OBOWLMorph: Starting Ontology Development from PURO Background Models

Marek Dudáš, Tomáš Hanzal, Vojtěch Svátek, and Ondřej Zamazal

Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic,
{marek.dudas|tomas.hanzal|svatek|ondrej.zamazal}@vse.cz

Abstract. We propose adding two additional steps to OWL ontology development and offer tools supporting it. A so-called PURO background model of an example situation to be covered by the ontology is first created, then a seed of the ontology is generated automatically from it, allowing users to choose suitable modeling style and import the ontology seed into a common ontology editor where it can be finalized. Using PURO as intermediary model should enable better collaboration, documentation and early detection of design problems. The paper focuses on OBOWLMorph: a tool for ontology generation from a PURO model.

1 Introduction

In the semantic web realms, the prevailing practice of formalizing ontologies is writing them, from the onset, in OWL, merely starting from textual specifications and informal charts. The advantages of OWL as uniform representation of ontologies throughout all 'formal' phases of their development lifecycle are its thorough standardization, solid support by authoring tools, and powerful reasoning abilities allowing formal consistency checking of the models. On the other hand, the direct transition from informal specifications to OWL puts quite high demands on ontology engineers. In software engineering, UML models are often created before the actual coding. Database designers use even two levels of intermediate models: conceptual and logical. The common benefits of such intermediate models are better collaboration, documentation and early detection of architectural or logical problems. Ontology engineers directly defining OWL entities based on informal specifications have to deal with two problems at the same time: (a) "What are the entities and relations inherently described in the specification?" and (b) "How to represent them with OWL constructs?" Moreover, the latter question often has several possible answers – choosing different *OWL modeling styles*. Therefore we investigate, and attempt to offer tools for, a more stepwise approach to ontological engineering where a 'seed' of the ontology is first drafted and exemplified in a less constrained language (called PURO [10]) and a basis of the ontology is then generated from it through pattern-based PURO-to-OWL transformation. This allows focusing on question (a) in the first step without having to solve question (b), which is dealt with in the second step.

The generated basis of an ontology can then be finalized using common ontology editors. While such a practice is not new per se (since some earlier methodologies [6] proposed to create the first formalization in first-order predicate calculus), the crucial point is the use of PURO as language with structure similar to OWL, giving way to the automatic transformation. Simple PURO-to-OWL transformation has recently been integrated [4] as additional feature into PURO Modeler (our graphical PURO model authoring tool prototype); it allows to display alternative OWL representations of uncomplicated PURO models, thus serving as an auxiliary tool for ontology developers, with educative role for novices. In this paper, in contrast, we present customizable PURO-to-OWL transformation functionality implemented in a dedicated tool, OBOWLMorph, which has the potential to play a more central role in the ontology development workflow.

2 PURO Language and OWL Modeling Styles

OWL ontology developer can often choose different combinations of language constructs to model the same situation. The choice might be driven by the intended usage of the ontology: web markup vocabularies often favor 'feature' assignment to entities through data properties, while linked data vocabularies prefer object properties for this purpose; reasoning-enabled ontologies, in turn, express 'features' as classes. We call sets of such choices *modeling styles*.[1]

PURO is an ontological modeling language recently drafted as common interlingua for different modeling styles in OWL. A model built in PURO is denoted as *ontological background model* (OBM). For example, the fact that a concrete book is a 'paperback' can be expressed, in OWL, using an object property assertion, a data property assertion or a class instantiation. In PURO it is always the last option, called 'B-instantiation' ('background-instantiation'), since 'in the background', individual paperback books and the notion of 'paperback' are interrelated via sound set membership.

PURO inventory is very similar to that of OWL, assuring easy understandability by OWL-bred engineers. It is based on two distinctions: between particulars and universals and between relationships and objects (hence the PURO acronym). There are six basic entity types: B-object (particular object), B-type (type of object/type), B-relationship (particular relationship), B-relation (type of relationship), B-valuation (particular assertion of quantitative value) and B-attribute (type of valuation). An OBM consists of named entities of these types, plus of subTypeOf and instanceOf relationships. Obviously, the 'object-type-relation' triad of PURO corresponds to the 'individual-class-property' triad of OWL, except that 1) PURO does not limit the arity of relations and allows higher-order classes, and, consequently, 2) enables abstracting from modeling style choices that are enforced by these limitations in OWL. For example the fact that a book is published by a publisher in a certain year normally requires reification to a new entity (e.g., of 'PublishingEvent' type) in OWL; in PURO a

[1] In agreement with [3] where OWL feature modeling styles are analysed.

ternary relationship suffices for this purpose. Every PURO OBM describes a concrete sample situation. *Instances* ('particulars') play central role in the model, helping designers to avoid speculating about abstract categories for which there would be no concrete data available and making them focus on sample situations to be covered. Particulars also glue different type-level entities ('universals') together in a contiguous model. A PURO OBM thus does not only map on OWL ontologies but also on samples of their respective fact bases (Aboxes).

The research of modeling styles is still in its infancy. To test OBOWLMorph, we implemented five ad-hoc modeling styles. In *Data property style*, data properties are used whenever possible. *Object property style* prefers object properties, even to model subTypeOf relationships and B-attributes. *Object-prop and subclassing style* is similar to *Object property style*, but subTypeOf is represented by *rdfs:subClassOf* and B-attributes are modeled as data properties. In *Reification style*, even binary B-relationships are reified into classes and pairs of object properties, otherwise it is same as *Object-prop and subclassing style*. So is *Class membership style* with the exception that binary B-relationships are turned into classes of subjects of the B-relationship having the value of the object.

3 Related research

The most similar to our approach is OntoUML [2]: a conceptual modeling language based on UML and grounded in Universal Foundational Ontology (UFO). OLED, the graphical editor for OntoUML, allows to transform it into OWL fragments. The transformation is hard-coded and each OntoUML element has its single OWL counterpart. The users can however select for each OntoUML element whether it may change in time or is 'read-only', and the choice is reflected in the transformation; such functionality is planned to be added to our framework as well. Bauman [1] implemented XSLT transformation of conceptual models into XML Schema, while OWL as target is only mentioned as possible future work. The user can choose a sort of modeling style, e.g., whether to transform a concept to an XML attribute or child-element. To allow reusing existing ER diagrams, Fahad [5] designed their rule-based transformation to OWL ontologies. The framework is however not intended as a general ontology development alternative. For transformation between different types of models such as UML, XML Schema or OWL, Kensche et al. [8] suggested to employ a generic metamodel (GeRoMe), as an abstraction of particular metamodels. In order to uniformly capture specific properties of models of different types, elements of GeRoMe are decorated with a set of role objects (e.g., a role attribute is mapped to a column in a relational schema and to a data property in OWL DL). Native models can be imported/exported into/from GeRoMe. In all mentioned OWL generation methods, the input model is created at the level of types. In our approach, in contrast, the input model is created as an example situation at the instance level. Finally, since the PURO language has also been proposed as means of formally testing the conceptual coherence of ontologies [10], it can be compared to OntoClean [7]; it differs in the 'meta-properties' assigned to entities.

4 OBOWLMorph Implementation and Example of Usage

The generation of OWL from an OBM is done with SPARQL. To allow that, the OBM is first serialized into RDF using a simple 'PURO vocabulary'. The serialization also includes information about the desired modeling style (gathered from the user) in the form of annotations of serialized PURO entities. The serialized OBM is then transformed to OWL with a set of SPARQL UPDATE[2] queries. Using SPARQL allows the transformation rules to be easily altered and extended by the semantic web community. So far we have defined 12 SPARQL patterns,[3] covering the most common combinations of PURO entities and their OWL representations in different modeling styles. The WHERE part of each query represents a pattern of an OBM fragment, including the modeling style annotations. The INSERT part describes a corresponding OWL fragment. The resulting OWL fragment is inserted into a separate RDF graph. All SPARQL queries are applied automatically in a sequence and the resulting OWL ontology 'seed' is then extracted from the graph. OBOWLMorph is implemented as a web application[4] and integrated with PURO Modeler, a visual OBM editor connected to same DB as OBOWLMorph.

The OBOWLMorph interface consists of two windows: one displays the loaded OBM, while the other shows the OWL ontology seed generated from the OBM and visualized in WebVOWL [9]. The user can choose a different target OWL modeling style for each OBM entity: s/he simply clicks on an entity, selects from available modeling styles shown in a pop-up window and clicks the 'update' button to see the change in the OWL ontology seed. A default modeling style is used for the entities unaffected by the user.

Use-case Scenario Example: Consider that during the development of a food ontology, a verbal example is gathered: *A boiled egg is a dish of size 100g, containing 12g of fat, 2g of carbohydrates and 800 kJ of energy. Its ingredient is one egg.* The knowledge engineer creates its OBM in PURO Modeler[5] as shown in Fig. 1. S/he may share and discuss it with other developers and check whether all concepts from the example are modeled and correctly labeled. Then s/he proceeds to OWL modeling. Keeping the default modeling style settings (*Object-prop and subclassing style* set for all entities), OBOWLMorph produces the result shown in Figure 2. The engineer now considers the intended usage of the ontology and decides to, e.g., simplify it by changing the modeling style of the "egg" entity to *Datatype*. On the other hand, the "energy" value needs to be modeled as instance, because the engineer decided to allow adding the unit specification to avoid confusion between kJ and calories. Therefore, s/he sets the *Reification* modeling style for the "800 kJ" entity. After updating, the ontology seed looks as shown in Figure 3. When the engineer is satisfied with the result, s/he may download the OWL ontology seed, import it to an ontology editor such as Protégé, and continue working on it.

[2] Used instead of CONSTRUCT for implementation-specific reasons.
[3] Available at `http://lod2-dev.vse.cz/puromodeler-v2/OBOWLMorph/patterns/`
[4] `http://lod2-dev.vse.cz/puromodeler-v2/OBOWLMorph/`
[5] Following the guidelines available at `http://bit.ly/1MFr8Lm` (in development)

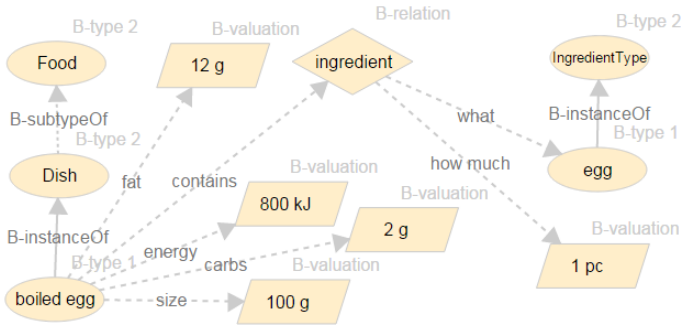

Fig. 1. OBM of a boiled egg as a dish with one ingredient and nutrition info.

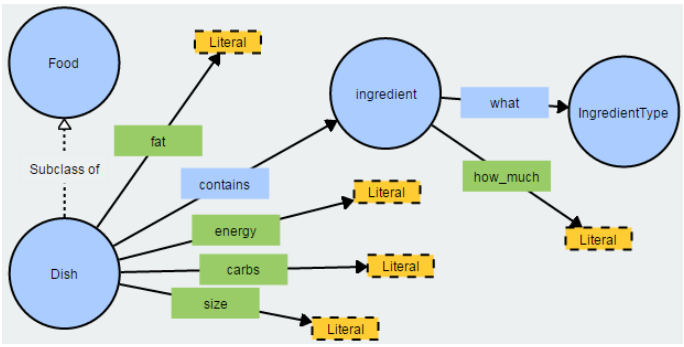

Fig. 2. OWL ontology seed generated from the OBM using default modeling style.

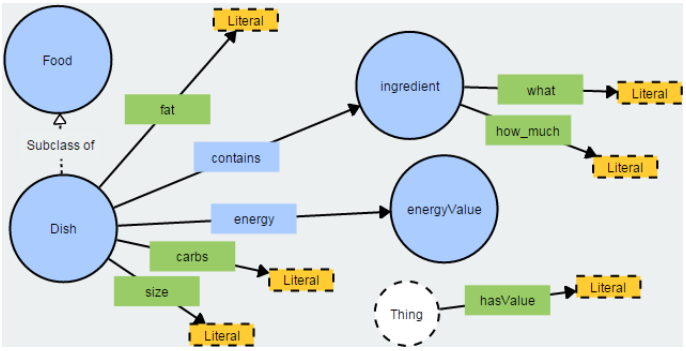

Fig. 3. OWL ontology seed with modeling style on some OBM entities altered.

5 Conclusions and Future Work

We proposed introducing two more steps into ontology design, to some extent analogical to logical data models or UML models in database and software engineering: the ontology designer first creates a PURO ontological background model, from which a seed of the desired OWL ontology is generated automatically. The approach is supported by an experimentally implementated web application, allowing to generate an ontology 'seed' in various modeling styles.

The current inventory of modeling styles is a proof-of-concept one; however, further styles can be implemented rapidly, being mere SPARQL UPDATE queries. Future work will also include reuse of entities (with fitting style) from existing vocabularies in addition to coining new ones, during transformation, as well as exploitation of entity naming conventions when using sophisticated patterns, e.g., those including reification. Thorough evaluation by user assessment and comparison to gold-standard ontologies is also foreseen.

The research is supported by UEP IGA F4/90/2015 and by long-term institutional support of research activities by Faculty of Informatics and Statistics, Univ. of Economics, Prague. Ondřej Zamazal is supported by CSF 14-14076P.

References

1. Bauman, B.T.: Prying apart semantics and implementation: Generating xml schemata directly from ontologically sound conceptual models. In: Proceedings of Balisage: The Markup Conference 2009. `http://www.balisage.net/Proceedings/vol3/print/Bauman01/BalisageVol3-Bauman01.html`
2. Benevides, A.B., Guizzardi, G.: A model-based tool for conceptual modeling and domain ontology engineering in OntoUML. In: Enterprise Information Systems, pp. 528–538. Springer (2009)
3. Dermeval, D., Tenório, T., Bittencourt, I.I., Silva, A., Isotani, S., Ribeiro, M.: Ontology-based feature modeling: An empirical study in changing scenarios. Expert Systems with Applications 42(11), 4950–4964 (2015)
4. Dudáš, M., Hanzal, T., Svátek, V., Zamazal, O.: OBM2OWL patterns: Spotlight on OWL modeling versatility. In: W'shop on Ontology and Sem. Web Patterns (WOP) at ISWC (2015), `http://lod2-dev.vse.cz/puromodeler/purom-wop15.pdf`
5. Fahad, M.: Er2owl: Generating owl ontology from er diagram. In: Intelligent Information Processing IV, pp. 28–37. Springer (2008)
6. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer (2004)
7. Guarino, N., Welty, C.A.: An overview of ontoclean. In: Handbook on ontologies, pp. 201–220. Springer (2009)
8. Kensche, D., Quix, C., Chatti, M.A., Jarke, M.: Gerome: A generic role based metamodel for model management. In: Journal on data semantics VIII, pp. 82–117. Springer (2007)
9. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: user-oriented visualization of ontologies. In: Knowledge Engineering and Knowledge Management, pp. 266–281. Springer (2014)
10. Svátek, V., Homola, M., Kluka, J., Vacura, M.: Metamodeling-based coherence checking of OWL vocabulary background models. In: OWLED (2013)