USE/OCL Modeling of the Formal Specification

For GMoDS Visualizer and Test Driver

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Mike Fraka
CIS 85 – MSE Project
Kansas State University
Table of Contents

1 Introduction........................................................................................................................................... 3
2 References............................................................................................................................................ 3
3 USE Modeling ....................................................................................................................................... 3
   3.1 Limitations of USE 2.6.2.................................................................................................................. 3
   3.2 Modeling EventScriptImpl::addEvent in USE............................................................................... 3
      3.2.1 Modeling a POSITIVE_TRIGGER event ................................................................................. 4
      3.2.2 Modeling an ACHIEVED event ............................................................................................... 9
      3.2.3 Modeling a MODIFIED event ...............................................................................................10
1 Introduction
This document the validation of the formal specification of the method EventScriptImpl::addEvent with USE version 2.6.2.

2 References
2. “USEOCLmodeling.zip” available at http://people.cis.ksu.edu/~mfraka/FrakaMSE.html. This file contains the USE model and command scripts used to model the pre and post conditions of the method EventScriptImpl::addEvent.

3 USE Modeling
An action item from MSE presentation 2 was:

- Perform USE/OCL modeling of state snapshots to validate the pre and post conditions of the EventScriptImpl::next method in the formal specification.

I performed this modeling using USE 2.6.2.

3.1 Limitations of USE 2.6.2
USE 2.6.2 does not support the OCL “isSent” operator (denoted ‘^’) necessary for the most important post conditions of the EventScriptImpl::next method. A MODIFIED event type causes the next method to send the message “modifyInstanceGoal” to GMoDS, and all other event types cause the next method to send the message “event”. In addition, USE 2.6.2 does not support the “init” constraint on a class attribute. Finally, I was unable to get USE 2.6.2 to allow more than 1 local variable to be defined in a “let” expression.

As a result of these limitations, I requested and was granted permission to model the EventScriptImpl::addEvent method.

3.2 Modeling EventScriptImpl::addEvent in USE
Table 1 below lists the scripts contained in [2] that I used to model the formal specification of the method EventScriptImpl::addEvent.

Table 1 USE scripts modeling EventScriptImpl::addEvent

<table>
<thead>
<tr>
<th>Script</th>
<th>Comment</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTD.use</td>
<td>OSE class, association, and constraint model</td>
<td>See [1] section 9 p. 28.</td>
</tr>
<tr>
<td>gtd-valid-pt.cmd</td>
<td>Snapshot of pre state adding a valid #POSITIVE_TRIGGER</td>
<td>Figure 1</td>
</tr>
</tbody>
</table>
### 3.2.1 Modeling a POSITIVE_TRIGGER event

Figure 1 below shows an object diagram of a pre state when adding a valid #POSITIVE_TRIGGER event.

<table>
<thead>
<tr>
<th>Script</th>
<th>Comment</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>gtd-valid-post.cmd</td>
<td>Script to invoke pre/post conditions (valid post conditions)</td>
<td>Figure 2</td>
</tr>
<tr>
<td>gtd-invalid-post.cmd</td>
<td>Script that invokes pre/post conditions (invalid post conditions)</td>
<td>Figure 3</td>
</tr>
<tr>
<td>gtd-invalid-specevt.cmd</td>
<td>Snapshot of pre state adding an invalid #POSITIVE_TRIGGER due to an invalid SpecificationEvent ID.</td>
<td>Figure 5</td>
</tr>
<tr>
<td>gtd-invalid-specgoal.cmd</td>
<td>Snapshot of pre state adding an invalid #POSITIVE_TRIGGER due to an invalid ParamterizedSpecificationGoal ID.</td>
<td>Figure 6</td>
</tr>
<tr>
<td>gtd-invalid-pt-paramnames.cmd</td>
<td>Snapshot of pre state adding an invalid #POSITIVE_TRIGGER due to an invalid parameter name.</td>
<td>Figure 7</td>
</tr>
<tr>
<td>gtd-invalid-achieved.cmd</td>
<td>Snapshot of pre state adding an invalid #ACHIEVED event due to referencing a non-leaf goal.</td>
<td>Figure 8</td>
</tr>
<tr>
<td>gtd-invalid-modified.cmd</td>
<td>Snapshot of pre state adding an invalid #MODIFIED event due to no parameters specified.</td>
<td>Figure 9 Figure 10</td>
</tr>
<tr>
<td>gtd-valid-modified.cmd</td>
<td>Snapshot of pre state adding a valid #MODIFIED event.</td>
<td>Figure 11 Figure 12</td>
</tr>
<tr>
<td>gtd-invalid-modified-paramnames.cmd</td>
<td>Snapshot of pre state adding an invalid #MODIFIED event due to mismatch on parameter names.</td>
<td>Figure 13</td>
</tr>
</tbody>
</table>
Figure 1: Valid snapshot prior to adding a POSITIVE_TRIGGER event

Figure 2 below shows that the pre conditions and post conditions are valid for the above snapshot when executing the script gtd-valid-post.cmd manually.

Figure 2: Valid pre/post conditions when adding a POSITIVE_TRIGGER event
Figure 3 Invalid post conditions

Figure 3 above shows that the post conditions are violated in the above snapshot if the script gtd-invalid-post.cmd is executed.
Figure 4 above shows that executing addEvent twice for the same event violates the “NotInScript” precondition.

Figure 5 above shows that the script gtd-invalid-specEvt.cmd violates the “ValidSpecEvent” precondition.
Figure 6 Invalid Specification Goal

Figure 6 above shows the script gtd-invalid-specgoal.cmd violates the "ValidSpecGoal" and "ValidSpecEvent" pre conditions. Figure 7 below shows the script gtd-invalid-pt-paramnames.cmd violates the "ValidTriggerParamNames" pre condition.

Figure 7 Invalid Parameter Names for a Positive Trigger Event
3.2.2 Modeling an ACHIEVED event

Figure 8 Above shows that the script gtd-invalid-achieved.cmd violates the “ValidAchievedEvent” pre condition. A slight modification of this script would violate the “ValidFailedEvent” pre condition.
3.2.3 Modeling a MODIFIED event

Figure 9 above shows a snapshot of an invalid #MODIFIED event which is invalid because it specifies no parameters.

Figure 9 Valid snapshot prior to adding an invalid MODIFIED event with no parameters
Figure 10 Invalid MODIFIED event with no parameters

Figure 10 above shows that invoking the addEvent violates the “ModifiedReqParam” precondition for the above snapshot.
Figure 11 above shows a snapshot of the pre state when adding a valid #MODIFIED event. Figure 12 below shows that invoking addEvent on this snapshot produces valid pre conditions.
Figure 12 Valid preconditions adding a MODIFIED event

Figure 13 below shows that the script gtd-invalid-modified-paramnames.cmd violates the “ValidModifiedParamNames” pre condition.

Figure 13 Invalid parameter name in a MODIFIED event