Achieving interoperability between WebODE and Protégé2000

Oscar Corcho
ocorcho@fi.upm.es

Ontology Group
Laboratorio de Inteligencia Artificial
Facultad de Informática. Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain
Table of contents

• The ontology translation problem
  – Translation problems
  – Current situation
  – Technology needed for transformation

• Transformation from WebODE to Protégé2000
  – WebODE vs Protégé2000 knowledge models
  – Transformation proposals
  – Layered transformations
    • Transformations at the knowledge level
    • Transformations at the symbol level
    • Transformations of user interfaces/visualization

• Future work
“The ontology translation problem appears when we decide to reuse an ontology (or part of an ontology) using a tool or language that is different from those in which the ontology is available”

• Translation problems
  – Different formats
    • Each language/tools stores and reads ontologies using different syntax
  – Different underlying KR paradigms
    • Frames, first order logic, description logic, semantic networks, conceptual graphs, and combinations
  – The knowledge lost in transformations should be preserved
    • In case that ontologies are restructured and translated back to the original language/tool
Current situation

• Existing approaches to ontology translation
  – Ontolingua
    • Ad-hoc translators, difficult to understand and maintain
      – Not documented
      – Translation decisions “hidden” in their code
  – “The family of languages” [Euzenat et al., 2002]
    • Formal approach, based on description logics
    • It does not deal with different formats
  – OntoMorph [Chalupsky, 2000]
    • Lisp-based tool to create translators more easily

• Why do we need better approaches?
  – Languages/tools evolve and new ones are created
    • RDF, RDF Schema, OIL, DAML+OIL, OWL
  – Translation decisions may change
Our proposal

• What do we need?
  – Declarative specifications of translations
    • Translators are automatically created from them
  – Specifications of knowledge transformations at different levels
    • Knowledge level
    • Symbol level
      – Syntax level
      – Lexical level
    • User interface
      – Semiotic level

[Diagram showing restructuring between Ontology in tool T, Ontology in tool T', and Ontology in tool T'']
Table of contents

• The ontology translation problem
  – Translation problems
  – Current situation
  – Technology needed for transformation

• Transformation from WebODE to Protégé2000
  – WebODE vs Protégé2000 knowledge models
  – Transformation proposals
  – Layered transformations
    • Transformations at the knowledge level
    • Transformations at the symbol level
    • Transformations of user interfaces/visualization

• Future work
What is WebODE?

• Ontology engineering workbench
  – Scalable and extensible
    • Application server and database storage
    • Plugable Services
  – Multiple users support
  – Technology support for METHONTOLOGY
    • Most of the activities of the ontology lifecycle
    • Conceptualization, documentation, reasoning, import/export, evaluation, merge, etc.
  – Ontology interoperability
    • Java API
    • Translation services
  – Ontology-based applications
    • ODE-KM (Knowledge Management)
    • ODESeW (Semantic Web portals)

• WebODE homepage: http://webode.fi.upm.es/
WebODE. Knowledge model

- Concepts
  - Class attributes
  - Instance attributes
- Concept groups (disjoint concepts)
- Relations
  - Built-in relations
    - Taxonomy: subclass-of, disjoint-decomposition, exhaustive-decomposition, partition
    - Mereology: transitive-part-of, intransitive-part-of
  - Ad-hoc binary relations
  - Relation properties (symmetry, transitiveness, etc.)
- Formal axioms and rules

- Instance sets (instances of concepts and relations)

- Constants, imported terms (URL based), bibliographic references, etc.

Protégé knowledge model

- Classes and metaclasses
  - Attached template slots (which can be inherited)
  - Own slots (obtained from metaclasses)

- Slots (first-class components)
  - Facets (predefined and ad-hoc)

- Taxonomy relations
  - Subclass of
  - Subslot of

- Class instances (individuals)

- PAL constraints and queries

Some transformation proposals

- Create the KR ontology of WebODE in Protégé2000
  - Easy transformation process
  - No information lost in the process
  - The transformed ontology is not legible, usable by Protégé2000 users

- Transform the WebODE ontology to the built-in Protégé2000 knowledge model
  - Some knowledge lost in the transformation (bibliographic references, synonyms, acronyms, etc.)

- Transform the WebODE ontology to the built-in Protégé2000 knowledge model, and preserve lost knowledge
  - Knowledge is preserved in the transformation
  - The transformed ontology is legible and usable
WebODE Concepts (I)

• **Previous transformations**

  – Hidden classes in Protégé:
    • :WebODESynonym
    • :WebODEAbbreviation
    • :WebODEReference

  – Hidden metaclass in Protégé:
    • :WebODEConcept (subclass of :STANDARD-CLASS)
      – Create template slot :Synonyms and attach it to :WebODEConcept.
      – Create template slot :Abbreviations and attach it to :WebODEConcept.
      – Create template slot :References and attach it to :WebODEConcept.
Transformations at the knowledge level (II)

WebODE Concepts (II)

For each WebODE concept

<table>
<thead>
<tr>
<th>WebODE</th>
<th>Protégé-2000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concept</strong></td>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>Name (max. 200 characters)</td>
<td>:NAME</td>
</tr>
<tr>
<td>Description (max. 2000 characters)</td>
<td>:DOCUMENTATION</td>
</tr>
<tr>
<td>Synonym</td>
<td>Instance of class :WebODESynonym (hidden class)</td>
</tr>
<tr>
<td>Name</td>
<td>:NAME</td>
</tr>
<tr>
<td>Description</td>
<td>:DOCUMENTATION</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Instance of class :WebODEAbbreviation (hidden class)</td>
</tr>
<tr>
<td>Name</td>
<td>:NAME</td>
</tr>
<tr>
<td>Description</td>
<td>:DOCUMENTATION</td>
</tr>
<tr>
<td>Axiom</td>
<td>:SLOT-CONSTRAINTS</td>
</tr>
<tr>
<td>Reference</td>
<td>Instance of class :WebODEReference (hidden class)</td>
</tr>
<tr>
<td>Name</td>
<td>:NAME</td>
</tr>
<tr>
<td>Description</td>
<td>:DOCUMENTATION</td>
</tr>
<tr>
<td><strong>Instance attribute</strong></td>
<td>See table 3</td>
</tr>
<tr>
<td><strong>Class attribute</strong></td>
<td>See table 4</td>
</tr>
</tbody>
</table>
Transformations at the knowledge level (III)

WebODE Concept groups (I)

• Previous transformations
  – Class in Protégé:
    • :PAL-DISJOINT-CONSTRAINT
      – :DOCUMENTATION: “Class that represents WebODE concept groups”
      – :ROLE: Concrete
      – :DIRECT-TYPE: :STANDARD-CLASS
      – :DIRECT-SUPERCLASSES: :PAL-CONSTRAINT

  – Create template slot :groupConcepts and attach it to the class.
    » Documentation: “concepts in the group”
    » Type: Class
    » Allowed parents: :THING (that is, any class in the ontology)
    » Cardinality: (2,N)
Transformations at the knowledge level (IV)

WebODE Concept groups (II)

For each WebODE concept group, create an instance of :PAL-DISJOINT-CONSTRAINT

:PAL-NAME: the concept group name in WebODE

:PAL-DESCRIPTION: the description of the concept group in WebODE.

:PAL-STATEMENT: the following expression

\[
\forall W \forall X \forall Y \forall Z \\
(\forall (\text{groupConcepts } W X) \land \text{groupConcepts } W Y) \\
(\text{subclass-of } Z X \land X \neq Y) \\
(\text{not (subclass-of } Z Y)))
\]

:PAL-RANGE: the following expression

\[
\text{(defrange } X :\text{FRAME :STANDARD-CLASS}) \\
\text{(defrange } Y :\text{FRAME :STANDARD-CLASS}) \\
\text{(defrange } Z :\text{FRAME :STANDARD-CLASS}) \\
\text{(defrange } W :\text{FRAME PAL_DISJOINT_CONSTRAINT})
\]

:groupConcepts: the concepts in the concept group in WebODE
Transformation at the knowledge level (V)

- WebODE class and instance attributes
Transformations at the symbol level (I)

- **Syntax transformations (slot value types)**

<table>
<thead>
<tr>
<th>WebODE</th>
<th>Protégé-2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Integer</td>
<td>Integer</td>
</tr>
<tr>
<td>Cardinal</td>
<td>Integer. :SLOT-MINIMUM-VALUE set to 0, unless the minimum value of the slot is set explicitly to another different value</td>
</tr>
<tr>
<td>Float</td>
<td>Float</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>Date</td>
<td>String with SlotWidgetDate in form</td>
</tr>
<tr>
<td>Numeric Range</td>
<td>Float. :SLOT-MINIMUM-VALUE and :SLOT_MAXIMUM_VALUE have been already set</td>
</tr>
<tr>
<td>URL</td>
<td>String with SlotWidgetURL in form</td>
</tr>
<tr>
<td>Instance of Concept</td>
<td>Instance</td>
</tr>
<tr>
<td>--</td>
<td>Class</td>
</tr>
<tr>
<td>--</td>
<td>Any</td>
</tr>
<tr>
<td>--</td>
<td>Symbol</td>
</tr>
</tbody>
</table>
Transformations at the symbol level (II)

• Lexical transformations
  – Identifiers
    • WebODE allows using the same name for a concept and an instance.
    • Protégé automatically renames one of it
  – Maximum length for fields
  – Characters that are not allowed
  – etc.
Transformations at the user interface

• Customization of Protégé2000 forms
  – Removing WebODE-related slot widgets
  – Specific slot widgets for dates and URLs
  – Hiding classes from project
  – Activate PAL Constraint tab

• Import of a project
  – WebODEAdditions.pprj
  – It contains the WebODE additional components identified in “previous transformations”
Table of contents

• The ontology translation problem
  – Translation problems
  – Current situation
  – Technology needed for transformation

• Transformation from WebODE to Protégé2000
  – WebODE vs Protégé2000 knowledge models
  – Transformation proposals
  – Layered transformations
    • Transformations at the knowledge level
    • Transformations at the symbol level
    • Transformations of user interfaces/visualization

• Future work
Future work

• Transformations from Protégé2000 to WebODE
  – Get back the knowledge preserved in transformation WebODE-Protégé
  – Maintain knowledge that could be lost in the transformation

• Declarative specifications of translators
  – Step 2:
    • Create a formal language from current tables
    • More automatization in translator creation
  – Step 3:
    • Ontology mapping between knowledge models
      – Based on existing approaches for information integration
      – Procedural attachments for “executing” mappings
Achieving interoperability between WebODE and Protégé2000

Oscar Corcho
ocorcho@fi.upm.es

Ontology Group
Laboratorio de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo sn,
28660 Boadilla del Monte, Madrid, Spain