UML Tool Actions – Project Actions 1.0 Component Specification

1. Design

The Project Actions component provides the Actions related to the project. The actions are strategy implementations of the action interfaces in the Action Manager component. The provided actions are for loading a TCUML / XMI file, saving to a TCUML / XMI file, for creating a new project, generating source code, printing diagrams and exporting diagrams to image files.

This design is quite straightforward, and it follows the requirement specification exactly to implement all required project actions as mentioned above.

1.1 Design Patterns

Strategy Pattern – All concrete action classes in this design are strategy implementations of the action interfaces in the Action Manager component.

1.2 Industry Standards

UML 1.5

1.3 Required Algorithms

None.

1.4 Component Class Overview

1.4.1 Package com.topcoder.uml.actions.project

- **CreateNewProjectAction**: CreateNewProjectAction class implements Action interface and it will create a new project. It will empty the Model, the list of ActivityGraphs and the list of Diagrams from the UMLModelManager. And it will also set the project language.
- **LoadXMIFileAction**: LoadXMIFileAction class implements Action interface and it will create a new project and will load the XMI file. It will use XMI Reader component to parse the contents into the UML Model. After that, it will get the Model from the UMLModelManager and check if it has a “ProjectLanguage”TagDefinition. It will get the tag value and set it in the UMLModelManager. The default project language from ProjectConfigurationManager is used, if this value is missing. A “ProjectLanguage” TagDefinition is created with the default language, in case it is missing, and adds it to the Model.
- **LoadTCUMLFileAction**: LoadTCUMLFileAction class implements Action interface and it will create a new project and will load the TCUML file, by unzipping it and loading the XMI file inside.
- **SaveTCUMLFileAction**: SaveTCUMLFileAction class implements TransientAction interface and it will save the project to a TCUML file, which is a zip file containing the actual XMI file inside.
- **ExportToXMIFileAction**: ExportToXMIFileAction class implements TransientAction interface and it will save the project to an XMI file. It will use XMI Writer component to transform the UML Model to XMI.
- **GenerateCodeAction**: GenerateCodeAction class implements TransientAction interface and it will generate code for the given packages or classifiers. It will use the Stub Class Generator component to perform the actual logic.
- **PrintDiagramAction**: PrintDiagramAction class implements TransientAction interface and it will print diagrams. The diagram should be received as a

java.awt.Component, not as a Diagram. The diagram Component should be the graphical representation of the diagram. It will be passed to the Print Manager component.

- **ExportDiagramToImageAction**: ExportDiagramToImageAction class implements TransientAction interface and it will export a diagram to an image file. The diagram should be received as a java.awt.Component, not as a Diagram. The diagram Component should be the graphical representation of the diagram. It will be passed to the Image Exporter component.

1.5 Component Exception Definitions

1.5.1 Custom Exceptions

No custom exceptions are necessary for this component, all action classes in this design will throw the ActionExecutionException exception from the Action Manager component in their execute methods.

1.5.2 System Exceptions

- **IllegalArgumentException**: Used wherever empty String argument is used while not acceptable. Normally an empty String is checked with trimmed result. It also thrown when null argument is passed in or some other cases when the argument is invalid.

1.6 Thread Safety

This component is not thread-safe, and there is no need to make it thread-safe, as each thread is expected to create its own action object to perform the task rather than share the same action object in different threads. And there is no such requirement too.

Though all action classes in this design are immutable, but the CreateNewProjectAction, LoadXMIFileAction and LoadTCUMLFileAction classes will change the internal state of the UMLModelManager object passed in, so they are not thread-safe; the SaveTCUMLFileAction and ExportToXMIFileAction will only read contents from UMLModelManager object, which should not be changed during this process to ensure the thread-safety, and both actions shouldn't write to the same file at the same time; the GenerateCodeAction's thread-safety depends on the Stub Class Generator component; the PrintDiagramAction's thread-safety depends on the Print Manager component; the ExportDiagramToImageAction's thread-safety depends on the Image Exporter component.

2. Environment Requirements

2.1 Environment

Java 1.5

2.2 TopCoder Software Components

- **Action Manager 1.0**: The Action and TransientAction interfaces implemented by this design are from this component.
- **UML Model Manager 1.0**: The UMLModelManager class used in this design is from this component.
- **UML Project Configuration 1.0**: The ProjectConfigurationManager class used in this design is from this component.
- **UML Model components**: The model data classes are used by all handlers directly and indirectly.
- **XMI Reader 1.0**: It is used to load the XMI file in this design.

- **XMI Writer 1.0**: It is used to save the XMI file in this design.
- **Stub Code Generator 1.0**: It is used to generate the code in this design.
- **Print Manager 1.0**: It is used to print the diagrams in this design.
- **Image Exporter 1.0**: It is used to export diagrams to image files in this design

NOTE: Configuration Manager is not used as all the configurable properties are passed into actions' constructors directly as arguments, which are more convenient as some arguments cannot be easily created from the configured properties (e.g. the UMLModelManager object, which is supposed to be shared in multiple actions), and it is also more flexible to set these arguments directly to constructor than load them from Configuration Manager.

2.3 Third Party Components

None.

3. Installation and Configuration

3.1 Package Name

com.topcoder.uml.actions.project

3.2 Configuration Parameters

None.

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

None.

4.3 Demo

```
Assume the modelManager (UMLModelManager object), configManager
(ProjectConfigurationManager object), codeGenerator (CodeGenerator
object), printManager (PrintManager object), and imageExporter
(ImageExporter object) are provided by application.
```

4.3.1 Execute CreateNewProjectAction

```
// create a new action
CreateNewProjectAction action1 =
      new CreateNewProjectAction("java", modelManager);

// a new empty project is created
action1.execute();
```

4.3.2 Execute LoadXMIFileAction

```
// create a new action
LoadXMIFileAction action2 = new LoadXMIFileAction(
      new File(“action.xmi”), modelManager, configManager);

// load the conent of xmi file into modelManager
action2.execute();
```

4.3.3 *Execute LoadTCUMLFileAction*

```
// create a new action
LoadTCUMLFileAction action3 = new LoadTCUMLFileAction(
      new File(“action.zuml”), modelManager, configManager);

// execute the action to load the xmi data from action.zuml
action3.execute();
```

4.3.4 *Execute SaveTCUMLFileAction*

```
// create a new action
SaveTCUMLFileAction action4 = new SaveTCUMLFileAction(
      new File(“action.zuml”), true, modelManager);

// save the model into the action.zuml file with diagram data
action4.execute();

// create a new action to save model without the diagram data
action4 = new SaveTCUMLFileAction(
      new File(“action.zuml”), false, modelManager);
action4.execute();
```

4.3.5 *Execute ExportToXMIFileAction*

```
// create a new action
ExportToXMIFileAction action5 = new ExportToXMIFileAction(
      new File(“action.xmi”), true, modelManager);

// save the model into the action.xmi file with diagram data
action5.execute();

// create a new action to save model without the diagram data
action5= new ExportToXMIFileAction(
      new File(“action.xmi”), false, modelManager);
action5.execute();
```

4.3.6 *Execute GenerateCodeAction*

```
// create Classifier objects for test
Classifier classifier1 = new ClassImpl();
classifier1.setName(“Test”);
Classifier classifier2 = new ClassImpl();
classifier2.setName(“Boo”);

List<Classifier> classifiers = new ArrayList<Classifier>();
classifiers.add(classifier1);
classifiers.add(classifier2);

// create Package objects for test
Package package1 = new PackageImpl();
package1.setName(“com.foo”);
Package package2 = new PackageImpl();
```

```
package2.setName("com.bar");

List<Package> packages = new ArrayList<Package>();
packages.add(package1);
packages.add(package2);

// the location to store generated code
String location = "project/test";

// java file is only generated for "Test" class
GenerateCodeAction action6 = new GenerateCodeAction("java",
       location, classifier1, codeGenerator);
action6.execute();

// java files are generated for all entities in "com.foo" package
action6 = new GenerateCodeAction("java",
       location, package1, codeGenerator);
action6.execute();

// java files are generated for both "Test" and "Boo" classes
action6 = new GenerateCodeAction("java",
       location, classifiers, codeGenerator);
action6.execute();

// java files are generated for entities in both "com.foo" and "com.bar"
// packages
action6 = new GenerateCodeAction("java",
       location, packages, codeGenerator);
action6.execute();
```

4.3.7 *Execute PrintDiagramAction*

```
// assume both comp1 & comp2 objects below are provided
// by the application
Component comp1 = …
Component comp2 = …

List<Component> comps = new ArrayList<Component>();
comps.add(comp1);
comps.add(comp2);

// assume both rect1 & rect2 objects below are provided
// by the application
Rectangle rect1 = …
Rectangle rect2 = …

List<Rectangle> rects = new ArrayList<Rectangle>();
rects.add(rect1);
rects.add(rect2);

// print a single diagram without viewport
PrintDiagramAction action7 = new PrintDiagramAction(comp1, printManager);
action7.execute();
```

```
// print a single diagram with viewport
action7 = new PrintDiagramAction(comp1, rect1, printManager);
action7.execute();

// print a list of diagrams without viewports
action7 = new PrintDiagramAction(comps, printManager);
action7.execute();

// print a list of diagrams with viewports
action7 = new PrintDiagramAction(comps, rects, printManager);
action7.execute();
```

4.3.8 *Execute ExportDiagramToImageAction*

```
// assume both comp1 & comp2 objects below are provided
// by the application
Component comp1 = …
Component comp2 = …

List<Component> comps = new ArrayList<Component>();
comps.add(comp1);
comps.add(comp2);

// assume both rect1 & rect2 objects below are provided
// by the application
Rectangle rect1 = …
Rectangle rect2 = …

List<Rectangle> rects = new ArrayList<Rectangle>();
rects.add(rect1);
rects.add(rect2);

// the location to store the generated images
String location = “project/test”;

// file names for the generated image files
String fileName1 = “class-diagram1.gif”;
String fileName2 = “class-diagram2.gif”;
List<String> fileNames = new ArrayList<String>();
fileNames.add(fileName1);
fileNames.add(fileName2);

// export a single diagram without viewport
ExportDiagramToImageAction action8 = new ExportDiagramToImageAction(
       comp1, location, fileName1, “gif”, imageExporter);
action8.execute();

// export a single diagram with viewport
action8 = new ExportDiagramToImageAction(
       comp1, rect1, location, fileName1, “gif”, imageExporter);
action8.execute();
```

```
// export multiple diagrams without viewport
action8 = new ExportDiagramToImageAction(
      comps, location, fileNames, “gif”, imageExporter);
action8.execute();

// export multiple diagrams with viewport
action8 = new ExportDiagramToImageAction(
      comps, rects, location, fileNames, “gif”, imageExporter);
action8.execute();
```

5 Future Enhancements

None.