Enabling Automated Traceability Maintenance by Recognizing Development Activities Applied to Models

Patrick Mäder, Orela Gotel and Ilka Philippow

23rd International Conference on Automated Software Engineering
September 17, 2008
- **Requirements traceability**
- Our approach
- Traceability maintenance rules
- Prototype and preliminary validation
- Conclusion and future work
Traceability of development processes

- Allows to follow the development process and relates its intermediate products

Requirements traceability can support

- Analyzing the impact of changing requirements
- Verifying the implementation of requirements
- Program comprehension
- Regression test
- Reuse …
Problems

- Necessity for manual work to handle relations (Large numbers of relations even for small systems)
- Accurate set required for comprehensive results
- Insufficient method and tool support

Focus

- Maintenance of traceability relations during evolution and refinement of structural UML models
- Requirements traceability
- Our approach
- Traceability maintenance rules
- Prototype and preliminary validation
- Conclusion and future work
Approach to tackle the problem of traceability decay due to evolution of related model elements

- Stage 1: Capturing elementary changes to model elements and generating events
- Stage 2: Recognizing the wider development activity applied to the model element, as comprised several elementary changes
- Stage 3: Updating the traceability relations associated with the changed model element
What do events consist of?

1) Elementary change type: ADD, DEL, MOD

2) Impacted element type: class, component, package, attribute, method, association, inheritance, dependency

3) Properties of the changed element
   - For example “class”: name, id, stereotype, parent.id, parent.name, parent.elementtype
   - For MOD pre- and post-change (preMOD/postMOD)
Example: Convert attribute to a class

Change of a traced use case
(Numbers on elements depict OUT−IN trace relations; relations backward from dependent to independent element)

Event 1: Del(UID=149531614123456789) {Traced: 'Traced'} {Name: 'Gello GmbH (Class' ; ...)
Event 2: Del(UID=12345) {Traced: 'Traced'} {Name: 'Audio System'; ...})
Event 3: Add(UID=23456) {Name: 'Audio System'} {Role: 2, Navigations: navigable; ...)}
Main challenges

- Relate several elementary changes to one activity
- Recognize different orders of the same set of elementary changes as one development activity
- Recognize differing sets of elementary changes as the same development activity
- Requirements traceability
- Our approach
- **Traceability maintenance rules**
- Prototype and preliminary validation
- Conclusion and future work
Traceability maintenance rules

- Patterns of development activities that require traceability updates and directions to perform it

Status

- XML rule-set with 21 rules (67 alternatives) capturing 38 development activities
Rule structure

- Mask – defines required properties of an event as static values or references (Boolean expressions possible)
- Change sequence – groups several masks to recognize whole development activity
- Alternative sequences – group different change sequences performing same activity
Rule example: Convert attribute to a class

One change sequence consisting of four masks
Rule application process

1) put event in EventCache

2) try to complete OpenActivities

3) browse RuleCatalog for rules with matching TriggerEvent

[on assignment] search EventCache for additional matching events

[if activity is completed] do link update

delete oldest event from EventCache
Limitations

- Only predefined activities can be recognized
- Current rule-set is reusable and stable, but don’t claim to reflect all possible activities

Support for rule definition?

- Rule editor and validator available
- Requirements traceability
- Our approach
- Traceability maintenance rules
- **Prototype and preliminary validation**
- Conclusion and future work
traceMaintainer

- Prototype implementation to be able to validate the approach
- Tool-independent rule engine
- Adapter per CASE tool necessary: Sparx EA, ARTiSAN Studio (prelim.); and EXTESSY ToolNET available
Two experiments using *traceMaintainer* to explore the following objectives:

1) Assess whether our rules account for all changes and the recognition of common development activities; and

2) Determine whether the correct rules are fired when there is variation in task execution.
**Experiment**

- Two developers spent three hours on two sample projects respectively
- Task: refining analysis model into design model
- Detailed analysis of refinement activities
- Definition of two possible paths of execution in terms of elementary changes resulting in the same model

<table>
<thead>
<tr>
<th></th>
<th>Library system</th>
<th>Wiper system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Changes</td>
<td>136</td>
<td>192</td>
</tr>
<tr>
<td>Activities</td>
<td>35</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td>126</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>46</td>
</tr>
</tbody>
</table>
Results

- Two scenarios used for pre-examination of the rule catalog leading to minor adjustments on two rules
- OBJ1: Development activities of the remaining six scenarios could be recognized with total accuracy
- OBJ2: Each scenario could also be recognized in five different orders of the same events
- Rules catalog sufficient for context and captured variety in which a practitioner could create the final design model
- Requirements traceability
- Our approach
- Traceability maintenance rules
- Prototype and preliminary validation
- Conclusion and future work
Conclusion

- Approach for the recognition of activities applied to models in the context of UML-based software development
- Rules to cope with the high variability in task execution
- Main motivation: maintenance of traceability relations while a project proceeds, addressing the problem of traceability decay

Ongoing and Future work

- More empirical results
- Tool-supported creation of new rules (rule recorder)
- Backward recognition of rules (undo function)
Thank you. Patrick Mäder
## Scenarios

<table>
<thead>
<tr>
<th></th>
<th>Library system</th>
<th>Wiper system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>ADD</td>
<td>45</td>
<td>84</td>
</tr>
<tr>
<td>DEL</td>
<td>17</td>
<td>56</td>
</tr>
<tr>
<td>MOD</td>
<td>74</td>
<td>52</td>
</tr>
<tr>
<td>Changes</td>
<td>136</td>
<td>192</td>
</tr>
<tr>
<td>Activities</td>
<td>35</td>
<td>49</td>
</tr>
</tbody>
</table>