

## UML Tool Actions - Class Elements Actions 1.0 Requirements Specification

### 1. Scope

#### 1.1 Overview

The Class Elements Actions component provides the Actions related to the model elements specific to a class diagram. The actions are strategy implementations of the action interfaces in the Action Manager component. The provided actions are for adding / removing / copying / cutting / pasting the elements and relationships. The elements are package, interface, class, exception and enumeration. The relationships are association, aggregation, composition, realization/abstraction and dependency.

#### 1.2 Logic Requirements

##### 1.2.1 *Add / Remove / Copy / Cut / Paste Package action*

This component will provide a concrete action for each operation.

The Add / Remove / Cut / Paste actions are UndoableActions.

The Copy action is a TransientAction.

Note that the Cut action can be implemented as a CompoundUndoableAction, made of a transient Copy action (wrapped in TransientUndoableAction) and an undoable Remove action.

##### 1.2.1.1 The Add action will be configured with:

- the Package
- the parent Namespace (optional)

The action will simply add the Package to the parent namespace's owned elements, or directly to the Model.

The action will also pass the element to the ProjectConfigurationManager, to apply any initial formatting.

##### 1.2.1.2 The Remove action will be configured with:

- the Package

The action will remove the element from the model. However, it is not responsible for removing the owned elements, or the relations connected to it or its owned elements.

##### 1.2.1.3 The Copy action will be configured with:

- the Package
- the Clipboard (defaults to the system clipboard)

The action will copy the element. However, it is not responsible for copying the owned elements, or the relations connected to it or its owned elements.

The copy information will be placed in the clipboard. Note that the Copy and the Paste action must function together.

The DataFlavor of the Transferable object used should be documented.

##### 1.2.1.4 The Cut action will be configured with:

- the Package

This action will Copy and Remove the element, as specified above.

1.2.1.5 The Paste action will be configured with:

- the Transferable content representing the Package
- the parent Namespace (optional)

The action will paste the element into the model in the same namespace it was in, or in the provided namespace. It will get the information from the received Transferable object.

1.2.2 *Add / Remove / Copy / Cut / Paste Interface action*

These actions are similar to the ones above. They use Interface, instead of Package.

1.2.3 *Add / Remove / Copy / Cut / Paste Class action*

These actions are similar to the ones above. They use Class, instead of Package.

1.2.4 *Add / Remove / Copy / Cut / Paste Exception action*

These actions are similar to the ones above. They use Class with an <<exception>> stereotype, instead of Package.

1.2.5 *Add / Remove / Copy / Cut / Paste Enumeration action*

These actions are similar to the ones above. They use Enumeration, instead of Package.

1.2.6 *Add / Remove / Copy / Cut / Paste Association Relationship action*

These actions are similar to the ones above. They use Association, instead of Package.

The aggregation and composition relationships are just sub-cases of the association relationship.

1.2.7 *Add / Remove / Copy / Cut / Paste Generalization Relationship action*

These actions are similar to the ones above. They use Generalization, instead of Package.

1.2.8 *Add / Remove / Copy / Cut / Paste Abstraction Relationship action*

These actions are similar to the ones above. They use Abstraction, instead of Package.

1.2.9 *Add / Remove / Copy / Cut / Paste Dependency Relationship action*

These actions are similar to the ones above. They use Dependency, instead of Package.

### **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.uml.actions.model.class

### 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?*

None

##### 3.2.2 *TopCoder Software Component Dependencies:*

- Action Manager 1.0
- UML Model Manager 1.0
- UML Project Configuration 1.0
- UML Model components
- 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.