Ontology Modeling 2.0: Next Steps

Pascal Hitzler
Data Semantics Laboratory (DaSe Lab)
Data Science and Security Cluster (DSSC)
Wright State University
http://www.pascal-hitzler.de
A module is a part of an ontology which corresponds to a conceptual part of the domain which can be summarized under one (key) notion within the topic domain.

E.g., in a chess games ontology there may be an “opening” module, and an “tournament” module.

E.g., in a travel planning support ontology, there may be a “hotel” module, and a “trajectory” module.

We understand ontologies to be composed of modules.

We understand modules to be obtained by instantiating, joining and modifying patterns.
Divide and Conquer

Problem Decomposition

- identify modules
- identify patterns

Ontology Assembly

- instantiate patterns
- assemble modules
- assemble ontology
Decomposition

Module identification:

- Main notions relevant to data and modeling problem.
- Driven by competency questions, use case descriptions, and inspection of available data sources.
- Best done on a whiteboard.
Pattern identification:

• For each module, which pattern(s) reflect the nature of this module?
• Driven by competency questions, use case descriptions, inspection of available data sources, identified modules, repository of available patterns.
• Best done on a whiteboard(?)
• E.g. (chess),
  “moves” → list pattern
  “players” → agent role pattern
  “tournament” → event pattern
Assembly

Pattern instantiation:

- Identified (generic) patterns to be used as templates:
  - Instantiate (change class/property names), and import to local namespace
  - Modify (adapt to the specific need)
- Provenance information (which pattern was used and how) should be kept.

- Strong tool support needed. Could be graphical, but needs to work on axiom level.
  (draft solution: Karl Hammar’s XD Protégé plug-in)
Player as AgentRole

owl:Thing \(\xrightarrow{\text{providesAgentRole}}\) AgentRole \(\xrightarrow{\text{performedBy}}\) Agent

AgentRole \(\xrightarrow{\text{startsAtTime, endsAtTime}}\) TimeInstant

BlackPlayerRole \(\xrightarrow{\text{rdfs:subClassOf}}\) AgentRole

WhitePlayerRole \(\xrightarrow{\text{rdfs:subClassOf}}\) AgentRole

ChessGame \(\xrightarrow{\text{providesAgentRole}}\) AgentRole \(\xrightarrow{\text{performedBy}}\) Agent
XD plug-in

[Hammar]
XD plug-in

[Hammar]
Module assembly:

- From instantiated patterns.
- Plus additional modifications/additions as needed.
- Modules may contain other modules.
- Provenance information (which pattern was used and how) should be kept.
- Modules should be identifiable from the OWL file.

- Strong tool support needed. Could be graphical, but needs to work on axiom level.
  (draft solution: Karl Hammar’s XD Protégé plug-in)
 XD plug-in

[Hammar]
Graphical mock-up

ChessTournament

ChessOpening

ChessOpening

ChessGame

ChessGame

ChessGameResult

ChessGame

WhitePlayerRole

AgentRole

BlackPlayerRole

Agent
Ontology assembly:

- From developed modules.
- Plus additional modifications/additions as needed.
- Modules should be identifiable from the OWL file.
- Used patterns should be identifiable from the OWL file.

- Strong tool support needed. Could be graphical, but needs to work on axiom level.
Axiomatization support

- **OWLAx Protégé plug-in [ISWC2016 demo]**
  - Start with schema diagram.
  - Quick and easy addition of most common axioms using check-box selection.

- **ROWL Protégé plug-in [ISWC2016 demo]**
  - Rule-based interface for adding complex OWL axioms.
  - Evaluated [ESWC2017 paper] showing that it improves modeling efficiency.
Axioms – Systematically

1. $A \cap B \subseteq \bot$
2. $\exists R. T \subseteq A$
3. $\exists R. B \subseteq A$
4. $T \subseteq \forall R. B$
5. $A \subseteq \forall R. B$
6. $A \subseteq R.B$
7. $B \subseteq R^-.A$
8. $T \subseteq \leq 1 R.T$
9. $T \subseteq \leq 1 R.B$
10. $A \subseteq \leq 1 R.T$
11. $A \subseteq \leq 1 R.B$
12. $T \subseteq \leq 1 R^- . T$
13. $T \subseteq \leq 1 R^- . A$
14. $B \subseteq \leq 1 R^- . T$
15. $B \subseteq \leq 1 R^- . A$

1. $A$ DisjointWith $B$
2. $R$ some $owl:Thing$ SubClassOf $A$
3. $R$ some $B$ SubClassOf $A$
4. $owl:Thing$ SubClassOf $R$ only $B$
5. $A$ SubClassOf $R$ only $B$
6. $A$ SubClassOf $R$ some $B$
7. $B$ SubClassOf $inverse$ $R$ some $A$
8. $owl:Thing$ SubClassOf $R$ max 1 $owl:Thing$
9. $owl:Thing$ SubClassOf $R$ max 1 $B$
10. $A$ SubClassOf $R$ max 1 $owl:Thing$
11. $A$ SubClassOf $R$ max 1 $B$
12. $owl:Thing$ SubClassOf $inverse$ $R$ max 1 $owl:Thing$
13. $owl:Thing$ SubClassOf $inverse$ $R$ max 1 $A$
14. $B$ SubClassOf $inverse$ $R$ max 1 $owl:Thing$
15. $B$ SubClassOf $inverse$ $R$ max 1 $A$

(disjointness)
(domain)
(scoped domain)
(range)
(scoped range)
(existing)
(inverse existential)
(functionality)
(qualified functionality)
(scoped functionality)
(qualified scoped functionality)
(inverse functionality)
(inverse qualified functionality)
(inverse scoped functionality)
(inverse qualified scoped functionality)
OWL Ax Protégé plug-in

In: Proc. ISWC 2016 poster & demos
http://dase.cs.wright.edu/content/ontology-axiomatization-support
ROWL Protégé plug-in

http://dase.cs.wright.edu/content/rowl
The hypotheses for time and for correctness (hard questions) were confirmed. For correctness (medium questions) the hypothesis was rejected.

<table>
<thead>
<tr>
<th>category</th>
<th>time</th>
<th>clicks</th>
<th>correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>easy</td>
<td>significant ($p &lt; 0.05$)</td>
<td>not significant</td>
<td>not significant</td>
</tr>
<tr>
<td>medium</td>
<td>significant ($p &lt; 0.01$)</td>
<td>significant ($p &lt; 0.05$)</td>
<td>not significant</td>
</tr>
<tr>
<td>hard</td>
<td>significant ($p &lt; 0.05$)</td>
<td>not significant</td>
<td>significant ($p &lt; 0.01$)</td>
</tr>
</tbody>
</table>

It appears that medium modeling problems (with some role restrictions) can be done correctly with the standard Protégé interface by this type of user, although more time is needed than when using ROWLTab.

It appears that hard problems (requiring rolification) cannot really be solved using the standard Protégé interface, and the unsuccessful solution attempts in addition require more time.
Keeping track

- Keeping track of modules and patterns within an ontology, and of their origins.

- Simple proposed solution: OPLa (Ontology Pattern Language):


  See the presentation on this later today.
What is missing?

- High-quality (well-documented) sets of ODPs. Perhaps we need to get away from loose collections of ODPs, and rather start talking (and developing) "ODP suites" which consist of uniformly modeled ODPs.

- Instantiation and composition tools (like Hammar's prototype, but on steroids). They require an ODP language, and graphical support.

- Good, well-written (textbook-style) tutorials tailored to the tools and suites.
Thanks!
References


References


Adila Krisnadhi, Ontology Pattern-Based Data Integration. Dissertation, Department of Computer Science and Engineering, Wright State University, 2015.


Karl Hammar, Content Ontology Design Patterns: Qualities, Methods, and Tools. Dissertation. Linköping University, Sweden 2017