Kickstarting OntoUML Modeling from PURO Instance-Level Examples

Marek Dudáš[1], Tomáš Morkus[1], Vojtěch Svátek[1], Tiago Prince Sales[2], and Giancarlo Guizzardi[2]

[1] Prague University of Economics and Business, Czech Republic
{marek.dudas|mort00|svatek}@vse.cz
[2] Free University of Bozen-Bolzano, Italy
{gguizzardi,tiago.princesales}@unibz.it

Abstract. OntoUML is a modeling language based on an underlying foundational ontology (UFO). Over the years, it has gained increasing attention in the area of ontology-driven conceptual modeling, becoming one of the most used languages in that field. In this paper, we report on a preliminary prototype that kickstarts the creation of pre-conceptual models in OntoUML from instance-level examples in PURO, a graphical modeling language originally developed for analogously kickstarting OWL ontology skeletons.

Keywords: Ontology-Driven Conceptual Modeling · Modeling support for Novice Users · OntoUML · PURO

1 Introduction

In recent years, there has been a growing interest in using foundational ontologies for building reference conceptual models, as well as domain and core ontologies. In particular, in the context of conceptual modeling, there is a tradition (going back to the mid 80's) of evaluating and redesigning modeling languages according to formal ontological theories. In this context, a modeling language that gained increasing attention is OntoUML and its underlying foundational ontology UFO (Unified Foundation Ontology) [2, 4, 3]. In fact, a recent study shows that UFO and OntoUML are, respectively, among the most used foundational ontology and modeling language in ontology-driven conceptual modeling [5].

Despite its increasing adoption, given its ontological expressivity, OntoUML novice users can benefit from different types of support smoothing their intial learning curve. These include computational supporting tools [2] (which include features such as model editing, verification, verbalization, and validation via visual simulation), as well as modern construction approaches that are pattern-based and dialogue-based [4], or that explore the canvas metaphor [3].

In this spirit, in the current paper, we report on a tool that aims at supporting the user in building preliminary pre-conceptual models in OntoUML from instance-level examples created in the PURO language [1]. The idea of using

instances for building or enriching (in a bottom-up manner) the understanding of general concepts is found in cognitive science (e.g., prototype-based and examplar-based view of concepts), in the tradition of model population in information modeling (e.g., in the ORM tradition) as well as in model-finding formal approaches [2].

The remainder of the paper is organized as follows. In section 2, we briefly describe the PURO and OntoUML languages. In section 3, we discuss the proposed prototype. This proof-of-concept prototype, via a step-by-step dialogue with the modeler, supports the construction of an OntoUML model from a PURO model describing the setting primarily at the instance level (this is the subject of the demo).

2 Background: PURO and OntoUML

2.1 PURO

PURO was primarily developed as a graphical[3] language for kick-starting the development of *OWL* ontologies [1]. It possesses a very simple inventory of modeling primitives (objects and their types, relationships, and quantitatively valuated attributes), which can be however assembled in an less constrained way than their counterparts (individuals, classes, and object/datatype properties) in OWL; in particular, multi-level types and relations of arbitrary arity are allowed. The designers can thus express their conceptualization with fewer artificial 'tweaks' such as reification or meta-modeling/punning. A PURO model is however, normally, not just a schema, but an important role is played by example *individuals* that tie the graph together. Through pattern-based transformation, the same PURO model can give rise to alternative OWL encodings adapted to different needs (e.g., with preference for class-level modeling, meta-modeling by instances, or by literal values); obviously, these are not complete ontologies but just skeletons that can be further extended in an OWL editing environment.

However, as demonstrated in this paper, the potential of interactive graphical modeling of instance (and linked schema) elements, as provided by the prototype PURO editing tool (PURO Modeler), is not restricted to scenarios having an OWL ontology as a target notation. Its distinct usage might be to kick-start the development of ontological conceptual models in a language such as *OntoUML*. OntoUML's modeling power (including the sheer number of its primitives grounded in the UFO foundational ontology) is much higher than that of PURO; this however can make it difficult for novice users to resolve all modeling decisions in one shot. Similarly to the PURO+OWL scenario, the proposed PURO+OntoUML synergy then consists in the initial drafting of an instance-level example in PURO, followed by its interactive transformation to OntoUML as the target language, and, finally, refinement in that language.

In order to illustrate the instance-based modeling capabilities of PURO, let us take the example model of Figure 1A. Suppose we start with a number of

[3] Its first-order formalization has been recently set up, but not yet published.

facts representing a particular trip, namely, the famous second voyage of the HMS Beagle (a PURO B-relation). This entity is described as being connected to the date of 1831 (via a B-attribute) but it also connects a number of other entities: the HMS Beagle itself, Plymouth, Salvador, Charles Darwin, and Robert FitzRoy. Now, suppose the user is able to tag these entities in the following manner: HMS Beagle is a ship, Darwin and FitzRoy are people, and Salvador and Plymouth are harbors.

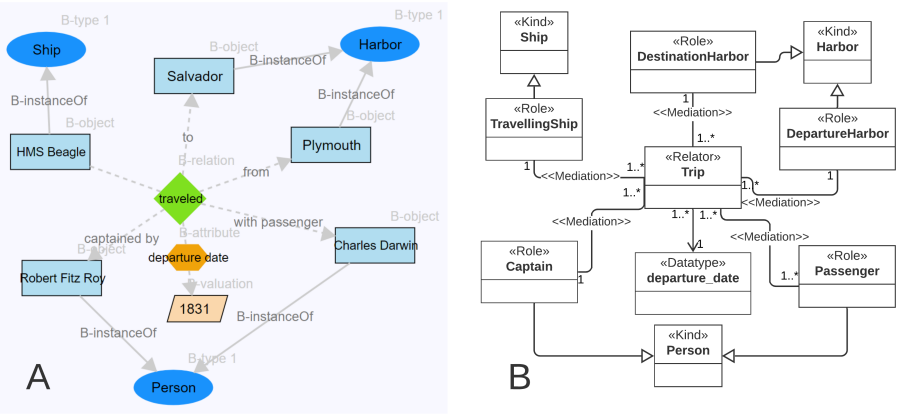

Fig. 1. Example of a PURO model (A) and the resulting OntoUML model (B)

2.2 OntoUML

In the sequel, we briefly explain a much reduced subset of the ontological distinctions in OntoUML by employing the model of Figure 1B. For more details on this language and its foundations, one can refer to [3, 4, 2].

This model (generated by the transformation discussed below) represents *endurants* (roughly objects), i.e., entities that can qualitatively change while maintaining their numerical identity. Central to this domain we will have a number of object *Kinds*, i.e., the genuine fundamental types of objects that exist in this domain. Kinds represent *essential* properties of objects (they are also termed rigid or static types). Typical examples of kinds include Person and Ship (see Figure 1B; stereotypes reflect the correspondence between the UML profile and UFO). There are, however, types that represent contingent or accidental properties of objects (termed anti-rigid types). These include *Roles*, which represent contingent properties that entities have in a relational context (e.g., 'being a husband' is to bear a number of commitments and claims towards a spouse in the scope of a marital relationship; 'being a student' is to bear a number of properties in the scope of an enrollment relationship with an educational institution.). These relational contexts are termed *Relators*. *Relators* represent clusters of relational

properties that "hang together" by a nexus (provided by a relator kind). They are existentially dependent entities (e.g., the marriage between John and Mary can only exist if John and Mary exist) that bind together entities (their relata) by the so-called *mediation* relations - a particular type of existential dependence relation. Examples of relators include marriages, enrollments, presidential mandates, citizenships, but also trips (see Figure 1B). Objects participate in relators playing roles. For instance, people play the role of spouse in a marriage relationship; a person plays the role of president in a presidential mandate; a harbor plays the role of a departure harbor and/or destination harbor in the scope of a trip (see Figure 1B). Finally, all types of endurants can be characterized by intrinsic properties (*qualities*) that can take their value in certain *quality structures*. UFO quality structures are represented in OntoUML via the constructs of datatypes (see the departure date in Figure 1B).

3 The Prototype Tool

The users can first create an example model in PURO Modeler[4] and then the *Build OntoUML* button takes them to the transformation app. The transformation is dialogue-based.[5] The app asks users a series of questions about the nature of modeled PURO entities in terms of OntoUML stereotypes. The app crawls the graph representation of the source PURO model, starting from relationship nodes, and fires the questions based on a set of rules. In each question, users are given applicable choices of OntoUML stereotypes for representing the given PURO entity in OntoUML. The resulting OntoUML skeleton is gradually displayed next to the source PURO model. The user can then download the result, load it into an OntoUML modeling tool and finalize it.

The authors (MD, TM and VS) have been partially supported by CSF 18-23964S.

References

1. Dudáš, M., Hanzal, T., Svátek, V., Zamazal, O.: OBOWLMorph: Starting ontology development from PURO background models. In: 12th OWLED@ISWC (2015)
2. Guerson, J. et al.: OntoUML lightweight editor: a model-based environment to build, evaluate and implement reference ontologies. In: IEEE 19th EDOCW (2015)
3. Guizzardi, G., Sales, T.: "As simple as possible but not simpler": Towards an ontology model canvas. In: JOWO 2017. vol. CEUR 2050 (2017)
4. Guizzardi, G. et al.: Design patterns and inductive modeling rules to support the construction of ontologically well-founded conceptual models in OntoUML. In: ODISE@CAISE 2011. pp. 402–413. Springer
5. Verdonck, M., Gailly, F.: Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. In: 35th International Conference on Conceptual Modeling (ER). pp. 83–97. Springer (2016)

[4] Available at http://protegeserver.cz/purom5/
[5] Demo available at https://tinyurl.com/puro-ontouml

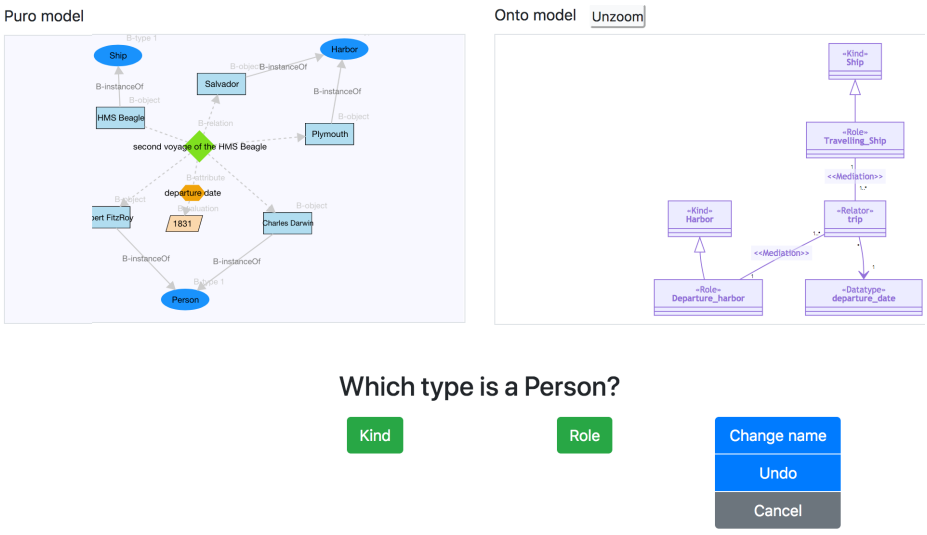

Fig. 2. The OntoUML Model Kickstart Prototype - a screenshot of the web application showing an intermediate step of a dialog-based transformation from PURO (on the left) to OntoUML (right).