Application Screening Requirements Specification

1. Scope

1.1 Overview

The TopCoder Software application methodology requires Project Managers to follow a strict guideline for creating requirement specifications. This component will aid Project Managers in creating their UML models by creating warnings when the proper format is not followed. By introspecting the XMI submitted by the Project Manager, the component will validate numerous rules. Additionally, this component will set up future stages of the application methodology based on the UML models created by the Project Manager.

1.2 Logic Requirements

1.2.1 Inputs

It is assumed the component will receive an XMI version 2.1 file as input. A sample XMI file may be found in the distribution.

1.2.2 Diagram Validation

Each XMI will be validated to make sure at least one use case diagram exists and at least one activity diagram exists.

Each use case diagram will be validated for the following
- at least one actor
- at least one use case

Each activity diagram will be validated for the following
- One initial state
  - The initial state must be labeled with a name
  - For example :
    <UML:Pseudostate xmi.id = 'I1aa8b42m10480f27c47mm7b51' name = 'Start' visibility = 'public' isSpecification = 'false' kind = 'initial'>

- One final state
  - The final state must be labeled with a name
  - For example:
    <UML:FinalState xmi.id = 'I1aa8b42m10480f27c47mm76c5' name = 'End' visibility = 'public' isSpecification = 'false'>

1.2.3 Diagram Validation Output

All violations of the Diagram Validation will be created as errors. The following are the error messages to be generated:
- Missing activity diagram or use case will state what is missing.
- Missing an actor will state the name of the use case diagram missing the actor.
- Missing a use case will state the name of the use case diagram without a single use case.
- If an initial state is missing or missing a label an error will be generated stating the name of the activity diagram and stating the error.
- If a final state is missing or missing a label an error will be generated stating the name of the activity diagram and stating the error.

1.2.4 Naming Convention Validation

Verify all use case bubbles have at least one corresponding activity diagram. The naming convention is for each use case bubble named *X*, there will exist at least one activity diagram named *X Activity – Y (- Y only exists if there is more than one activity diagam)*, where *X* is the name of the use case and *Y* is the name of the activity diagram. The name matching will be case insensitive and ignore the phrase “Activity Diagram”. For example, in the attached XMI and ZUML, you’ll notice there exists a Register Use Case and a Register Activity Diagram. This example would pass screening.

1.2.5 Naming Convention Output

All violations of the naming convention will be an error. These errors will be outputted as an XML string with a list of all errors. If there are no errors the user of the component will easily be able to know there were zero errors. Two types of errors will be generated:

- Use case with no corresponding activity diagrams – This error message will contain the name of the use case
- Activity diagram with no corresponding use cases – This error message will contain the name of the activity diagram.

1.2.6 Activity Diagram Path Generation

The second validation performed by this component will be generating an XML output of all the unique paths through an activity diagram. All unique paths will be traversed and created in the XML output document.

- Each use case will have one to many activity diagrams following the naming convention stated above.
- Each activity diagram will have one to N number of paths.
- The following are required path elements that will be captured in the XML document:
  - Numbered paths – each path will be numbered in sequential order and each number will be unique for each XML generated.
  - Each path node name will be the activity diagram bubble name and the guard followed, if the guard is named. If there is no guard, just the activity diagram bubble name will be used.

1.2.7 Activity Diagram Path Generation Output

For example taking the Register Activity Diagram the following XML would be created. The XML format is up to the designer, this is just an example. The designer will come up with the best format

```
<ActivityDiagramPaths>
  <UseCase name="Register">
    <ActivityDiagram name="Register Activity Diagram">
      <Path>
        <Number>1</Number>
        <Node>User Fills Out Registration Form - no account exists for email</Node>
        <Node>validate user - information valid</Node>
        <Node>Send success</Node>
      </Path>
      <Path>
        <Number>2</Number>
        <Node>User Fills Out Registration Form - no account exists for email</Node>
        <Node>validate user - else</Node>
        <Node>notify user</Node>
        <Node>user fixes error</Node>
        <Node>validate user - information valid</Node>
        <Node>Send success</Node>
      </Path>
      <Path>
        <Number>3</Number>
        <Node>User Fills Out Registration Form - else</Node>
        <Node>Redirect user to Login page</Node>
      </Path>
    </ActivityDiagram>
    <ActivityDiagram name="Sample more activity diagrams if more than one exist for the use
case">
      <Path>
        <Number>1</Number>
      </Path>
      <Path>
        <Number>N</Number>
      </Path>
    </ActivityDiagram>
  </UseCase>
  <UseCase name="additional use cases">
  </UseCase>
</ActivityDiagramPaths>
```

1.2.8 XMI Parser

If additional classes are required for the XMI Parser for the UseCaseDiagram and ActivityDiagram, these classes will be added directly to the XMI parser.

1.3 Required Algorithms

None

1.4 Example of the Software Usage

Similar to the TopCoder Component automated screening process, the application screening process will be automated. One of the tests will be the inspection of the XMI submitted by the screening.

1.5 Future Component Direction

Additional rules may be created.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

XMI file input version 1.2

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.apps.screening.applications.specification

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

As defined above.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

- XMI Parser

**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.

3.4 Required Documentation

None

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- XML Schema Definition

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.