Diagram UML Auxiliary Elements 1.0 Requirements Specification

1. Scope

1.1 Overview

The Diagram UML Auxiliary Elements component provides the comment, free text and polyline graphical diagram elements.

1.2 Logic Requirements

1.2.1 Comment Node

This class is a concrete Node. It takes its information from the Comment 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.

There is no stereotype compartment.

The whole component will react to a mouse double-click event, by showing the edit control of the DiagramViewer in order to edit the body of the node. The body of the comment 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 actual 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 text will not be set. An event is generated instead, with the old and new text, and with the node and graph node for which the text is set. The node will also provide a method to check the size that will be required for the node if the text 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 text. The component should make it easy for the application to set the new text, by performing the resize needed when the text is changed. The method for setting the text will perform the resize of the text, 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 text compartment. It should also provide a method to compute the preferred size if the text.

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

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 shape, taking into consideration the folded corner.

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

The component should receive events only in the drawn shape.

1.2.1.1 CommentLink Edge

This class is a concrete Edge. The relation is represented only by the fact that the Comment is in the list of the ModelElement's comments and that the ModelElement is in Comment's list of annotedElements.

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

There is no stereotype compartment.

The edge has a dashed line.

There edge ends are missing (the edge end is "none"):

- ----

The edge doesn't support any text fields attached to the edge.

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

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

1.2.1.2 FreeText Node

This class is a concrete Node. It takes its information from the TextElement from the DiagramInterchange and from the GraphNode that contains it.

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

There is no stereotype compartment (as it is not a model element).

The whole component will react to a mouse double-click event, by showing the edit control of the DiagramViewer in order to edit the body of the node. The body of the text field 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 actual 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 text will not be set. An event is generated instead, with the old and new text, and with the node and graph node for which the text is set. The node will also provide a method to check the size that will be required for the node if the text 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 text. The component should make it easy for the application to set the new text, by performing the resize needed when the text is changed. The method for setting the text will perform the resize of the text, 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 text compartment. It should also provide a method to compute the preferred size if the text.

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

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 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. There will be a popup general for the component.

The component will not show a border by default and there will be no background.

The node should have several properties configurable through graph node:

- the stroke color (defaults to none)
- the fill color (defaults to none)
- the font color (defaults to black)
- the font family (defaults to Arial)
- the font size (defaults to 10).

The component should receive events only in the drawn shape.

1.2.1.3 Polyline Node

This class is a concrete Node. It takes its information from the Polyline from the DiagramInterchange.

There is no special structure for the Polyline. It is represented as is.

The node is similar to actions are similar to the one above.

The component will not show a border by default (stroke color will be black) and there will be a background (fill color will be black).

The component does not react to double-click, as there is no text to be entered for it.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool to display the auxiliary model and diagram related elements on a diagram in the diagram viewer.

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.auxiliaryelements

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 - Auxiliary 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.