CoModIDE – The Comprehensive Modular Ontology Engineering IDE*

Cogan Shimizu[1]✉ and Karl Hammar[2][0000−0001−8767−4136]

[1] Data Semantics Lab
Wright State University, USA
shimizu.5@wright.edu
[2] Jönköping AI Lab
Jönköping University, Sweden
karl.hammar@jonkoping.ai

1 Introduction

Ontology engineering is a complex and time-consuming process, requiring an intimate knowledge of description logic and predicting non-local effects of different ontological commitments. Acquiring such expertise is a major hurdle in the adoption of semantic web technologies. As proliferating our techniques and technologies is a major goal of the semantic web community [4], there have been many attempts to improve the accessibility of ontology engineering through intuitive methodologies and robust tooling. Pattern-based modular ontology engineering coupled with a graphical modelling paradigm can help bridge this gap [6,9].

A pattern-based modular ontology is an ontology that retains the patterning metadata associated with the modules that comprise it. An ontology design pattern (ODP) is a small, reusable set of concepts and axioms that solve a problem that is invariant across many domains. To create such ontologies, some methodologies have been developed (e.g. Extreme Design [2,3] and the eponymous Modular Ontology Modelling [5]). Unfortunately, neither methodology focuses on the retention of pattern metadata, but how to identify and instantiate patterns.

Over the years, there have been various approaches for representing ontologies visually and enabling their development through a graphical modelling interface, e.g., VOWL, the visual syntax for OWL and its WebVOWL editor [7,11]; OWLGrEd, a graph editor that displays a UML inspired subsumption hierarchy [1]; Gra.fo[3], an online, collaborative platform that supports ontology development via lightweight semantics and a modified VOWL syntax.; and OWLAx [8] and SDOnt [10], Protégé plugins that enable axiom generation from schema diagrams, and diagram generation from axioms, respectively. However, none of these tools offer any support for graphical pattern discovery or instantiation.

To combine pattern-based modular ontology engineering with the graphical modelling paradigm, we have developed the Comprehensive Modular Ontology IDE (CoModIDE, pronounced 'commodity'), a plug-in for the Protégé platform.

[3] https://gra.fo/

2 System Design and Features

CoModIDE was designed to simplify ontology engineering for users who are not ontology experts. Our experience indicates that such non-experts rarely need or want to make use of the full set of language constructs that OWL 2 provides; instead, they typically, at least at the outset, want to model rather simple semantics. Our experience also indicates that such users (and, indeed also more advanced users) prefer to do initial modeling graphically – whether that be on whiteboards, in vector drawing software, or even on paper. Finally, our experience indicates that while this category of users are generally enthusiastic about the idea of reusing design patterns, they are quickly turned off of the idea when they are faced with patterns that lack documentation or that exhibit link rot[4].

These experiences led directly to the design criteria for CoModIDE:

- CoModIDE should support visual-first ontology engineering, based on a graph representation of classes, properties, and datatypes. This graphical rendering of an ontology built using CoModIDE should be consistent across restarts, machines, and operating system or Protégé versions.
- CoModIDE should support the type of OWL 2 constructs that can be easily and intuitively understood when rendered as a schema diagram. To model more advanced constructs (unions and intersections in property domains or ranges, the subsumption hierarchy, property chains, etc), the user can drop back into the standard Protégé tabs.
- CoModIDE should embed an ODP repository. Each included ODP should be free-standing and completely documented. There should be no external dependency on anything outside of the user's machine. If the user wishes, they should be able to load a separately downloaded ODP repository, to replace or complement the built-in one.
- CoModIDE should support simple composition of ODPs; patterns should snap together like Lego blocks, ideally with potential connection points between the patterns lighting up while dragging compatible patterns. A pattern or ontology interface concept will need be developed to support this.

CoModIDE is developed as a plugin to the versatile and well-established Protégé ontology engineering environment. The plugin provides three Protégé views, and a tab that hosts these views (see Figure 1). The *schema editor* view provides an a graphical overview of an ontology's structure, including the classes in the ontology, their subclass relations, and the object and datatype properties in the ontology that relate these classes to one another and to datatypes. All of these entities can be manipulated graphically through dragging and dropping. The *pattern library* view provides a set of built-in ontology design patterns, sourced from various projects and from the ODP community wiki[5]. A user can drag and drop design patterns from the pattern library onto the canvas to instantiate those patterns as modules in their ontology. The *configuration* view lets

[4] Typically patterns that depend on other patterns which no longer resolve.
[5] `http://ontologydesignpatterns.org/`

the user configure the behavior of the other CoModIDE views and their components. For a detailed description, we refer the reader to the video walkthrough on the CoModIDE webpage[6]. We also invite the reader to download and install CoModIDE themselves, from that same site.

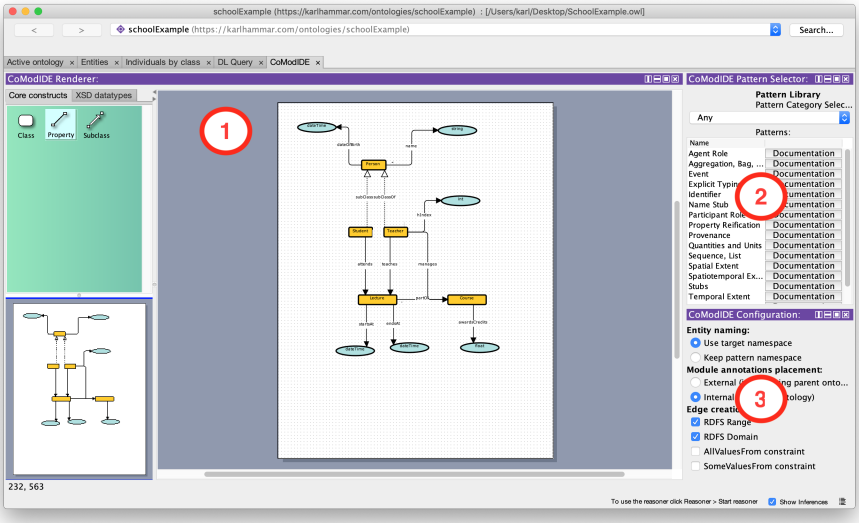

Fig. 1. CoModIDE User Interface featuring 1) the schema editor, 2) the pattern library, and 3) the configuration view.

When a pattern is dragged onto the canvas, the constructs in that pattern are copied into the ontology (optionally having their IRIs updated to correspond with the target ontology namespace), but they are also annotated using the OPLa vocabulary, to indicate 1) that they belong to a certain pattern-based module, and 2) what pattern that module implements. In this way module provenance is maintained, and modules can, provided that tool support exists (see Section 3) be manipulated (folded, unfolded, removed, annotated) as needed.

3 Discussion and Future Work

CoModIDE is under active development and is not yet feature-complete. Specifically, during the autumn of 2019 we will implement the following features:

– Wrapping instantiated modules (e.g., in dashed-line boxes) to indicate cohesion and to allow module folding/unfolding.

[6] `https://comodide.com`

- An interface feature, allowing design patterns to express how they can be connected to one another; and adding support for this to the canvas, lighting up potential connection points as the user drags a pattern.
- Support for custom pattern libraries; and vocabulary specifications indicating hos pattern libraries should be annotated to be useful with CoModIDE.

To evaluate the viability of our approach to ontology engineering, and the usability of the CoModIDE tool, we will be deploying CoModIDE in two research projects with non-ontologist domain experts.

References

1. Barzdins, J., Barzdins, G., Cerans, K., Liepins, R., Sprogis, A.: OWLGrEd: a UML style graphical notation and editor for OWL 2. In: Proceedings of the 7th International Workshop on OWL: Experiences and Directions (OWLED 2010), San Francisco, California, USA, June 21-22, 2010. CEUR-WS.org, vol. 614 (2010)
2. Blomqvist, E., Hammar, K., Presutti, V.: Engineering Ontologies with Patterns – The eXtreme Design Methodology. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns: Foundations and Applications, pp. 23–50. IOS Press (2016)
3. Hammar, K.: Ontology design patterns in WebProtégé. In: Villata, S., Pan, J.Z., Dragoni, M. (eds.) Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015). CEUR Workshop Proceedings, vol. 1486. CEUR-WS.org (2015)
4. Hammar, K., Blomqvist, E., Carral, D., et al.: Collected research questions concerning ontology design patterns. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns - Foundations and Applications, pp. 189–198. IOS Press (2016)
5. Hitzler, P., Krisnadhi, A.: A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. CoRR **abs/1808.08433** (2018)
6. Hitzler, P., Shimizu, C.: Modular ontologies as a bridge between human conceptualization and data. In: Chapman, P., Endres, D., Pernelle, N. (eds.) 23rd International Conference on Conceptual Structures, ICCS 2018, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10872, pp. 3–6. Springer (2018)
7. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016)
8. Sarker, M.K., Krisnadhi, A.A., Hitzler, P.: Owlax: A protege plugin to support ontology axiomatization through diagramming. In: Kawamura, T., Paulheim, H. (eds.) Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016. CEUR Workshop Proceedings, vol. 1690. CEUR-WS.org (2016)
9. Shimizu, C.: Towards a comprehensive modular ontology IDE and tool suite. In: Kirrane, S., Kagal, L. (eds.) Proceedings of the Doctoral Consortium at ISWC 2018. CEUR Workshop Proceedings, vol. 2181, pp. 65–72. CEUR-WS.org (2018)
10. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: A modular ontology design library. CoRR **abs/1904.05405** (2019), `http://arxiv.org/abs/1904.05405`
11. Wiens, V., Lohmann, S., Auer, S.: Webvowl editor: Device-independent visual ontology modeling. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks. CEUR Workshop Proceedings, vol. 2180. CEUR-WS.org (2018)