GMoDS Test Driver and Visualizer
MSE Project Presentation III

Mike Fraka
Agenda

- Action Items from Presentation II
- Component Design
- Assessment Evaluation
- Project Evaluation
- Demonstration
- Questions/Comments
Action Items from Presentation II

- Perform USE/OCL modeling of EventScriptImpl::addEvent(e : GoalEventImpl)
  - Limitations of USE 2.6.2
    - Does not support the OCL “isSent” operator (denoted ‘^’) necessary for the most important post conditions of the EventScriptImpl::next method
    - Does not support the “init” constraint on a class attribute
    - Unable to get more than 1 local variable defined in a “let” expression
Formal Specification

```plaintext
context EventScriptImpl::addEvent(e : GoalEventImpl)
-- The event does not already exist in the script
  pre NotInScript: event->excludes(e)
-- The added event's type is valid
  pre ValidType:
    e.type = #ACHIEVED or e.type = #FAILED or e.type = #POSITIVE_TRIGGER or
    e.type = #NEGATIVE_TRIGGER or e.type = #MODIFIED
-- At least one parameter must be provided if type is #MODIFIED
  pre NotModifiedSeqParams: e.type = #MODIFIED implies e.param->size > 0
-- A #MODIFIED event's parameter names must match specification goal's parameter names
  pre ValidModifiedParamNames:
    e.type = #MODIFIED and e.param->size > 0 implies
    e.param->forall(ep | goods.specTree.goal->exists(ag | sg.id = e.specGoalID and
    sg.param->exists(sep | sep.id = ep.id)))
-- The added event refers to a ParameterizedSpecificationGoal that
  exists in GMoDS specification tree
  pre ValidSpecGoal: goods.specTree.goal->exists(se | sg.id = e.specGoalID)
-- An #ACHIEVED event will access the special 'ACHIEVED' event of GMoDS and
  must apply to a leaf specification goal.
  pre ValidAchievedEvent: e.type = #ACHIEVED implies e.specEventID = 'ACHIEVED' and
  goods.specTree.goal->exists(se | sg.id = e.specGoalID and sg.issLeaf = true)
-- A #FAILED event will access the special 'FAILED' event of GMoDS and
  must apply to a leaf specification goal.
  pre ValidFailedEvent: e.type = #FAILED implies e.specEventID = 'FAILED' and
  goods.specTree.goal->exists(se | sg.id = e.specGoalID and sg.issLeaf = true)
-- If the type is #POSITIVE_TRIGGER or #NEGATIVE_TRIGGER
  -- the added event refers to a SpecificationEvent that exists in GMoDS specification tree,
  -- the event's specification goal is a leaf goal, and the event's specification event's
declared goal exists, and the event's specification goal is either the goal on which the
  -- event was declared or a descendant of the declared goal.
  pre ValidSpecEvent: e.type = #POSITIVE_TRIGGER or e.type = #NEGATIVE_TRIGGER implies
    (goods.specTree.event->exists(se | se.id = e.specEventID and
    goods.specTree.goal->exists(se, sg | sg.issLeaf = true and sg.id = e.specGoalID and
dg.id = se.declareGoalID and dg.descendantsAndSelf()->includes(sg))
-- if the type is #POSITIVE_TRIGGER or #NEGATIVE_TRIGGER
  -- then it must provide the parameters required by the specification event
  pre ValidTriggerParamNames:
    e.type = #POSITIVE_TRIGGER or e.type = #NEGATIVE_TRIGGER implies
    goods.specTree.event->exists(se | se.id = e.specEventID and
    se.param->forall(sep | e.param->exists(sep | sep.id = sep.id))|
-- The event is added to the script if all preconditions are met
post NotInScript: event->includes(e)
-- The number of events is increased by 1
post OneMoreEvent: (event->assSet -> event@pre->assSet)->size = 1
-- The new event is appended to the end of the script
post Appended: event->last = e
```
Specification Tree Modeled
Snapshot of Valid Preconditions for Adding a POSITIVE_TRIGGER
Evaluating Pre/Postconditions

```plaintext
Shortcut to USE 2.6.2 use.bat

```/gnds/gtd-valid-pm.cmd> finsert <g11, g11P> into InstGoalParams
```/gnds/gtd-valid-pm.cmd> finsert <g21, g21P> into InstGoalParams
```/gnds/gtd-valid-pm.cmd> use> open ../gnds/gtd-valid-post.cmd
```/gnds/gtd-valid-post.cmd> fopenter s1 addEvent(gel)
precondition 'MotInScript' is true
precondition 'ValidType' is true
precondition 'ModifiedReqParam' is true
precondition 'ValidModedParamNames' is true
precondition 'ValidSpecGoal' is true
precondition 'ValidAchievedEvent' is true
precondition 'ValidFailedEvent' is true
precondition 'ValidSpecEvent' is true
precondition 'ValidTriggerParamNames' is true
```/gnds/gtd-valid-post.cmd> -- Enforce post conditions
```/gnds/gtd-valid-post.cmd> finsert (s1,gel) into Events
```/gnds/gtd-valid-post.cmd> !opexit
postcondition 'NovInScript' is true
postcondition 'OneMoreEvent' is true
postcondition 'Appended' is true
```/gnds/gtd-valid-post.cmd> use>
```
USE Modeling Affected the Formal Specification

- Corrected OCL syntax
  - <> instead of !=
  - “…implies…” instead of “if …implies…”

- Realized I left out preconditions
  - ValidModifiedParamNames
  - ValidTriggerParamNames
GMoDS Test Driver
Random Events
GMoDS Visualizer Instance Tree View
GMoDS Visualizer Part
InstanceTreeUI update(arg : Object)
FlashDaemon run()
## Assessment Evaluation Summary

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>Test Case Title</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC.GTD-1</td>
<td>Load Event Script</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GTD-2</td>
<td>Event Script Operation</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GTD-3</td>
<td>Random Event Script Operation</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GTD-4</td>
<td>Save Event Script</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-1</td>
<td>Display Specification Tree</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-2</td>
<td>Display Instance Tree</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-3</td>
<td>Zooming</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-4</td>
<td>Show/Hide Instance Goals of Specific Types</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-5</td>
<td>Show/Hide All Specification Goal Parameters</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-6</td>
<td>Show/Hide All Instance Goal Parameters</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-7</td>
<td>Show/Hide Specific Instance Goal Parameters</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-8</td>
<td>Collapse/Expand Instance Goal Sub-tree</td>
<td>Pass</td>
</tr>
<tr>
<td>TC.GV-9</td>
<td>Change Instance Goal State Colors</td>
<td>Pass</td>
</tr>
<tr>
<td>Test Case</td>
<td>Failure</td>
<td>Action to Resolve</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>TC.GTD-2 – Event Script Operation</td>
<td>Failed to catch an IllegalGoalEventException thrown when an invalid GoalEvent is encountered during event script play.</td>
<td>Deferred and accumulated IllegalGoalEventExceptions while it is unsafe to throw them. Added try/catch surrounding applicable code in the play, pause, and hasNext methods. Added throw of the cumulative IllegalGoalEventException when it becomes safe to do so.</td>
</tr>
<tr>
<td>TC.GV-1 – Display Specification Tree</td>
<td>Rightmost specification goal cutoff in drawing.</td>
<td>Add margins to the image on the right and bottom.</td>
</tr>
<tr>
<td>TC.GV-3 – Zooming</td>
<td>Specification tree right/bottom cutoff when zoomed in.</td>
<td>The content of the JScrollPane, the AbstractCanvas, resets its preferred size and then calls the method “revalidate” after a zoom.</td>
</tr>
<tr>
<td>TC.GV-5 - Show/Hide All Specification Goal Parameters</td>
<td>Specification tree trigger parameters not hidden.</td>
<td>Added a flag to AbstractTriggerUI to prevent drawing parameters when hidden.</td>
</tr>
<tr>
<td>TC.GV-2 – Display Instance Tree</td>
<td>MODIFIED event results in incorrect colors and flashing</td>
<td>Reset flash count if instance goal is modified.</td>
</tr>
</tbody>
</table>
## Project Evaluation

### Project Size

<table>
<thead>
<tr>
<th>Phase</th>
<th>SLOC Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>4K</td>
</tr>
<tr>
<td>Phase 2</td>
<td>7K</td>
</tr>
<tr>
<td>Phase 3</td>
<td>8.3K (Actual)</td>
</tr>
</tbody>
</table>

### Project Duration

<table>
<thead>
<tr>
<th>Phase</th>
<th>Estimated End Date</th>
<th>Actual End Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>11/10/2010</td>
<td>11/22/2010</td>
</tr>
<tr>
<td>Phase 2</td>
<td>2/14/2011</td>
<td>1/13/2011</td>
</tr>
<tr>
<td>Phase 3</td>
<td>4/14/2011</td>
<td>3/16/2011</td>
</tr>
</tbody>
</table>
Task Breakdown in Phase 1

Percent of Time Spent on Each Task

- Coding: 51.55%
- Designing: 12.18%
- Documenting: 30.18%
- Email: 4.36%
- Presenting: 0.28%
- Reading: 0.46%
- Resolving Defect: 0.00%

Total: 1.00
Task Breakdown in Phase 2

Percent of Time Spent on Each Task

- Coding: 29.15%
- Designing: 17.52%
- Documenting: 45.36%
- Presenting: 6.20%
- Reading: 0.72%
- Resolving defect: 1.03%

Total: 100.00%
Task Breakdown in Phase 3

Percent of Time Spent on Each Task

Total

- Coding: 36.26%
- Designing: 5.00%
- Documenting: 38.73%
- Modeling: 0.34%
- Reading: 4.00%
- Resolving defect: 7.73%
- Rework: 5.16%
- Testing: 0.00%
Product Quality
Rework Ratio

\[ RW = \frac{E_{Defects}}{E_{Development}} \]

Rework Ratio

![Graph showing the trend of Rework Ratio from 10/1/2010 to 3/1/2011. The Rework Ratio increases over time.]
Product Quality
Mean Time between Failures

MTBF

Cumulative Testing Time (Hrs)
Future Work

- Dynamic specification tree display
- Display events executed in the GMoDS Test Driver
Demonstration
Questions/Comments