Diagram Viewer 1.0 Requirements Specification

1. Scope

1.1 Overview

The Diagram Viewer component provides a SWING tabbed panel that will display the diagrams. The component will also provide the general diagram panel (that can be zoomed and can have a background grid) and the input text control used in GUI applications to enter text for different elements present in the diagram.

1.2 Logic Requirements

1.2.1 DiagramViewer

The component will provide this concrete JComponent. It will look like the sample provided.

It must be able to display multiple diagrams inside tabs. The number of open tabs should be configurable. There should be an option to display the text for each tab in full version or shortened version, in which case a tool tip should be available with the whole name.

It should have an X button in the upper right corner to close the active diagram. It should allow no diagram to be opened.

It should have the zoom spin button in the lower right corner. This should be configurable (increment, allowed values, minimum and maximum values).

1.2.2 DiagramView

The component will provide this concrete JComponent. It will correspond to a Diagram from Diagram Interchange component. The diagram will have a null layout manager, so the elements will be added according to the (x,y) coordinates.

It should have an option to set a background grid (by using a tiled picture or by using a configurable drawing algorithm - should choose the best option). The grid should look as shown in the sample.

The grid should be shown or hidden according to a general flag kept in the DiagramViewer. This flag should be changeable through the API.

The diagram will have a preferred size, according to the contained elements plus a configurable margin. However, it should paint on any clip rectangle (meaning that if the size is (200,200), but the diagram viewer sets its size to (500,500), it should paint the entire surface). The grid in the active area should be painted differently (lighter) than the grid outside the active area.

There should be a way to initialize the diagram view component from a Diagram object from Diagram Interchange component. The graphic diagram will represent the Diagram object throughout its existence.

1.2.3 Zoom

The diagrams will be zoomed according to their own zoom factor. The Zoom Panel component should be used behind the scenes by this component.

The zoom should be changed according to the zoom spin button, but a way to set this through the API should also be provided (the value in the spin button should be changed accordingly).

In addition, the zoom should be changeable if a mouse scroll event is received while Ctrl is pressed (the normal usage).

The zoom change (from within this component) should trigger a proper event: the diagram for which the zoom was changed and the new zoom factor.

1.2.4 Scrolling

There should be a way to react to diagram scrolling. Therefore, the component should provide a way for the API to register to these events.

1.2.5 Popup

There should be a way for the application to register a popup for the diagram, which will be shown if a mouse popup trigger event occurs.

1.2.6 Selected elements

The diagram viewer should maintain a list of selected elements inside the diagram. This should be editable through the API. There should be a method also to check if a certain element is selected.

1.2.7 Mouse events

1.2.7.1 Selection rectangle

The component will draw a rectangle using a dashed line if the mouse is dragged on the diagram. This option is for visually selecting elements on the diagram. The component should trigger events related to the rectangle that is drawn, so the application can register to this event and 'select' the elements that are located inside the rectangle.

A default listener should be provided that checks which elements intersect the rectangle and updates the list of 'selected' elements. The elements will be of Node and Edge type from Diagram Elements component.

1.2.7.2 Add new element (drop target)

The diagram 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 diagram.

1.2.7.3 Add new element (add element state for the diagram)

The application should be able to change the state of the diagram (using some flag) so that if an element is selected to be added to the diagram, the mouse events will be processed differently. The graphical element will be obtained from a pluggable source.

There are two ways of adding elements:

- when the mouse is clicked
- after the mouse draws a bounds rectangle for the element.

The option about how to add the element should be configurable also through the API (meaning that when an element should be added, the application should have a way to configure how the element will be added).

1.2.8 Text input tool

The diagram should have a text input tool that will be displayed as a popup. The text field's text will be configurable through the API. It will disappear if Enter or Escape keys are pressed (setting or ignoring the input), or when the mouse is pressed outside the text input (in which case the input is accepted).

This text input will resize automatically along with the text inside. It will support new lines if Ctrl+Enter are pressed. However, the new lines option (whether to accept new lines or not) should be configurable through API.

When the text is set, it will trigger an event, so the application should be able to register for it.

The application will provide the coordinates where it wants it to be displayed according to the zoomed diagram. It should apply the zoom transformation to the coordinates so that the text field is shown on top of the zoomed diagram in the right place.

The text field will not be zoomed.

1.2.9 Creating Diagram Views

The diagram viewer should provide a method that creates a diagram view or an array of diagram views. The application will use these objects to print them or export them to images.

The diagram viewer should have a configurable cache for the diagram views it creates. The creation of diagram views might be an expensive operation, so these objects should be kept in cache, even if they were never actually displayed to the user (they might be required a bit later).

A cache is required, as opposed to a non-expiring map, as a design with many diagrams might require too much memory and, since only a few diagrams are viewed, the others could be removed from memory.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool as 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

None.

2.1.3 Environment Requirements

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

2.1.4 Package Structure

com.topcoder.gui.diagramviewer

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The elements mentioned in the Logic Requirements section

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

- Diagram Interchange 1.0
- Diagram Elements 1.0
- Zoom Panel 1.0

**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.