Diagram UML Class Edges 1.0 Requirements Specification

1. Scope

1.1 Overview

The Diagram UML Class Edges component provides the graphical diagram edges representing the model relations specific to a class diagram.

The edges provided are the association, the generalization, the abstraction and the dependency. The edges are very similar, only the way the edge ends looks and the way the line is drawn might be different. The edges extend from the base edge from the Diagram Edges component, providing the visual aspect and concrete graphical edge ends.

1.2 Logic Requirements

1.2.1 AssociationEdge

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

The structure of the GraphEdge that corresponds to an Association is given in the "UML Tool – Class Diagram Elements Compartments.rtf" file posted on the forums. The file shows the structure of GraphNodes (from Diagram Interchange component) that makes up a GraphNode of an Association.

1.2.1.1 The stereotype compartment

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 association that the compartment's visibility has changed.

1.2.1.2 The association has a continuous line.

1.2.1.3 There should be several association end types defined:

- none: there is no association end drawn (if the association end is not 'navigable', of if both association ends are navigable, in which case the arrow is not shown)
  - o ----
- simple arrow: for the directed association (if the association end is navigable and the other end is not)
  - o --->
- aggregation: an empty (white) diamond, with a border (with the color of the association line), for directional aggregation (if the association end's aggregation is 'aggregation' and it is not navigable)
  - o ---< >
- composition: a filled diamond (with the color of the association line), for directional composition (if the association end's aggregation is 'composition' and it is not navigable)
  - o ---<*>
- aggregation, bi-directional: an empty (white) diamond, with a border (with the color of the association line), and a simple arrow, for bi-directional association, with aggregation on one end (if the association end's aggregation is 'aggregation' and it is navigable)
  - o --->< >

- composition, bi-directional: a filled diamond (with the color of the association line), and a simple arrow, for bi-directional association, with composition on one end (if the association end's aggregation is 'composition' and it is navigable)
    - o ---><*>

The association end should be set properly at both ends according to the attributes of the association end.

1.2.1.4 Edge text fields functionality:

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

b) It also supports the name of the association end and the multiplicity attached to each of the association ends.

c) If there is nothing set, or if the multiplicity is "1" (though this should be configurable), the text fields would not be shown, only when the edge is selected.

d) Selecting the text fields of the edge will result in selecting the association. Selecting the text fields of the edge ends results in selecting the association ends.

e) 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 stereotypes compartment doesn't react to this event.

1.2.1.5 Popups

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, and a popup for the edge ends (with the corresponding edge end texts).

1.2.2 GeneralizationEdge

This class is a concrete Edge, similar to the Association. It takes its information from the Generalization class from the UML Model and from the GraphEdge associated with it.

The structure of the GraphEdge that corresponds to a Generalization is given in the "UML Tool – Class Diagram Elements Compartments.rtf" file posted on the forums. The file shows the structure of GraphNodes (from Diagram Interchange component) that makes up a GraphNode of a Generalization.

1.2.2.1 The Generalization uses a continuous line.

1.2.2.2 There is no notion of 'active' edge ends or texts attached to the edge ends. There is only the edge with its name and stereotype compartment. Of course, the edge end will be drawn appropriately, but is will not react to mouse events; instead, it will let the events pass to the edge.

1.2.2.3 There should be two Generalization end types defined:

- parent end: an empty (white) triangle, with a border (with the color of the association line)
    - o ----|>
- child end: none
    - o ----

1.2.2.4 There should be a way to set a popup general for the component.

1.2.3 AbstractionEdge

This class is a concrete Edge, similar to the Generalization. It takes its information from the Abstraction class from the UML Model and from the GraphEdge associated with it.

The structure of the GraphEdge that corresponds to a Abstraction is given in the "UML Tool – Class Diagram Elements Compartments.rtf" file posted on the forums. The file shows the structure of GraphNodes (from Diagram Interchange component) that makes up a GraphNode of a Abstraction.

1.2.3.1 The difference is that Generalization uses a dashed line.

1.2.4 DependencyEdge

This class is a concrete Edge, similar to the Abstraction. It takes its information from the Dependency class from the UML Model and from the GraphEdge associated with it.

The structure of the GraphEdge that corresponds to a Dependency is given in the "UML Tool – Class Diagram Elements Compartments.rtf" file posted on the forums. The file shows the structure of GraphNodes (from Diagram Interchange component) that makes up a GraphNode of a Dependency.

1.2.4.1 The difference is that Dependency has different end types defined:

- supplier end: none
  - ----
- client end: simple arrow
  - ---->

1.2.4.2 Uses a dashed line.

1.2.5 Showing / hiding compartments

The elements above will be able to show or hide the compartments or elements according to the DiagramElement.isVisible attribute.

There should be a way to set through the API the visible flag of the contained compartments and elements.

There should be a way for the edges to configure for what values of the edge texts these texts will not be shown and to configure which texts will be shown when the edge is selected.

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 UML class edges in the diagrams.

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.classelements

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 - 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.