ZUML_2 TCUML_Converter_Seqeunce Diagram 1.0 Component Specification

1. Design

The ZUML 2 TCUML Converter - Sequence Diagrams component provides means to help convert the ZUML format from Poseidon to the TCUML format in TC UML Tool. This component provides the Sequence Diagram conversion tasks on a ZUML file.
This component will be used in the TC UML Tool to load a ZUML file and transform it into its internal model. The action where this component will be used will be a modified Open file action, which will apply different transformations to the model while reading it, or after the reading process.

- Design Overview

1. This design provides the model beans for the UML2 elements for collaboration in ZUML file, then the XMI2ModelHandler can be used to read the UML2 collaboration elements if you configure the XMI Reader Model Plug-in 1.0 component, then the application can use XMISequenceDiagramConverterUtil utility class to convert the UML2 model to TC UML model and convert the Sequence Diagram instance to TC UML Sequence Diagram instance.

2. When converting the UML2 collaboration classes, we should keep the converted instance in the old model because the UML model doesn't have a id variable but in the XMI file, model instance can be referenced by id.
   For example:
   The UML2:Interaction element (Model: Interaction) should be converted into UML:Collaboration(Model: Collaboration), and the owner of the Sequence Diagram is UML2:Interaction, so the owner should be also set to converted UML:Collaboration instance. In current design, the converted UML:Collaboration instance will be stored in the original Interaction instance, so that we can replace the owner of Sequence Diagram easily.

3. The ownedBehavior (<UML2:BehavioredClassifier.ownedBehavior>) is not a property in Collaboration, UseCase or Class model beans. So we need to rename ownedBehavior to ownedElement before the original XMI handler processing the element. We use RenameConverter class to do the renaming task; the RenameConverter is defined in ZUML 2 TCUML Converter 1.0 component. This is discussed in
   http://forums.topcoder.com/?module=Thread&threadID=591392&start=0&mc=11
   See 3.3.2 for how to configure ZUML 2 TCUML Converter 1.0 component.

- **Enhancements**
    1. The requirements only require converting the model of sequence diagram. (RS 1.2.1.4 and 1.2.2). But the sequence diagram (Diagram instance) is also different between ZUML and TCUML. So in this design, the functionality to convert the sequence diagram (Diagram instance) is also provided.

1.1 Design Patterns

None

1.2 Industry Standards

XML
Java

1.3 Required Algorithms

1.3.1 Differences of sequence diagram instance between ZUML and TCUML

The difference of the model between ZUML and TCUML is discussed in the requirements. In this section we'll discuss the differences of Diagram instances. (The conversion is described by the SDs, so we won't discuss them in algorithm.)

1. The diagram's size, title node and properties will be converted by ZUML 2 TCUML converter component.
2. The owner of diagram is different
   ZUML – UML2 Interaction instance
   TCUML – Collaboration instance
3. for each of the contained graph nodes
   a) The element in semanticModel is different
      ZUML – UML2 Lifeline instance
      TCUML – Object instance
   b) ZUML has a contained graph node with typeInfo "HeaderCompartment", TCUML doesn't.
   c) ZUML may have the anchorages in the contained graph node with semanticModel "ExecutionOccurrence" reference, but TCUML will always have all the anchorages in current graph node.
   d) TCUML has no contained elements.
4. for each of the contained graph edges
   a) The element in semanticModel is different
      ZUML – UML2 Message instance
      TCUML – Link instance
   b) ZUML and TCUML has different contained graph nodes, but the TCUML's contained graph nodes are static.

1.3.2 How to handle comments in the sequence diagram

There are two types of comments, one is without connector and one is with connector. The comment nodes in sequence diagram are same for ZUML and TCUML, but the models are different. (See SDs-Pos.txt & SDs-Tc.txt)

ZUML: the comments are in the ownedElement element of UML2:Interaction, and it may have annotatedElement if it has a connector(I.E. to Lifeline). But you don't need to care about the annotatedElement as it is not needed in TC UML. The Comment also have name and body attributes. (When parsed into model bean, they will be instance variables.)
TCUML: the comments are in the ownedElement element of UML:Model, it has name, body and namespace sub-elements. (When parsed into model bean, they will be instance variables) The namespace will always be the Model instance.

How to convert: When converting the Interaction in model in XMISequenceDiagramConverterUtil.convertCollaborationInModel method, move all the Comment instances from Interaction instances' ownedElements to the Model instance's ownedElements and for each Comment instance, set its namespace as Model instance.

1.4 Component Class Overview

- **Interaction:** This interface represents <UML2:Interaction> element. It is the model class for Sequence Diagram in ZUML file (UML2).It extends GeneralizableElement interface. It also extends ZUML2TCUMLTransformer<Collaboration> interface so that it will be transformed to Collaboration.
- **ZUML2TCUMLTransformer<T extends ModelElement>:** This interface defines the method to transform the implementation UML2 model classes to corresponding UML1.4 (TCUML) model classes. And the TCUML classes must extend ModelElement class.
- **Lifeline:** This interface represents <UML2:Lifeline> element. It is the model class for a lifeline in Sequence Diagram in ZUML file (UML2).
- **Message:** This interface represents <UML2:Message> element. It is the model class for a message in Sequence Diagram in ZUML file (UML2).
- **EventOccurrence:** This interface represents <UML2:EventOccurrence> element. It is the model class for an event occurrence in Sequence Diagram in ZUML file (UML2).
- **ExecutionOccurrence:** This interface represents <UML2:ExecutionOccurrence> element. It is the model class for an execution occurrence in Sequence Diagram in ZUML file (UML2).
- **InteractionImpl:** This class extends GeneralizableElementAbstractImpl and implements Interaction interface, it represents <UML2:Interaction> element in ZUML file.

- **LifelineImpl:** This class extends ModelElementAbstractImpl and implements Lifeline interface, it represents <UML2:Lifeline> element in ZUML file.
- **MessageImpl:** This class extends ModelElementAbstractImpl and implements Message Interface, it represents <UML2:Message> element in ZUML file.
- **ExecutionOccurrenceImpl:** This class extends ModelElementAbstractImpl and implements ExecutionOccurrence interface, it represents <UML2:ExecutionOccurrence> element in ZUML file.
- **EventOccurrenceImpl:** This class extends ModelElementAbstractImpl and implements EventOccurrence interface, it represents <UML2:EventOccurrence> element in ZUML file.
- **XMISequenceDiagramConverterUtil:** This utility class is the main class in this component. It provides two post reading tasks the application can call: 1.Converting the Collaboration in Model for sequence diagram from ZUML to TCUML. 2. Converting the sequence diagram instances from ZUML to TCUML

1.5 Component Exception Definitions

- **IllegalArgumentException:** This exception is thrown if any argument is invalid in this component. (For example, null, empty argument sometime is not accepted.)

1.6 Thread Safety

This component is not thread-safe as the thread-safe is not a requirement of this component.

The XMISequenceDiagramConverterUtil utility class is stateless and thread-safe. But all the UML2 model classes are mutable and not thread-safe. It will not cause any problem because this component is used in SAX Parsing, and the SAX Parsing will be always in single thread.

To use this component in multi-threads environment, you need to synchronize the bean instance shared by multiple threads.

2. Environment Requirements

2.1 Environment

- Development language: Java1.5
- Compile target: Java1.5

2.2 TopCoder Software Components

- **XMI Reader UML Model Plugin 1.0**: This component is used to read the UML2 elements in the ZUML file to UML2 model classes.
- **ZUML 2 TCUML Converter 1.0**: The RenameConverter in this component will be used to rename the ownedBehavior element.
- **UML Model Action 1.0**: The model classes in the component are used in this component.
- **UML Model Collaborations 1.0**: The model classes in the component are used in this component.
- **UML Model Common Behavior 1.0**: The model classes in the component are used in this component.
- **UML Model Core 1.0**: The model classes in the component are used in this component.
- **UML Model Core Auxiliary Elements 1.0**: The model classes in the component are used in this component.
- **UML Model Core Classifiers 1.0**: The model classes in the component are used in this component.
- **UML Model - Model Management 1.0**: The model classes in the component are used in this component.
- **Diagram Interchange 1.0.1**: The diagram relative classes are defined in the component.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Names

com.topcoder.umltool.xmiconverters.poseidon5
com.topcoder.umltool.xmiconverters.poseidon5.model
com.topcoder.umltool.xmiconverters.poseidon5.model.impl

3.2 Configuration Parameters

None

3.3 Dependencies Configuration

3.3.1 You need to configure the XMI Reader UML Model Plugin 1.0 component correctly.

In the XMI Reader “handlers” configuration property, add the following handlers:

```
<Property name="UML2:Interaction">
   <Value>com.topcoder.xmi.reader.handlers.uml.model.XMI2ModelHandler</Value>
</Property>
<Property name="UML2:EventOccurrence">
   <Value>com.topcoder.xmi.reader.handlers.uml.model.XMI2ModelHandler</Value>
</Property>
<Property name="UML2:ExecutionOccurrence">
   <Value>com.topcoder.xmi.reader.handlers.uml.model.XMI2ModelHandler</Value>
</Property>
<Property name="UML2:Message">
```

```
    <Value>com.topcoder.xmi.reader.handlers.uml.model.XMI2ModelHandler</Value>
</Property>
<Property name="UML2:Lifeline">
    <Value>com.topcoder.xmi.reader.handlers.uml.model.XMI2ModelHandler</Value>
</Property>
```

In the ModelElementFactory “xml_to_element_mapping” configuration property, add the following mappings (UML:Collaboration mapping should be replaced):

```
<Value>UML2:Interaction,
com.topcoder.umltool.xmiconverters.poseidon5.model.impl.InteractionImpl</Value>
<Value>UML2:EventOccurrence,
com.topcoder.umltool.xmiconverters.poseidon5.model.impl.InteractionImpl.EvnetOccurrenceImpl
</Value>
<Value>UML2:ExecutionOccurrence,
com.topcoder.umltool.xmiconverters.poseidon5.model.impl.ExecutionOccurrenceImpl</Value>
<Value>UML2:Message,
com.topcoder.umltool.xmiconverters.poseidon5.model.impl.MessageImpl</Value>
<Value>UML2:Lifeline,
com.topcoder.umltool.xmiconverters.poseidon5.model.impl.LifelineImpl</Value>
```

3.3.2 You need to configure the ZUML 2 TCUML Converter 1.0 component correctly.

In the namespace for RenameConverter, add following to “ToRenameNames” property:

```
<Property name=”UML2:BehavioredClassifier.ownedBehavior”>
        <Value> UML:Namespace.ownedElement </Value>
</Property>
```

Assume the object name of RenameConverter instance in Object Factory is “converters:renameownedbehavior”, add following to “Converters” property in namespace of XMIConvertersUtil.config method:

```
<Property name=”UML:Collaboration”>
        <Value>converters:renameownedbehavior</Value>
</Property>
<Property name=”UML:Class”>
        <Value>converters:renameownedbehavior</Value>
</Property>
<Property name=”UML:UseCase”>
        <Value>converters:renameownedbehavior</Value>
</Property>
```

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

- Configure XMI Reader UML Model Plugin 1.0
- Configure RenameConverter, see 3.3.2, see test_files/XMIConvertersUtil.xml for detail

- Configure XMIReader (see test_files/reader_config.xml)
- Configure ModelElementFactory (see test_files/factory_config.xml)

4.3 Demo

4.3.1 Set up this component

See 4.2 to configure this component and dependent components.
The application uses XMI Reader to parse the ZUML file. Like this:

```
//Before creating XMIReader, it's required to configure RenameConverter,
//XMIReader and ModelElementFactory correctly. See demo.java

XMIReader reader = new XMIReader();

//Set up ModelElementFactory for UML:Diagram
DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler) reader.getHandler(
        "UML:Diagram");
handler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));

//Set up ModelElementFactory for UML:Model
XMI2ModelHandler xhandler = (XMI2ModelHandler) reader.getHandler(
        "UML:Model");
xhandler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));

(previous part is done in setup method)

//Configure the RenameConverter and XMIReader
XMIConvertersUtil.config(reader);

//It should process successfully
reader.parse(new File("test_files/zuml_1.xml"));
```

The application may use other components such as ZUML 2 TCUML Converter component to convert the ZUML file.

4.3.2 Convert the Collaboration in model

```
// get the Model instance from Model Manager; this is done by application, not in this component.
// Like this:
XMIReader reader = new XMIReader();

//Set up ModelElementFactory for UML:Diagram
DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler) reader.getHandler(
        "UML:Diagram");
handler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));

//Set up ModelElementFactory for UML:Model
XMI2ModelHandler xhandler = (XMI2ModelHandler) reader.getHandler(
        "UML:Model");
xhandler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));

(Previous part is done in setUp method)

 //Configure the RenameConverter and XMIReader
```

```
XMIConvertersUtil.config(reader);

reader.parse(new File("test_files/zuml_1.xml"));

XMI2ModelHandler handler = (XMI2ModelHandler) reader.getHandler(
        "UML:Model");
UMLModelManager manager = handler.getUmlModelManager();
Model model = manager.getModel();

//It should process successfully
XMISequenceDiagramConverterUtil.convertCollaborationInModel(model);
```

4.3.3 Convert the Sequence Diagrams instances

```
// get the Diagram instances from Model Manager; this is done by application,
// not in this component. like this:
XMIReader reader = new XMIReader();

//Set up ModelElementFactory for UML:Diagram
DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler) reader.getHandler(
        "UML:Diagram");
handler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));

//Set up ModelElementFactory for UML:Model
XMI2ModelHandler xhandler = (XMI2ModelHandler) reader.getHandler(
        "UML:Model");
xhandler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));
```

(The previous part is done in the setUp method)

```
//Configure the RenameConverter and XMIReader
XMIConvertersUtil.config(reader);

reader.parse(new File("test_files/zuml_1.xml"));

DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler) reader.getHandler(
        "UML:Diagram");
UMLModelManager manager = handler.getUmlModelManager();

List<Diagram> diagrams = manager.getDiagrams();

//It should process successfully
XMISequenceDiagramConverterUtil.convertSequenceDiagrams(diagrams);
```

4.3.4 Convert single Sequence Diagram instance

```
// get the Diagram instances from Model Manager; this is done by application,
// not in this component. like this:
XMIReader reader = new XMIReader();

//Set up ModelElementFactory for UML:Diagram
DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler) reader.getHandler(
        "UML:Diagram");
handler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));
```

```
//Set up ModelElementFactory for UML:Model
XMI2ModelHandler xhandler = (XMI2ModelHandler) reader.getHandler(
        "UML:Model");
xhandler.setModelElementFactory(new ModelElementFactory(
        ModelElementFactory.class.getName()));
```

(The previous part is done in the setUp method)

```
//Configure the RenameConverter and XMIReader
XMIConvertersUtil.config(reader);

reader.parse(new File("test_files/zuml_1.xml"));

DiagramInterchangeXMIHandler handler = (DiagramInterchangeXMIHandler)reader.getHandler(
        "UML:Diagram");
UMLModelManager manager = handler.getUmlModelManager();

List<Diagram> diagrams = manager.getDiagrams();

Diagram diagram = (Diagram) diagrams.get(0);

//It should process successfully
XMISequenceDiagramConverterUtil.convertSequenceDiagram(diagram);
```

5. Future Enhancements

None