UML Tool Actions - Element Properties Actions 1.0 Requirements Specification

1. Scope

1.1 Overview

The Element Properties Actions component provides general and specialized Actions related to the model elements. The actions are strategy implementations of the action interfaces in the Action Manager component. The provided actions are general actions that apply to any element, classifier feature actions and a few custom actions.

1.2 Logic Requirements

1.2.1 General actions

1.2.1.1 Change documentation action

This action changes the documentation of a model element. The documentation is added as a taggedValue to the model element, using the “documentation” TagDefinition. The documentation for the operation arguments is also added like this, not using documentation tags as shown below.

The action is an UndoableAction.

The action will be configured with:

- the model element
- the new documentation

1.2.1.2 Update documentation tags action

This action updates the documentation tags of a model element. The documentation tags are added as taggedValues to the model element, using the “throws” ... TagDefinitions.

The action is an UndoableAction.

The action will be configured with:

- the model element
- the operation type (“AddDocumentationTag”, “RemoveDocumentationTag”, “UpdateDocumentationTag”)
- the documentation tag involved
- the documentation for the tag

1.2.1.3 Change zOrder action

This action changes the order of the elements in the same namespace. It changes the order of the contained graph elements of a namespace.

The action is an UndoableAction.

The action will be configured with:

- the graph element
- the operation type: forward, backward, to front, to back.

1.2.1.4 Change style action

This action is for changing style properties of a diagram element. The properties are added as Property elements to the diagram element.

The action is an UndoableAction.

The action will be configured with:

- the diagram element
- the property name
- the property value

1.2.1.5 Change position action

This action is for changing the position of a graph element. The position is saved as the graph element’s position attribute.

The action is an UndoableAction.

The action will be configured with:

- the graph element
- the new position

1.2.1.6 Change size action

This action is for changing the size of a graph node. The position is saved as the graph node’s size attribute.

The action is an UndoableAction.

The action will be configured with:

- the graph node
- the new size

1.2.1.7 Change entity name action

This action will change the name of a model element - ModelElement.name property.

The action is an UndoableAction.

The action will be configured with:

- the model element
- the new name

1.2.1.8 Set entity stereotypes action

This action updates the list of stereotypes of a model element - ModelElement.stereotypes property.

The action is an UndoableAction.

The action will be configured with:

- the model element
- the operation type (“AddStereotypeTag”, “RemoveStereotypeTag”, “UpdateStereotypeTags”)
- the stereotype(s)

1.2.1.9 Change entity visibility action

This action will change the visibility of a model element - ModelElement.visibility property.

The action is an UndoableAction.

The action will be configured with:

- the model element
- the visibility

1.2.1.10 Change relationship path action

This action will change the path of a graph edge.

The action is an UndoableAction.

The action will be configured with:

- the graph edge
- the waypoints
- the position for each graph connectors

1.2.2 Classifier features actions

1.2.2.1 Add attribute action

This action will add an attribute to a classifier. The attribute is added to the features list of the classifier.

The action is an UndoableAction.

The action will be configured with:

- the attribute
- the classifier
- the position among attributes (optional)

1.2.2.2 Add operation action

This action will add an operation to a classifier. The operation is added to the features list of the classifier.

The action is an UndoableAction.

The action will be configured with:

- the operation
- the classifier
- the position among operations (optional)

1.2.2.3 Update attribute action

This action will update the given attribute. The received parameters should be complete (or a way to mention that they are not altered should be provided). If there is a missing argument, it will be assigned a default value.

The action is an UndoableAction.

The action will be configured with:

- the attribute
- the visibility
- the name
- the return type
- the return value
- the changeability
- the multiplicity
- the ordering
- the modifiers (static, transient)

1.2.2.4 Update attribute order action

This action will update the position of the attribute among the other attributes of the classifier. Note that the features list contains both the attributes and operations.

The action is an UndoableAction.

The action will be configured with:

- the attribute
- the classifier
- the position among attributes

1.2.2.5 Update operation action

This action will update the given operation. The received parameters should be complete (or a way to mention that they are not altered should be provided). If there is a missing argument, it will be assigned a default value.

The action is an UndoableAction.

The action will be configured with:

- the operation
- the visibility
- the name
- the parameters
    - o the parameter names
    - o the parameter types
- the return type
- the concurrency
- the modifiers (abstract, static, final)

1.2.2.6 Update operation order action

This action will update the position of the operation among the other operations of the classifier. Note that the features list contains both the attributes and operations.

The action is an UndoableAction.

The action will be configured with:

- the operation
- the classifier
- the position among operations

1.2.3 Custom model actions

1.2.3.1 Mark classifier abstract action

This action will mark a class abstract - the GeneralizableElement.isAbstract property.

The action is an UndoableAction.

The action will be configured with:

- the classifier
- the new abstract modifier

1.2.3.2 Mark class final action

This action will mark a class static - the GeneralizableElement.isLeaf property.

The action is an UndoableAction.

The action will be configured with:

- the classifier
- the new final modifier

1.2.3.3 Mark nested classifier static action

This action will mark a classifier static. This is accomplished using a TagDefinition(“static”) with a taggedvalue in the Classifier with value “true”.

The action is an UndoableAction.

The action will be configured with:

- the classifier
- the new static modifier

1.2.3.4 Update association end action

This action will update the attributes of an association end.

The action is an UndoableAction.

The action will be configured with:

- the association end
- the visibility
- the name
- the type
- the return value
- the changeability
- the multiplicity

- the ordering
- the modifiers (static, transient, navigable)
- the aggregation

1.2.3.5 Set transition guard action

This action will set / remove / change the guard for a transition from an activity graph.

The action is an UndoableAction.

The action will be configured with:

- the transition
- the guard
  - the guard name
  - the guard expression

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.general

com.topcoder.uml.actions.model.class.feature

com.topcoder.uml.actions.model.custom

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.