ZUML 2 TCUML Converter - Activity Diagrams 1.1 Requirements Specification

1. Scope

1.1 Overview

The ZUML 2 TCUML Converter - Activity Diagrams component provides means to help convert the zuml format from Poseidon to the tcuml format from TC UML Tool. This component provides the Activity 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.

Version 1.1 provides utility methods to fully convert the model and also the graph node structures. Version 1.0 only provides the UML model elements from UML 2.0 (as Poseidon uses UML 2.0). TC UML Tool uses UML 1.4, and this component must replace the UML 2.0 elements from the model with those from UML 1.4.

The Diagram Interchange representation of the activity diagrams is also different between Poseidon 1.5 and TC UML Tool, so this component must also convert the diagram interchange graph node structure. See the red requirements below for more details.

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 Activity Diagrams

1.2.1.1 zuml and tcuml representations of Activity Diagrams

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

1.2.1.2 Description of the AD elements in UML 1.4 and tcuml’s XMI format

In tcuml, the ActivityGraph represents the root element for the activity diagram model. It contains a CompositeState (under “top” property) which is the container for all the nodes (states) that belong to the activity graph. The ActivityGraph also contains a list with the edges (transitions), and the “context” element to which the activity graph belongs.

The list of nodes:

a) initial node – represented by UML:Pseudostate kind=”initial”

b) final node – represented by UML:FinalState

c) flow final node – represented by UML:FinalState + TaggedValue

d) fork node – represented by UML:Pseudostate kind=”fork”

e) join node – represented by UML:Pseudostate kind=”join”

f) decision node – represented by UML:Pseudostate kind=”choice”

g) merge node – represented by UML:Pseudostate kind=”junction”

h) object node – represented by UML:ObjectFlowState

i) call action – represented by UML:ActionState

j) send signal action – represented by SimpleState + TaggedValue

k) accept event action – represented by SimpleState + TaggedValue

The list of edges is made of UML:Transition elements, which should be containing a source and a target node. There is a problem here, though: the source and target nodes of a transition are not kept in the model at this moment, but in diagram interchange graph nodes. This should be fixed by TopCoder in the near future, so the designer should consider these as being present.

A Transition contains a guard of type UML:Guard, which has an expression of BooleanExpression type, and its body is the actual guard text displayed.

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

In zuml, the Collaboration (or any concrete BehavioredClassifier) has an Activity element (which represents the root element for the activity diagram model), which contains a list with the nodes (states) and a list with the edges (transitions).

The list of nodes:

a) initial node – represented by an UML2:InitialNode element

b) final node – represented by an UML2:ActivityFinalNode element

c) flow final node – represented by an UML2:FlowFinalNode element

d) fork node – represented by an UML2:ForkNode element

e) join node – represented by an UML2:JoinNode element

f) decision node – represented by an UML2:DecisionNode element

g) merge node – represented by an UML2:MergeNode element

h) object node – represented by an UML2:Pin element

i) call action – represented by an UML2:CallAction element

j) send signal action – represented by an UML2:SendSignalAction element

k) accept event action – represented by an UML2:AcceptEventAction element

The list of edges is made of UML2:ActivityEdge elements, each containing a source and a target node. Also, an ActivityEdge contains a guard of type UML2:OpaqueExpression, and its body is the actual guard text displayed.

1.2.2 Convert the AD elements from zuml's XMI to UML 1.4 model

This component should be able to convert the activity diagrams represented in zuml format (UML 2.0) to 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 ADs as read from zuml, and convert them into tcuml format after the reading is over.

1.2.3 Convert the Activity structures into ActivityGraph structures

Version 1.0 provides the model entities from UML 2.0 (as Poseidon 1.5 uses UML 2.0). Version 1.1 of this component will provide a utility method that will convert the Activity structures from the Model (UML 2.0 format) into the ActivityGraph structures (UML 1.4 format). The utility method will have a Model as an input and will return the ActivityGraphs.

The Activity nodes can be found in any place inside the Model, in the "ownedBehaviour" property of any concrete BehaviouredClassifier. This component will need to identify them, transform them into ActivityGraphs, remove them from the Model and return the list of ActivityGraphs. These will be stored by the application directly in the UMLModelManager from UML Model Manager component.

1.2.4 *Convert the ActivityDiagram graph node structures from Poseidon to TC UML Tool*

Version 1.1 will provide a utility method that will convert the GraphNode structure of the ActivityDiagrams from Poseidon's format to TC UML Tool's format. Only the graph nodes equivalent to those supported by TC UML Tool will be considered. The rest should simply be removed.

The GraphNode structure of the diagrams can be found by saving a project with an activity diagram and by inspecting the XMI content (both in Poseidon and TC UML Tool). There will be Diagram elements for every activity diagram contained in the model. The GraphNodes and GraphEdges contained in the Diagram node represent the visual representation of every activity diagram model element, containing at their turn graph nodes for the name, stereotypes, ...

Note that there is a general converter component "ZUML to TCUML Converter" that takes care of removing the title graph node of the diagram, which has no correspondent in TC UML Tool. This component should not duplicate that functionality.

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*

PosActivity.rtf: is the file containing the hierarchy of activity diagram elements in zuml format in a simplified version

TcActivity.rtf: is the file containing the hierarchy of activity 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.