# Module merging in PURO visual modeling

Marek Dudáš<sup>1,\*,†</sup>, Daniel Bedrníček<sup>1,†</sup> and Vojtěch Svátek<sup>1,†</sup>

<sup>1</sup>Department of Information and Knowledge Engineering, Prague University of Economics and Business, Czech Republic

#### Abstract

PURO is a primarily graphical language for capturing ontological conceptualizations at the level of interconnected example entities and their types. The PURO Modeler tool allows users to create the PURO models and transform them to skeletons of models in other languages: OWL and OntoUML. In real-world scenarios, a single PURO model quickly becomes too large to be managed due to its graphical nature. We demonstrate a solution consisting of modularization of the models and a semiautomated way of merging the modules before they are transformed into OWL or OntoUML.

#### **Keywords**

ontology, PURO, modularity, alignment, merging

## 1. Introduction

Modularization is an obvious approach to handling large knowledge bases and ontologies. While the size of ontologies represents a challenge for multiple tasks, including reasoning, it is probably most harmful when it comes to human interaction. Therefore, ontological modeling languages that emphasize visual browsing and editing by human users are particularly sensitive in this respect.

An example of such a language is PURO [1], which can serve as an easy and flexible prototyping language for other languages that are richer, more operational, but require modeling or encoding decisions that need not be made in the early phase of domain capturing. PURO models can then be semi-automatically converted to model skeletons in either OWL [2] or OntoUML [3], which can be then further extended in dedicated environments (such as Protégé for OWL or Menthor Tool for OntoUML). PURO follows an example-based approach to modeling; therefore, even if the amount of ontological entities (Tbox) in a PURO model is lower than in fully-blown ontologies, the model's size is, on the other hand, increased through the presence of example instances and their relationships.

The original tool for authoring PURO models, PURO Modeler, only allowed users to create monolithic models, as no module-handling support was available. This paper presents a new addition to PURO Modeler, which allows merging independently developed

International Workshop on Knowledge Graph Generation from Text (TEXT2KG 2022) and Modular Knowledge, May 29, 2022, Hersonissos, Greece

 $<sup>^{\</sup>ast}\mbox{Corresponding}$  author.

<sup>&</sup>lt;sup>†</sup>These authors contributed equally.

<sup>🛆</sup> marek.dudas@vse.cz (M. Dudáš); bedd01@vse.cz (D. Bedrníček); svatek@vse.cz (V. Svátek)

D 0000-0002-9388-8322 (M. Dudáš); 0000-0002-2256-2982 (V. Svátek)

<sup>© 2022</sup> Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

models (modules), thus enabling drafting larger skeletons of ontologies within a single project.

# 2. PURO and PURO Modeler

PURO was primarily developed as a graphical 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 a 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. Moreover, a PURO model is not just a schema. An important role is played by example individuals that tie the model together. Through pattern-based transformation, the same PURO model can give rise to alternative OWL encodings adapted to different needs (e.g., with a 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.

An alternative use case identified for PURO is to kick-start the development of ontological conceptual models in a semantically rich language such as OntoUML [3]. 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 PURO+OntoUML synergy consists of 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 [4].

Both use cases are supported by the prototype PURO editing tool, PURO Modeler.<sup>1</sup> However, a weakness of the original PURO Modeler was the monolithic nature of the developed PURO models, which often became unmanageable before reaching the size of a useful ontology skeleton.

## 3. PURO model alignment techniques

The manageable size of PURO models can be achieved through their modularization. Since each model is meant to represent an example real-world situation, it is natural to confine each model to some small domain area, for example, to create one module (a partial model) per competency question; e.g., one module may cover the notion of a person and its relationships, and another one may cover the topic of organizations. Since the goal is to create a single, coherent ontology skeleton, the modules need to be merged into a single PURO model before the transformation. Some automation and visualization techniques (since we still need to show the whole merged model to enable

<sup>&</sup>lt;sup>1</sup>Available at http://protegeserver.cz/purom5/

checking for errors and manual editing) can be employed to make the merging easier. Namely, we can match identical entities present in different modules<sup>2</sup> automatically in a way similar to ontology matching [5]. Furthermore, some entities in the merged model can be identified as redundant and omitted. Finally, to make the visualization of the result readable, we can still employ modularization by grouping parts of the model and enabling expanding/collapsing of the modules as needed.

### 3.1. Matching entities and merging different modules

The heuristic matching process is applied to 2 modules at a time (called source and target, where the target module stores the result of the merging) in the following way: (1) A set P of pairs of B-types, and of B-objects, respectively, is found, such that either the string similarity of their names is above a user-controlled threshold or they are synonyms in WordNet [6]. (2) P is shown to the user, who can modify P and add further pairs to it. (3) Pairs of entities (s, t) having identical types and names are found, such that there exists a pair  $(a, b) \in P$  such that s is linked to a and t is linked to b; each such pair (s, t) is added to P.<sup>3</sup> (4) Entities not present in P are copied from the source module to the target module. (5) Links between entities are copied from the source module to the target module, according to the matched pairs from P.

### 3.2. Simplifying the merged model for clearer visualization

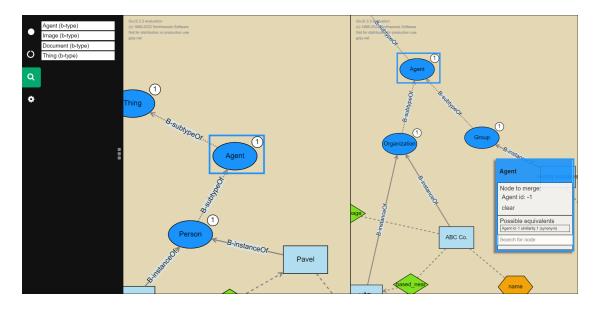
Different modules might include example instance-level entities of the same type involved in different relationships. For example, we can have a module with a person instance John having hobbies, and another module with Bob being employed. To decrease the number of visible nodes and links, we can merge John and Bob into a single entity named, e.g., 'some person,' involved both in the 'hasHobby' and 'isEmployed' relationships. Such merging can be optionally turned on by the user and is very straightforward: each group of B-objects that are instances of the same B-type is merged into a single placeholder B-object that bears the name of the parent B-type.

#### 3.3. Grouping in visualization

Any part of the model can be selected and grouped in the visualization. The group can then be collapsed into a single node, with relationships to the nodes inside the group preserved in the visualization.

 $<sup>^2\</sup>mathrm{E.g.},$  the person module will probably include an organization entity that can be matched to the organization module.

<sup>&</sup>lt;sup>3</sup>Since entities other than B-types and B-objects are always dependent on some B-type or B-object, we can only match them when they are linked to an already matched pair of B-types or B-objects.



**Figure 1:** Merging two modules that include the Agent entity. (Number labels on the nodes show B-type hierarchy level redundant in the context of this paper.)

## 4. Implementation

As a proof of concept, a prototype PURO Joiner was implemented as a Javascript web app.<sup>4</sup> It allows for authoring and merging pairs of PURO models. When the merging is initiated, the modules are displayed side by side, and a list of automatically matched entities is shown (see example in Figure 1). The user can check and edit the mapped pairs, as well as add the mappings manually. After the merge operation has been performed, a single PURO model is displayed where the mapped entities are merged into a single node. The user can optionally ask for the automatic merging of instances. When the respective button is clicked, all instances of the same B-type are merged into a single node named 'general name of the B-type.' Figure 2 then shows the collapsing of a group of entities into a single node (in the visualization).

## 5. Related work

The task addressed by our new tool consists in supporting interactive merging of ontology modules for a particular modeling language, namely, PURO.

Automation of ontology module merging has so far received smaller attention than ontology alignment [5]. The reason probably is that the most popular ontology language is nowadays OWL, which is a language intended for ultimate publishing of operational, reusable ontologies. Merging of existing ontologies would typically result in the loss of

 $<sup>\</sup>label{eq:action} ^{4} Available \ at \ http://protegeserver.cz/PUROMJoiner/, \ https://github.com/TheCandy/PUROMJoiner, \ screencast: \ https://screencast-o-matic.com/watch/c3elYGVFTLI.$ 

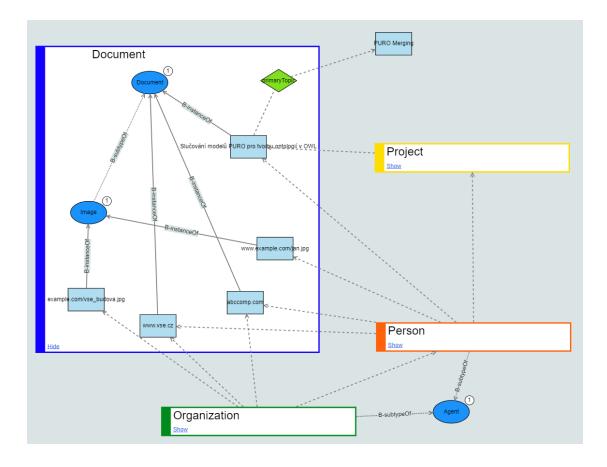


Figure 2: Larger merged model with some nodes grouped and collapsed

entities that are redundant from the point of view of domain modeling. However, such entities may have already been referred to by datasets / knowledge graphs, and data interoperability would then be compromised. Therefore, the usual result of ontology alignment is a mere creation of link sets (consisting of triples with predicate owl:sameAs or a similar one such as skos:exactMatch), while the original ontologies remain unaltered. However, since PURO addresses the early phase of ontology development when the entities are still under the control of the development team, substantial updates of the model structure do not affect other resources. Furthermore, ontology alignment primarily aims at fully automated alignment, which is desirable for large ontologies. In contrast, the manually created structures of PURO models are relatively small, which makes their interactive alignment and merging thus more feasible. However, they feature example instances that require specific handling.

A well-known, early tool suite similar to ours (but having more functionality) is PROMPT [7]. It enabled users to both align and merge (frame-based) ontological modules in an interactive mode. The main difference from our approach is in the substantial dissimilarity of the underlying modeling language (esp. the presence of instances, meta-types, and n-ary relationships in PURO), in the visual interface (PURO Modeler/Joiner uses a node-link view, while PROMPT relied on an indented list display), and possibly also in the overall scenario: PURO Joiner is primarily meant for combining modules that may possibly overlap in a few entities, rather than for merging multiple ontologies describing the same domain, as shown in the demonstration of PROMPT [7].

A more recent representative of interactive OWL ontology alignment tools is the Alignment tool [8]. It offers both a list-based and graph-based view of the two to-bealigned ontologies. However, it does not support merging.

## 6. Conclusions and future work

The proposed extension of PURO Modeler is likely to significantly improve its scalability for larger ontology skeletons, thus potentially leading to its broader adoption by ontology engineers.

We envision several improvements. The tool could automatically search for existing related entities in other modules and suggest them to the user. This would encourage the reuse of existing entities whenever possible, making the subsequent merging possibly fully automated. To enable collaboration between multiple users, a git-inspired versioning system could be implemented. The layout of the merged model could be arranged automatically for more clarity, e.g., collapsing parts of it based on some heuristics.

An important next step will be testing with real users. The first experiment will be qualitative. Several users will create partial modules, and then the group will collaborate on merging them, while we analyze the process and results. Future experiments will compare performance of users working on large models with and without the modularization.

## Acknowledgments

Supported by CHIST-ERA within the CIMPLE project (CHIST-ERA-19-XAI-003).

## References

- M. Dudáš, T. Hanzal, V. Svátek, O. Zamazal, OBOWLMorph: Starting ontology development from PURO background models, in: 12th OWLED@ISWC, 2015.
- [2] OWL 2 web ontology language structural specification and functional-style syntax (second edition), W3C Recommendation, 2012. URL: https://www.w3.org/TR/owl2syntax/.
- [3] G. Guizzardi, Ontological foundations for structural conceptual models, CTIT, Centre for Telematics and Information Technology, 2005.
- [4] M. Dudás, T. Morkus, V. Svátek, T. P. Sales, G. Guizzardi, Kickstarting OntoUML modeling from PURO instance-level examples, in: Proc. EKAW 2020 Posters and Demonstrations, volume 2751 of CEUR Workshop Proceedings, CEUR-WS.org, 2020, pp. 36–40.
- [5] J. Euzenat, P. Shvaiko, Ontology Matching, Second Edition, Springer, 2013.

- [6] C. Fellbaum (Ed.), WordNet An Electronic Lexical Database, The MIT Press, Cambridge, MA, 1998.
- [7] N. F. Noy, M. A. Musen, The PROMPT suite: interactive tools for ontology merging and mapping, Int. J. Hum. Comput. Stud. 59 (2003) 983–1024.
- [8] S. Karampatakis, C. Bratsas, O. Zamazal, P. Filippidis, I. Antoniou, Alignment: A hybrid, interactive and collaborative ontology and entity matching service, Inf. 9 (2018) 281.