# ZUML 2 TCUML Converter - Sequence Diagrams 1.0 Requirements Specification

## 1. Scope

### 1.1 Overview

The ZUML 2 TCUML Converter - Sequence Diagrams component provides means to help convert the zuml format from Poseidon to the tcuml format from 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.

### 1.2 Logic Requirements

"zuml" represents the file format used by Poseidon.

"tcuml" represents the file format used by TC UML Tool.

This component will provide classes that will help convert a zuml file into the TC UML Tool's internal model. The user will eventually use the TC UML Tool to save the model into tcuml format.

Since the .tcuml XMI files are the "views" of the internal model, they are described as a way to present the Tool's internal representation.

*1.2.1 Description of the UML elements / XMI format of the Sequence Diagrams*

1.2.1.1 zuml and tcuml representations of Sequence Diagrams

Attached to the distribution are two files containing the hierarchy of sequence diagram elements in both formats in a simplified version (see 2.1.2 for more details).

1.2.1.2 Description of the SD elements in UML 1.4 and tcuml's XMI format

In tcuml, the Collaboration has a CollaborationInstanceSet, which contains the list of Objects and Links (these are only referenced using the ids). The order of the Links shows the order in which they are executed, as shown in the Sequence Diagram (there will be only one Link on one level).

The Collaboration's ownedElements will contain the description of the actual Objects and Links (wrapped in Stimulus objects).

The Objects will have the "name" property and the "classifier" property, as a reference to a Classifier.

The Stimulus will contain the "communicationLink":Link property, which will have the description of the actual Link. The link will have two "connections":LinkEnd property, each containing the reference to the Object linked (specified as Instance type).

The Stimulus will also contain the "dispatchAction":Procedure property containing the actual Action that will determine the type of the link:

- synchronous call - CallOperationAction (isAsynchronous="false")
- asynchronous call - CallOperationAction (isAsynchronous="true")
- create call - CreateObjectAction
- send signal call - SendSignalAction
- return call - no procedure (or no action)

1.2.1.3 Description of the SD elements in UML 2.0 and zuml's XMI format

The Collaboration (which is used for Activity Diagrams also) contains an Interaction, which has 3 main parts: fragment, message and lifeline.

The "lifeline" section contains the actual Lifelines, with their description. A Lifeline contains an ordered list of EventOccurence and ExecutionOccurence objects (only referenced), describing the actual interaction with this Lifeline, following the top-down flow of time. A received arrow or an arrow that starts from this Lifeline translates to an EventOccurence, and an execution segment translates to ExecutionOccurence.

The “fragment” section contains the list of EventOccurence and ExecutionOccurence objects, with the information regarding the Lifelines they are related to and the actual Message associated with them. This list is ordered, so the flow in time can be followed by iterating the event and execution occurrences, as shown in the Sequence Diagram.

The “message” section contains the list of Message objects.

The type of the arrows is determined by the Message object’s attributes:

- synchronous call – Message has messageSort="synchCall" and there is an EventOccurence that has a “startExec” ExecutionOccurence for this message.
- asynchronous call - Message has messageSort="asynchCall"
- create call – similar as synchronous call, but has a stereotype.
- send signal call – N/A
- return call - Message has messageSort="synchCall" and there is an EventOccurence that has a “finishExec” ExecutionOccurence for this message.

1.2.1.4 UML 2.0 – UML 1.4 differences

The Lifeline objects in UML 2.0 correspond to the Object elements in UML 1.4.

The EventOccurence and ExecutionOccurence objects in UML 2.0 contain the information present in Links in UML 1.4 (order of calls and lifeline linkage).

The Messages in UML 2.0 contain the information present in Procedure (wrapping Actions) in UML 1.4.

*1.2.2 Convert the SD elements from zuml’s XMI to UML 1.4 model*

This component should be able to convert the sequence diagrams represented in zuml format (UML 2.0) in tcuml format (UML 1.4). This may be done either by using plugins for XMI Reader component, or by creating the UML 2.0 classes needed to represent the SDs as read from zuml, and convert them into tcuml format after the reading is over.

**1.3 Required Algorithms**

None.

**1.4 Example of the Software Usage**

The component will be used in the TopCoder UML Tool import .zuml files saved by Poseidon.

**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*

*SDs-Pos.txt:* is the file containing the hierarchy of sequence diagram elements in zuml format in a simplified version

*SDs-Tc.txt:* is the file containing the hierarchy of sequence diagram elements in tcuml format in a simplified version

*2.1.3 Environment Requirements*

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

*2.1.4 Package Structure*

com.topcoder.umltool.xmiconverters.poseidon5

## 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?*

XML

*3.2.2 TopCoder Software Component Dependencies:*

- XMI Reader 1.0
- XMI Reader UML Model Plugin 1.0
- UML Model components
- Configuration Manager 2.1.5

**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.