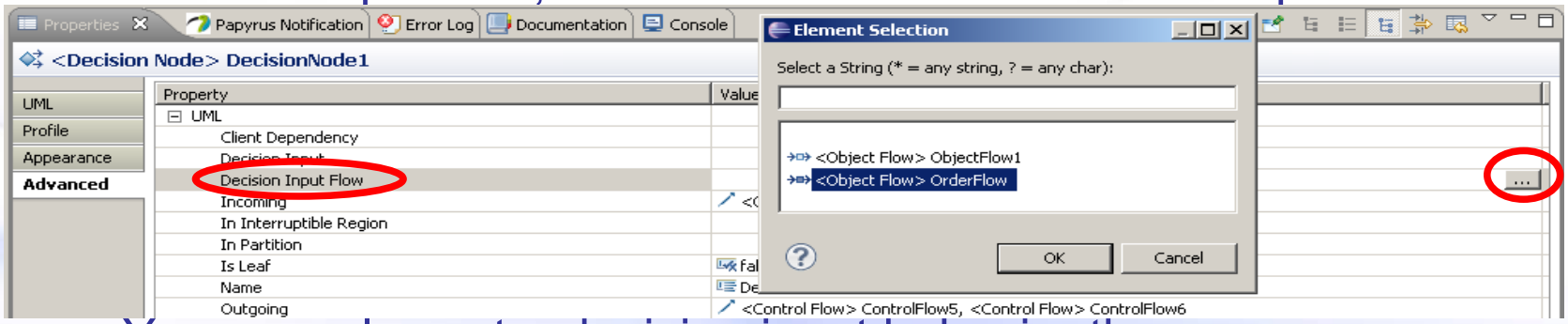
Control nodes (2/4).

- Decision node : allow to choose one among several outgoing edges.
  - Generally, incoming and outgoing edges must be of the same kind (object XOR control flows) and there must be only one incoming edge.





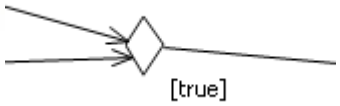
  - There may be an input object flow, used as an input for the decision, instead of or in addition to the usual incoming edge. This one has a specific label. To set this decision input flow, use the Advanced tab of the Properties view :



  - You may also set a decision input behavior the same way.
    - (see UML specification for more details on the semantic)

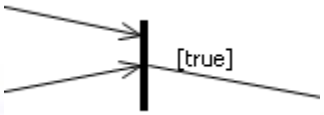
Control nodes (3/4).

- Merge node : allow to transfer tokens coming from several edges.



  - Incoming and outgoing edges must be of the same kind (object XOR control flows).
  - There must be only one outgoing activity edge.

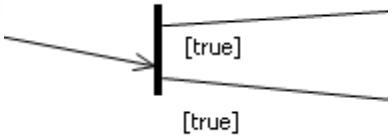
- Join node : allow to wait for all incoming token before processing.



  - There must be only one outgoing activity edge.
  - The outgoing edge is an object flow when there is at least one incoming object flow (to accept incoming data token carried by object flows).

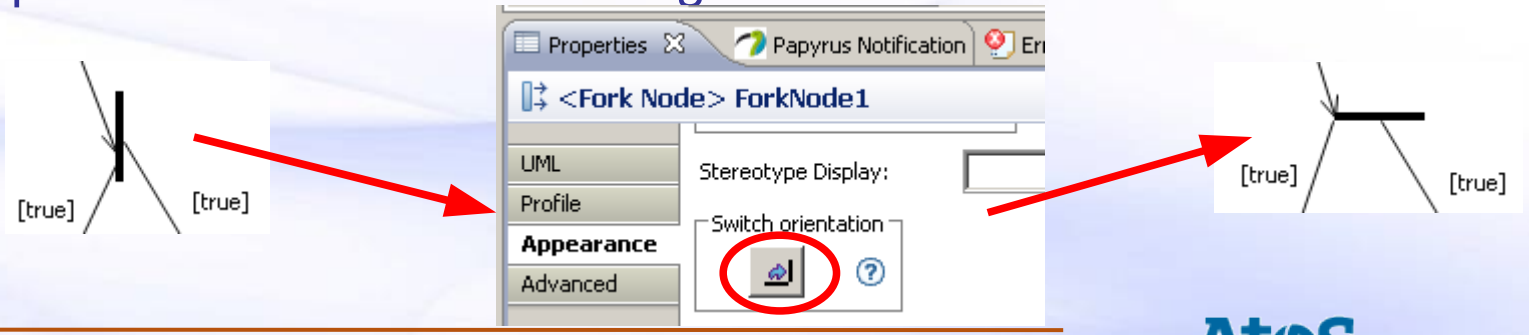
Control nodes (4/4).

- Fork node : allow to duplicate a token on all outgoing edges.



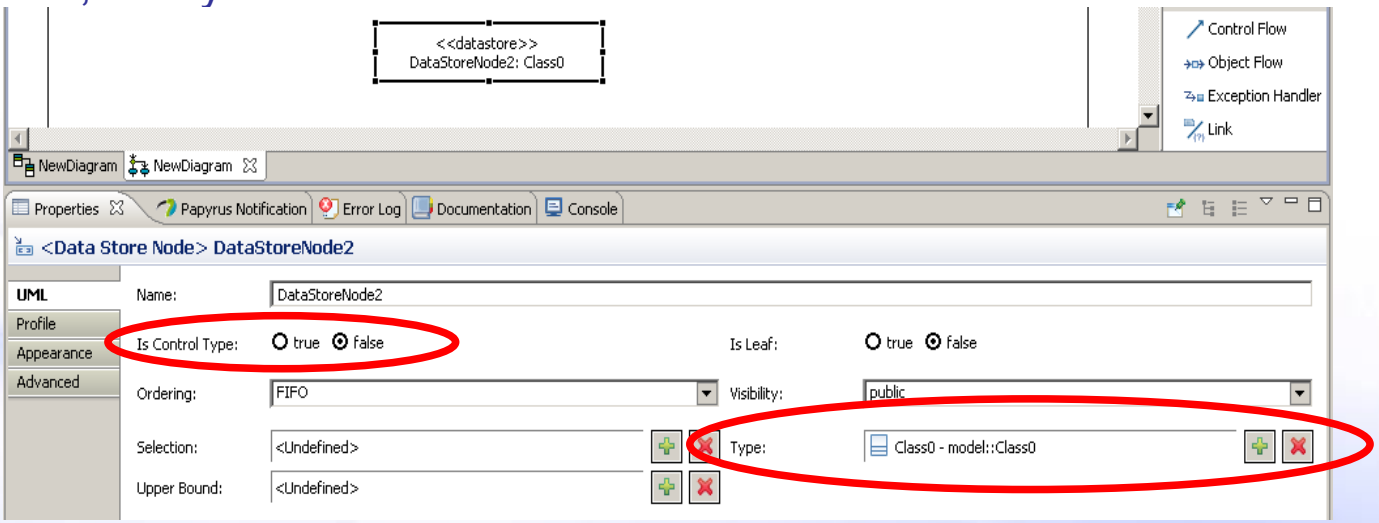
  - incoming and outgoing edges must be of the same kind (object XOR control flows).
  - There must be only one incoming activity edge.

- On join and fork nodes, you can use a button in the Appearance tab of the Properties view to switch the segment's orientation :



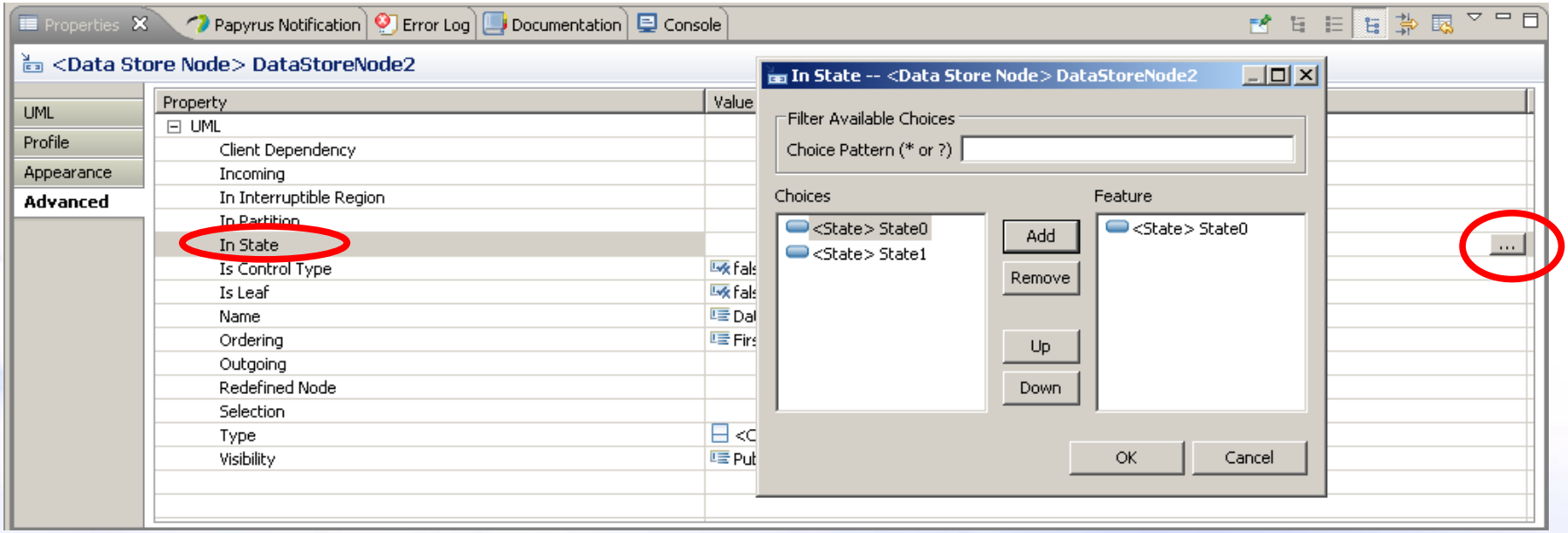
Object nodes (1/4).

- Object nodes indicate an instance of a particuliar classifier may be available at a particular point in the activity.
- Since object nodes handle data or object token, you can generally only draw object flows from or to it.
- In the Properties view, you can edit the object node's properties.
  - « Type » specifies the classifier.
  - « Is Control Type » specifies that tokens will be handled like control tokens. Hence, that you can draw control flows.



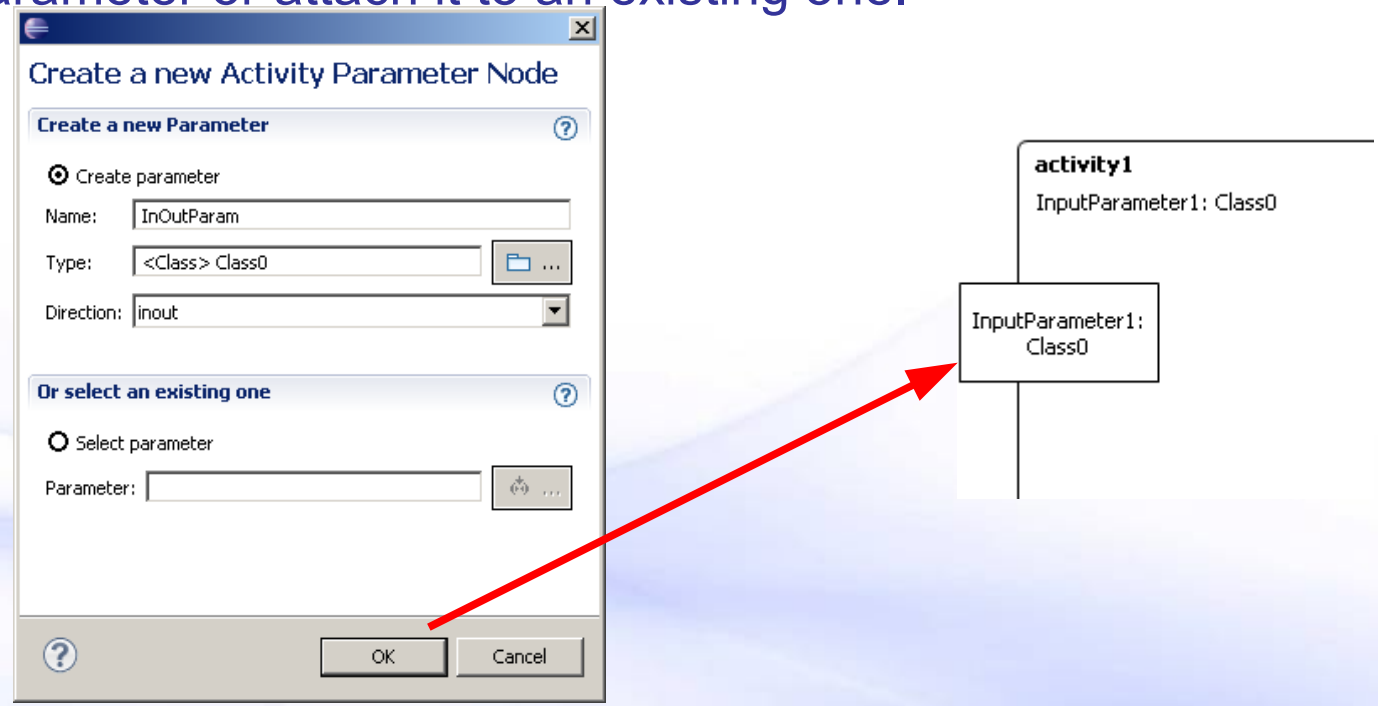
Object nodes (2/4).

- In the Advanced tab of the Properties view, you can edit the « In State » property.
  - The assigned states must have been previously created in a state machine.
  - This indicates the required states for the object handled by the object node.



Object nodes (3/4).

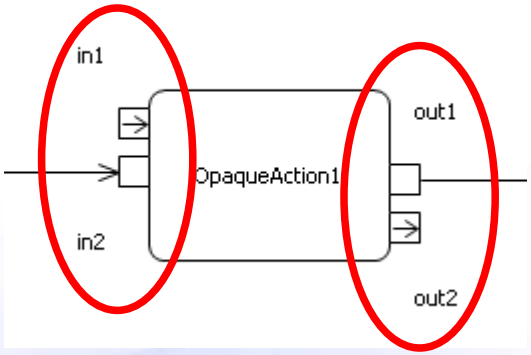
- These object nodes are available in the activity diagram :
  - Activity parameter node : provides inputs and outputs to the activity.
    - Activity parameter nodes are mapped on the activity's parameters. When creating a new one, you will be asked to either create a new parameter or attach it to an existing one.



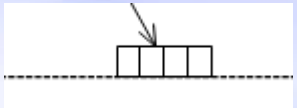
Object nodes (4/4).

- Data store node : stores incoming values before sending a copy of them on outgoing edges.

- Pin (output pin, input pin, action input pin, value pin) : provides inputs and outputs to an action. These may be constrained by the owning action.



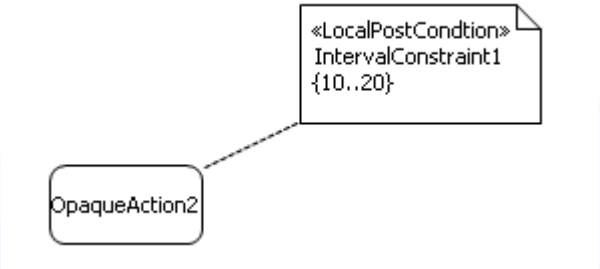
- Expansion node : provides inputs and outputs to an expansion region.



Actions (1/7).

- Actions are activity nodes representing an individual step within an activity.

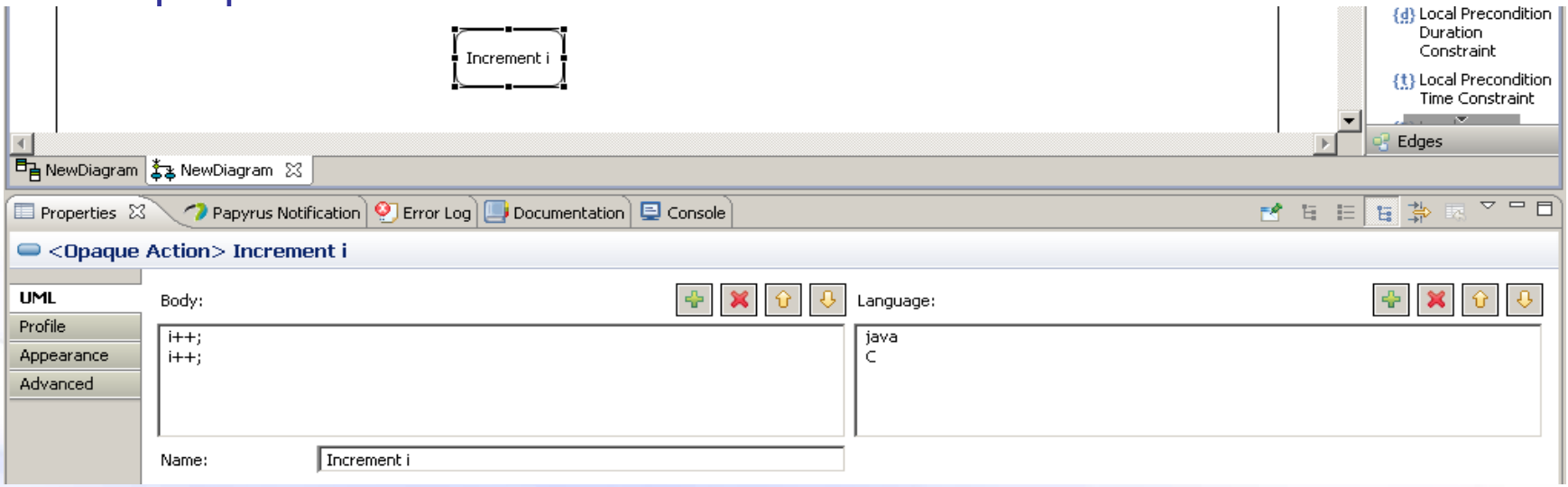
- Local pre and post conditions can be created on an action :
  - Select the appropriate local condition tool in the palette,
  - Click on the action,



  - You can then edit the condition's name and expression.

Actions (2/7).

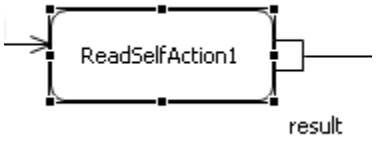
- You may create the following actions :
  - Opaque action : specifies an action with implementation-specific semantics.
    - Use the Properties view to set the « Body » and « Language » properties.



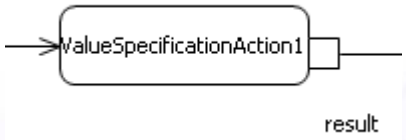
    - You may also use an opaque action if you do not want to specify the action's implementation.

Actions (3/7).

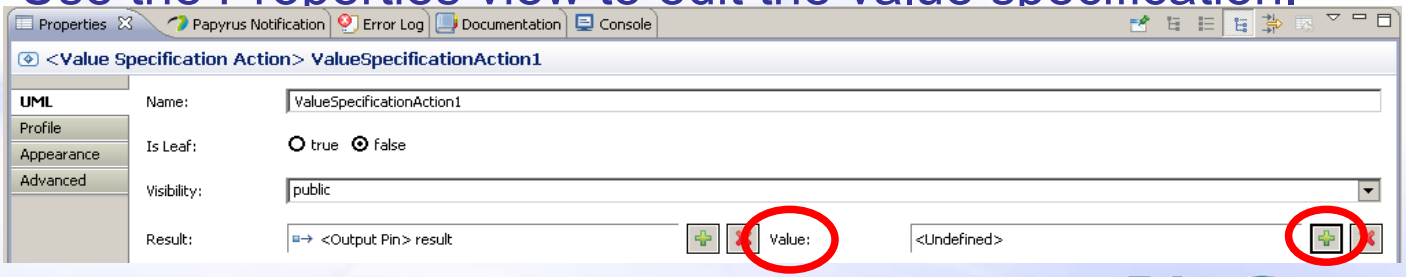
- Read self action : retrieves the host object.
  - This action always has a result output pin to return the hosting object.



- Value specification action : returns the result of evaluating a value specification.
  - This action always has a result output pin to return the evaluation result.

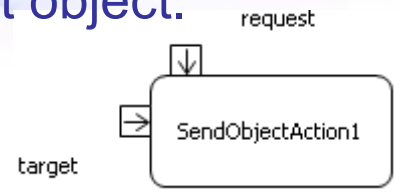


  - Use the Properties view to edit the value specification.

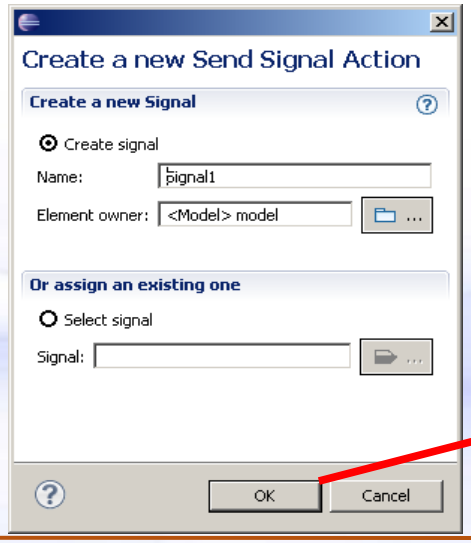


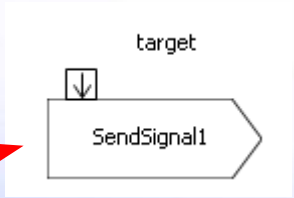
Actions (4/7).

- Send object action : sends a request to a target object.
  - This action always have target and request input pins.



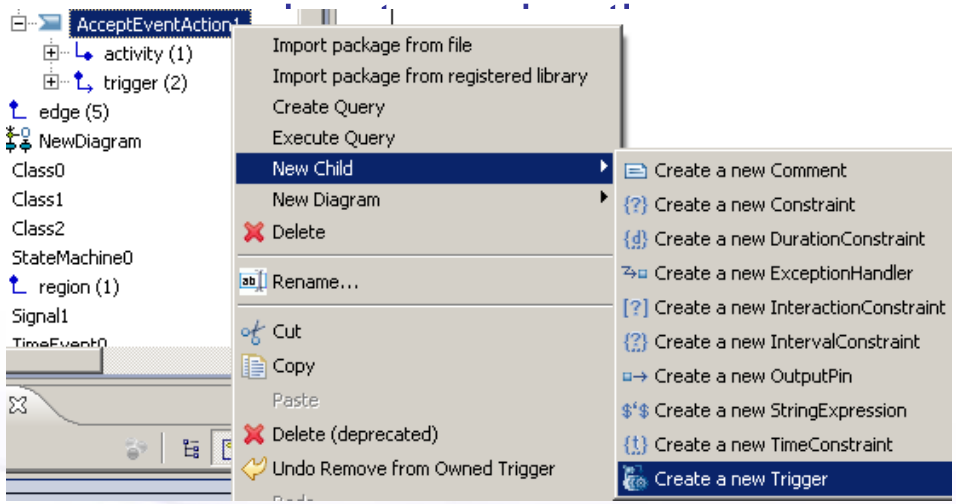
- Send signal action : sends a signal to a target object.
  - This action always has a target input pin.
  - When creating the action, you will be asked to either create a new signal or assign an existing signal to send.



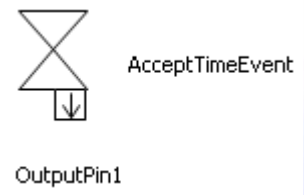


Actions (5/7).

- Accept event Action : waits for a particular event.
  - This action may not have any input pin.
  - Create triggers from the Model Explorer view or use the Properties

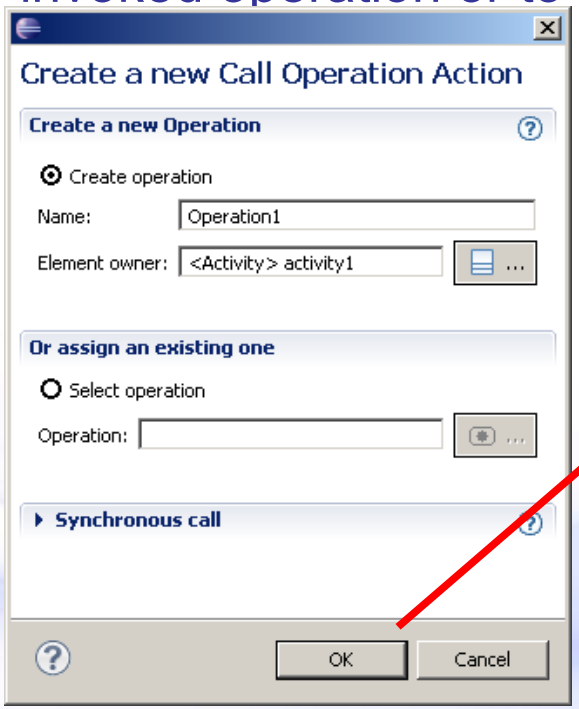


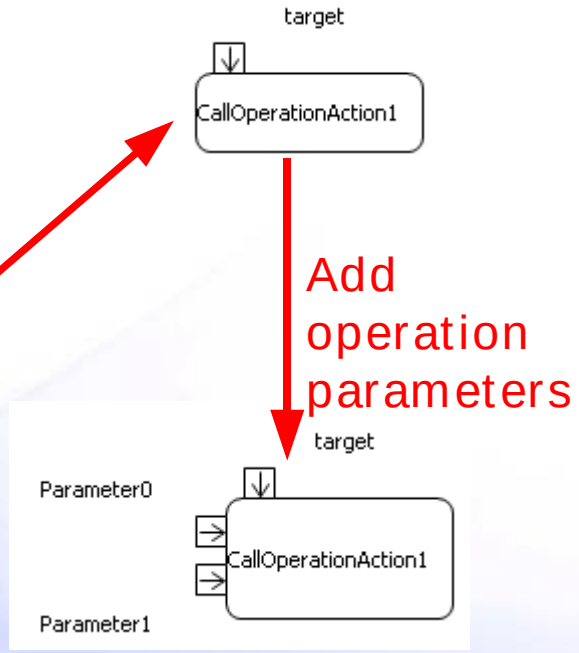
  - The number of output pins must depend on the assigned triggers.
    - (see UML specification)
  - If the triggers all have time events, the figure will change.



Actions (6/7).

- Call operation action : transmits an operation call request to a target.
  - This action always has a target input pins.
  - When creating the action, you will be asked either to create the new invoked operation or to assign an existing one.

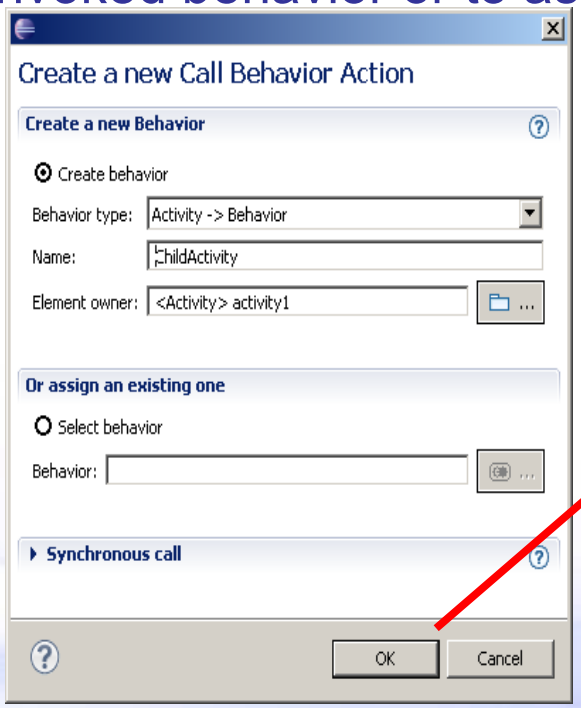


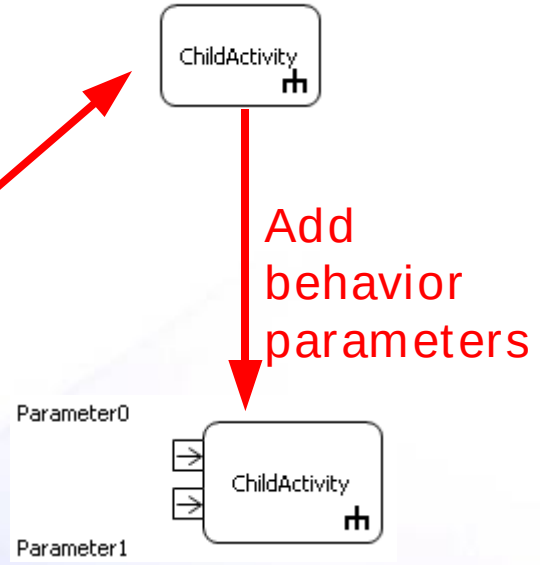


  - The action's pins are synchronized with the operation's parameters.

Actions (7/7).

- Call behavior action : invokes a behavior.
  - When creating the action, you will be asked either to create a new invoked behavior or to assign an existing one.

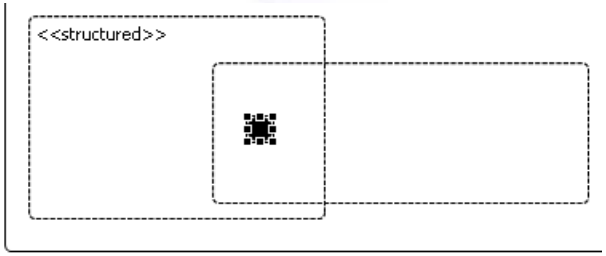




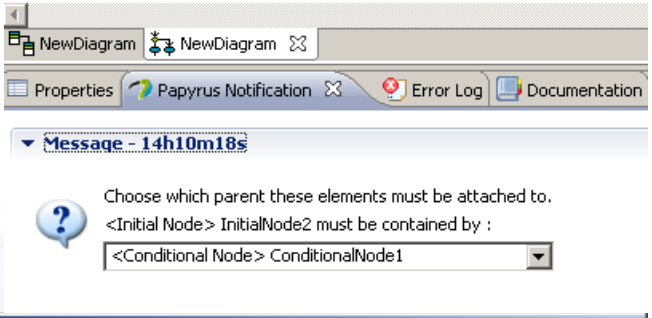
  - The action's pins are synchronized with the behavior's parameters.
  - The rake symbol indicates that the called behavior is an activity.

Activity groups (1/3).

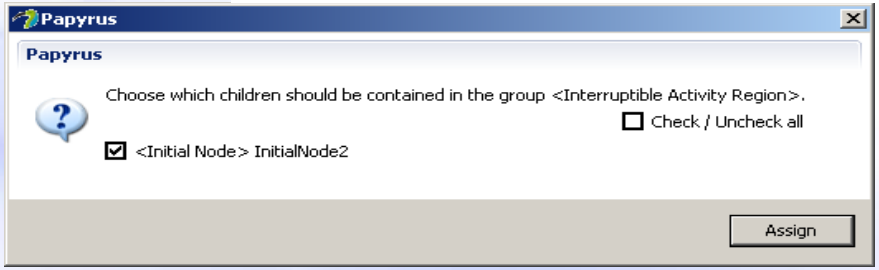
- Activity groups define sets of nodes and edges. Nodes and edges can belong to several groups. Yet, the graphical figure moves with the group which graphically contains it.



- When a node is placed in the intersection of several groups, you will be asked which group the figure shall graphically belong to.



- When a group is drawn over nodes already belonging to a group, you will be asked which nodes shall graphically belong to it.



Activity groups (2/3).

- You may create the following activity groups :
  - Activity partition : identifies nodes with a common characteristic.
    - Activity partitions can share some content with any group.

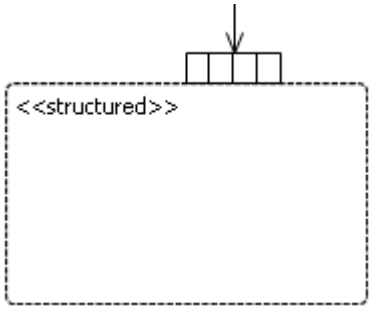
  - Interruptible activity region : terminates as soon as one a token leaves it through one of its interrupting edges.
    - Interruptible activity regions can share some content with any group.

  - Structured activity node : is an action with structured content.
    - Structured activity nodes can not share some content with each other.

Activity groups (3/3).

- Conditional node : a structured activity node representing an exlusive choice among several alternatives.

- Expansion region : a structured activity node that executes multiple times, once for each element of an input collection.
  - The input collection ugh an expansion node.

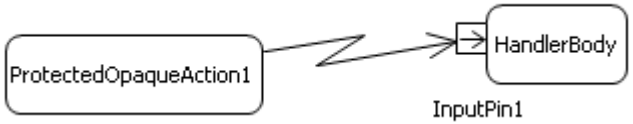


- Loop node : a structured activity node representing a loop with setep, test and body sections.

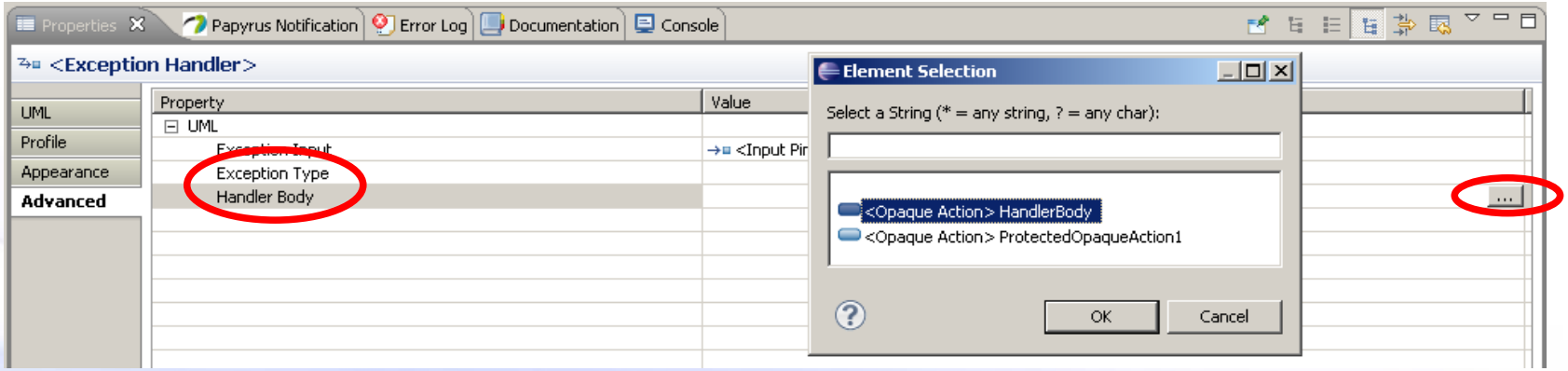
- Sequence node : a structured activity node that executes its actions in

Exception handlers.

- An exception handler is used to catch an exception.
  - Draw it from the palette, from the protected node to an input pin,



  - Then, in the Advanced tab of the Properties view, assign the « Handler Body » and the « Exception Type » properties.



  - The handler body must have only the exception input as input pin.
  - The handler body must have result output pins corresponding to the result output pins of the protected node.

Comments.

- You can create comments with the appropriate tools in the palette.
  - Select the « Comment » tool and click on the diagram to create a comment node.
  - Select the « Link » too and attach it to the commented element.
  - Fill the text in the comment node.