Using Ontologies for Flexibly Specifying Multi-User Processes

ICSE 2010 Workshop on Flexible Modeling Tools
Cape Town, South Africa, 2 May 2010

Gregor Gabrysiak, Holger Giese and Andreas Seibel
System Analysis and Modeling Group
Hasso Plattner Institute
University of Potsdam
Formal Tools

Pros
- automatic detection of errors
- maintenance throughout the model is simple
- reusable throughout a project

Cons
- restricted by metamodel
- early commitment
- overhead for small projects

Figure: UML Model (MS Visio)
Informal and General-Purpose Tools

Pros
- easy to use
- everything can be captured
- degrees of freedom are similar to whiteboards

Cons
- no metamodel
- changing references
- presentable, rarely reusable

Figure: Process Model (Keynote)
Ontologies as Flexible Metamodels

- less restrictive
  - missing concepts can be added on demand
- capable of capturing the modeler’s intent
  - a model can always be interpreted with its corresponding metamodel
Using a Formal Modelling Tool

Metamodel (Formal Tool)

User Generated Model

Gregor Gabrysiak, FlexiTools 2010, 2 May 2010
Extending the Meta Model

Metamodel

Project-Specific

User Generated Model
Using Ontologies as Meta Model

Basic Ontology

Project-Specific

User Generated Model

Gregor Gabrysiak, FlexiTools 2010, 2 May 2010

Introducing Ontologies 7
Using the terminology defined in the ontology, an analyst can...

- describe an observed situation
- describe the follow-up state
- the difference between both specifies an action
  - specified in the terminology defined in the ontology
Creating Behavioral Specifications II

Send Proposal

: Proposal

this : Customer

has

: EMail

b : Assistant

++ attached

++ sender

++ receiver

Gregor Gabrysiak, FlexiTools 2010, 2 May 2010
So, what do we have?

- behavioral and situational specification of a process
  - formal specifications that can be simulated
- all specifications reference elements of the flexible metamodel
  - specified in the terminology defined in the ontology
  - specification is affected by changes in the ontology
So, what do we have?

- behavioral and situational specification of a process
  - formal specifications that can be simulated
- all specifications reference elements of the flexible metamodel
  - specified in the terminology defined in the ontology
  - specification is affected by changes in the ontology
So, what do we have?

- behavioral and situational specification of a process
  - formal specifications that can be simulated
- all specifications reference elements of the flexible metamodel
  - specified in the terminology defined in the ontology
  - specification is affected by changes in the ontology
When Working with Flexible Metamodels...

We have to deal with ...

- addition, modification, and removal of
- classes, properties, methods, and associations
When Working with Flexible Metamodels...

We have to deal with ...

- addition, modification, and removal of
- classes, properties, methods, and associations

Worst Case Scenario
modeled specifications become **unreadable** with modified metamodel
## Modifications of the Ontology

<table>
<thead>
<tr>
<th></th>
<th>Addition</th>
<th>Removal</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>✓</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Property</td>
<td>✓</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>Association</td>
<td>✓</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

**Addition of any new Element**

- not referenced yet $\implies$ no action necessary
- save as often as possible
  - small deltas between versions
  - old version of the ontology is kept
  - when saving, *IDs* can be added
# Modifications of the Ontology

<table>
<thead>
<tr>
<th></th>
<th>Addition</th>
<th>Removal</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>✓</td>
<td>✓ (user assisted)</td>
<td></td>
</tr>
<tr>
<td>Association</td>
<td>✓</td>
<td>✓ (user assisted)</td>
<td></td>
</tr>
</tbody>
</table>

## Removal of any referenced Element

- **Class**: dangling references are redirected (recursively) to the corresponding superclass
- **Property**: either delete references or move to superclass
- **Association**
  - pointing to deleted class: redirect to superclass
  - else: either delete references or propose substitutes
# Modifications of the Ontology

<table>
<thead>
<tr>
<th></th>
<th>Addition</th>
<th>Removal</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>✓</td>
<td>✓</td>
<td>✓ (user assisted)</td>
</tr>
<tr>
<td>Property</td>
<td>✓</td>
<td>✓ (user assisted)</td>
<td>✓ (user assisted)</td>
</tr>
<tr>
<td>Association</td>
<td>✓</td>
<td>✓ (user assisted)</td>
<td>✓ (user assisted)</td>
</tr>
</tbody>
</table>

## Handling Ontology Modifications

- **Addition of any new Element**
  - not referenced yet
    - no action necessary
  - save as often as possible
  - small deltas between versions
  - old version of the ontology is kept
  - when saving, IDs can be added

- **Removal of any referenced Element**
  - Class: dangling references are redirected (recursively) to the corresponding superclass
  - Property: either delete references or move to superclass
  - Association
    - pointing to deleted class: redirect to superclass
    - else: either delete references or propose substitutes

- **Modification of any referenced Element**
  - modeler’s intent is unclear
  - syntactical change (**Bosss** becomes **Boss**):
    - **same concepts** apply, references still valid
    - IDs can be used to redirect from **Bosss** to **Boss**
  - semantical change (**+getX()** becomes **+setX()**):
    - **different concepts** apply, references invalid
    - remove old version + add new version
Conclusions

- ontologies as flexible metamodels
- concepts for handling metamodel changes
- implementation in Eclipse & EMF
  - automatic reload of changes in metamodel
  - Addition of elements fully functional