Event Manager 1.0 Requirements Specification

1. Scope

1.1 Overview

The Event Manager component provides a general framework for handling events triggered from the GUI. The component handles simple GUI events by notifying the listeners, will handle action events by validating the events, executing the actions and notifying the listeners and will handle undo / redo events.

1.2 Logic Requirements

The GUI events, that are needed just to signal changes from one part of the GUI to another, are passed directly to the listeners. For the events that carry an action, the action will be executed using the Action Manager component. The undo / redo events are also handled using ActionManager.

The component defines the interfaces for the action event validators and for the listeners. It also defines the ActionEvent, representing the event object for action events and the events for undo / redo actions.

1.2.1 EventManager

The manager will be responsible for handling action events and GUI events.

For the ActionEvents, it will validate the event using the event validators, it will extract the action from the event, will invoke the proper method of the ActionManager and will notify the listeners

- if the event is a simple ActionEvent, it will invoke ActionManager.executeAction(..) and notify the ActionEventListeners using actionPerformed(..) method;
- if the event is UndoChangesEvent, it will invoke ActionManager.undoActions(..) and will notify the ActionEventListeners using undoActionPerformed(..), passing each of the undoable actions that were undone to the action in the event, in reverse order. Note that only the registered listeners for each undone action are notified;
- if the event is RedoChangesEvent, it will invoke ActionManager.redoActions(..) and will notify the ActionEventListeners using redoActionPerformed(..), passing each of the undoable actions that were undone to the action in the event, in the normal order. Note that only the registered listeners for each redone action are notified.

For the GUI events, it will simply notify the registered listeners.

The manager will allow the user to register validators for concrete actions or for all actions, action listeners for concrete actions or for all actions and to register listeners for concrete GUI events and for all events.

1.2.2 ActionEvent

The action event extends EventObject and has a reference to the action to be executed. Applications should create actions events if the action to be taken is significant (affects the application model) and pass it to the event manager to be handled.

1.2.3 UndoChangesEvent

This class extends from the ActionEvent. The action inside must be an UndoableAction. The ActionManager will undo all the actions down to this one.

1.2.4 RedoChangesEvent

This class extends from the ActionEvent. The action inside must be an UndoableAction. The ActionManager will redo all the actions up to this one.

1.2.5 ActionEventValidator interface

This interface represents the validator invoked by the manager in order to validate an action event. The validators will be mapped to the concrete class of the actions. There should be an option to register a validator for all action events.

The validator's validate(..) method will return EventValidation.SUCCESSFULL if the validation succeeded with no change, will return EventValidation.EVENT_MODIFIED if the validation succeeded, but the action was modified and will return EventValidation.DENIED if the action is not valid.

The manager will execute the action if it wasn't denied. It will notify the ActionEventListeners, regardless of the validation result, but the validation result will be passed, so the listeners can take the proper actions.

1.2.6 ActionEventListener

This interface represents the listener notified by the event manager about action events and about the undo / redo events. It provides a method to be invoked about the execution of an action, using the action event and the validation result. It will also provide two methods to be used when an undo / redo event occurred. These methods will be invoked for each undoable action that is undone / redone.

1.2.7 GUIEventListener

This interface represents the listener notified by the event manager about simple GUI events that must be triggered by one part of the GUI in order to notify other parts of the GUI, so that they stay in sync.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used in the TopCoder UML Tool as the central place for handling action events, undo / redo events and simple GUI events triggered by the GUI.

1.5 Future Component Direction

None at this moment.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

The design must follow the interfaces found in the class diagram with the component interfaces. The designer is encouraged to add to the existing interfaces, 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.util.eventmanager

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The action event validators, the action listeners and the GUI event listeners.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

- Action Manager 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.