Diagram UML Use Case Elements 1.0 Requirements Specification

1. Scope

1.1 Overview

The Diagram UML Use Case Elements component provides the graphical diagram elements and edges representing the model elements specific to a use case diagram.

1.2 Logic Requirements

1.2.1 Subsystem Node

This class is a concrete NodeContainer. It takes its information from the Subsystem class from the UML Model and from the GraphNode associated with it.

The structure of the GraphNode that corresponds to a Subsystem is given in section 3.2.1.

The stereotype and package compartments could be hidden. The 'isVisible' attribute of the compartment graph node is for this property. There should be methods to tell the node that the compartments' visibility has changed. The node will be resized and an event will be triggered, to signal the size change (the reason is also passed, as a string). The node should also have a method that computes the preferred size, according to the new visible/hidden compartment.

The stereotype and package compartments do not react to any events. They let the events pass to the NameCompartment, which reacts as described below.

The whole component will react to a mouse double-click event, by showing the edit control of the DiagramViewer in order to edit the name of the subsystem. The text of the Name compartment will not be shown while the edit control is up (though the node will not be resized). The event should provide to the DiagramViewer the position where to show the edit control, so it fits on top of the Name compartment, and the initial text. It will also register a listener to receive the event that the text was entered or cancelled in the edit control. It will remove the listener after receiving the event.

The new name will not be set. An event is generated instead, with the old and new name, and with the subsystem and graph node for which the name is set. The node will also provide a method to check the size that will be required for the node if the name would be set (this is required, as other representations of the same Subsystem could be resized as a side effect). The application will register for the event and, eventually it will set the new name. The component should make it easy for the application to set the new name, by performing the resize needed when the name is changed. The method for setting the name will perform the resize of the node, and it will generate a resize event (the reason is also passed, as a string).

The graphical component will be implemented as a drag and drop DropTarget. It should provide a pluggable handler for the Transferable element that will transform it into the actual graphical element. The element will be added after that to the node's body compartment. The component should make it easy to the application to add the new graphical element to the BodyCompartment (note that the new graphical element is not added in the JComponent container hierarchy; it is a parallel element of the Sbsystem added to the diagram view, but drawn in front of the node). There should also be a way for the application to query the size of the BodyCompartment and whether the new graphical element would fit at a certain location, in order to be able to perform the resize. The location of the mouse event is translated so it fits on top of the BodyCompartment.

The node should define a minimum size. It should also have a method that computes the preferred size, according to the name compartment and the existing contained graph nodes. It should also provide a method to compute the preferred size if a new element would be added (same thing as for the new name situation) or if the size of a certain contained element is changed.

In case the diagram viewer's flag for adding new elements from the toolbar is on, the node should react to mouse events differently. When the mouse is pressed, it should consume the event:

- in case the new element is a node:
    - mouse clicked - by accepting the new element. It should translate the point to an appropriate position first, in case the event has not occurred on the body compartment. It should generate an event, with the position where the element should be added.
- in case the new element is an edge:
    - mouse pressed - by accepting the edge end, if the edge is not an Include or Extend. Otherwise, it will consume the event by and calling a pluggable handler that will bring the state of the diagram viewer and element toolbar to a neutral state (setting the flag of the diagram viewer to false...).
    - mouse released - by accepting the edge end, if the edge is not an Include or Extend, or a Generalization with the other end not a Subsystem.

Therefore, for all the situations above, there should be an event triggered, with the proper information. Other handlers will perform the actual operations.

The connector for the edges will behave like the connector for the rectangle shaped elements.

There should be a way to set a popup that will be shown if a popup trigger mouse event action occurs. The popup will be general for the whole component.

The node should have several properties configurable through the graph node:

- the stroke color (defaults to black)
- the fill color (defaults to white)
- the font color (defaults to black)
- the font family (defaults to Arial)
- the font size (defaults to 10).

The node should receive events only in the drawn shape.

1.2.2 Actor Node

This class is a concrete Node. It takes its information from the Actor from the UML Model and from the GraphNode associated with it.

The structure of the GraphNode that corresponds to it is given in section 3.2.1.

The stereotype compartment could be hidden. The 'isVisible' attribute of the compartment graph node is for this property. There should be methods to tell the node that the compartment's visibility has changed. The initial node will be resized and an event will be triggered, to signal the size change (the reason is also passed, as a string). The node should also have a method that computes the preferred size, according to the new visible/hidden compartment.

The name compartment will not show "anonymous" if there is no name, or it could be hidden.

The stereotype compartment do not react to any events. They let the events pass to the component, which reacts as described below.

The whole component will react to a mouse double-click event, by showing the edit control of the DiagramViewer in order to edit the name of the node. The text of the Name compartment will not be shown while the edit control is up (though the node will not be resized). The event should provide to the DiagramViewer the position where to show the edit control, so it fits on top of the Name compartment, and the initial text. It will also register a listener to receive the event that the text was entered or cancelled in the edit control. It will remove the listener after receiving the event.

The new name will not be set. An event is generated instead, with the old and new name, and with the node and graph node for which the name is set. The node will also provide a method to check the size that will be required for the node if the name would be set (this is required, as other representations of the same model element could be resized as a side effect). The application will register for the event and, eventually it will set the new name. The component should make it easy for the application to set the new name, by performing the resize needed when the name is changed. The method for setting the name will perform the resize of the name, and it will generate a resize event (the reason is also passed, as a string).

The graphical component will not be implemented as a drag and drop DropTarget, as it is not a container. However, it should not interfere with the drag and drop action initiated by the user. The user should be able to drop the element on top of the node and the event should be handled by the diagram behind it (as the intention of the user is to add the element to the diagram).

The node should define a minimum size. It should also have a method that computes the preferred size, according to the name compartment and the stereotype compartment. It should also provide a method to compute the preferred size if the name or the stereotypes change.

The node will show all the selection corners, when it is selected. The corners will be shown around the ‘actor’ shape.

The ‘actor’ shape is the base of the component. It should remain at the same place if the stereotype compartment is shown or hidden.

In case the diagram viewer’s flag for adding new elements from the toolbar is on, the node should react to mouse events differently, by letting the events pass to the element behind.

There will be a connector for the edges defined for the node. It will behave differently than the connector for the rectangle shaped elements. The last segment of the edge should point to the closest point on the round shape. The name compartment, the namespace compartment and the stereotype compartment should be taken into consideration, when they are active.

There should be a way to set a popup that will be shown if a popup trigger mouse event action occurs. There will be a popup general for the component.

The node should have several properties configurable through graph node:

- the stroke color (defaults to black)
- the fill color (defaults to black)
- the font color (defaults to black)
- the font family (defaults to Arial)
- the font size (defaults to 10).

The name could be shown in italics if the instance is abstract.

The component should receive events only in the drawn shape and in the text compartments.

1.2.3 UseCase Node

This class is a concrete Node. It takes its information from the UseCase from the UML Model and from the GraphNode associated with it.

The structure of the GraphNode that corresponds to it is given in section 3.2.1.

The node is similar to the one above, with a few differences.

The stereotype compartment and the name compartment are contained in the shape.

The connector for the edges is not the default connector for the rectangle shaped elements. The last segment of the edge should point to the closest point on the eliptic shape.

1.2.4 Extend Edge

This class is a concrete Edge. It takes its information from the Extend class from the UML Model and from the GraphEdge associated with it.

The structure of the GraphEdge that corresponds to an Extend is given in section 3.2.1.

The stereotype compartment could be hidden. The ‘isVisible’ attribute of the compartment graph node is for this property. There should be methods to tell the edge that the compartment’s visibility has changed.

The edge has a dashed line. The stereotype compartment contains the “extends” stereotype (as KeywordMetaclass). It cannot be removed, it is shown first and is separated from the other stereotypes by a blanc space.

There should be two Extend end types defined:

- extension end: none
    - o ----
- base end: a simple arrow, with the color of the edge
    - o ---->

The edge supports the name and the stereotypes as text fields attached to the edge.

Selecting the text fields of the edge will result in selecting the edge.

Double clicking on a text fields will bring the diagram’s viewer editing text field in front, with the text that should be edited. The name is used for this.

There should be a way to set popups that will be shown if a popup trigger mouse event action occurs. There will be a popup general for the component.

1.2.5 Include Edge

This class is a concrete Edge. It takes its information from the Include class from the UML Model and from the GraphEdge associated with it.

The structure of the GraphEdge that corresponds to an Include is given in section 3.2.1.

The edge is similar to the one above, with a few differences.

The stereotype compartment’s default stereotype (the KeywordMetaclass) is “include”.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool to perform model related actions.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

The design must follow the interface found in the class diagram with the component interfaces. The designer is encouraged to add to the existing interface, but not to remove anything.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5

2.1.4 Package Structure

com.topcoder.gui.diagramviewer.uml.usecaseelements

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

The structure of the Diagram Interchange elements should be respected. The names of the compartments are provided in this file.

UML Tool - UseCase Diagram Elements Compartments.rtf

3.2.2 TopCoder Software Component Dependencies:

- Diagram Viewer 1.0
- Diagram Elements 1.0
- Diagram Edges 1.0
- UML Model Manager 1.0
- UML Model components
- Diagram Interchange 1.0
- Configuration Manager 2.1.5 - recommended

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7

- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.