Towards a Comprehensive Modular Ontology IDE and Tool Suite

Cogan Shimizu

Data Semantics Laboratory, Wright State University, Dayton, OH, USA

Abstract. Published ontologies frequently fall short of their promises to enable knowledge sharing and reuse. This may be due to too strong or too weak ontological commitments; one way to prevent this is to engineer the ontology to be modular, thus allowing users to more easily adapt ontologies to their own individual use-cases. In order to enable this engineering paradigm, there is a distinct need for developing more supporting tools and infrastructure. This increased support can be immediately impactful in a number of ways: guides engineers through best practices and promote ontology design pattern discovery, sharing, and reuse. To meet these needs, this PhD project explores the development of a comprehensive modular ontology IDE and tool suite.

1 Problem Statement

One of the central tenets of the Semantic Web is to enable the sharing and reuse of knowledge. Unfortunately, published ontologies infrequently live up to these promises, as they are not developed with best practices in mind, such as modularization, documentation, and annotation.

Ontologies with too strong or too weak ontological commitments are undesirable. Strong ontological commitments lead to overspecialization; this may constrain ontologies to be useful only for the single usecases for which the ontologies were developed. Conversely, weak commitments lead to overly ambigious models, thus making it difficult to understand how to use the ontology, at all. In order to combat this, during development an ontology should be sufficiently modularized. Such ontologies [6] are designed so that engineers may adapt them to individual use-cases, yet still maintain compatibility and integration with other versions of the ontology [7].

Furthermore, it is necessary to properly document and annotate the developed ontology. In order to exhibit and promote the use of the ontology among a domain, other engineers must understand *how* to use or adapt the ontology, as well as understand the nature of certain ontological commitments or other design decisions [5]. In addition, annotations made with a pattern representation language allow engineers to understand how an existing ontology made use of or relates to another ontology [4]. Finally, both documentation and annotation allow content providers or data publishers utilize these models.

However, developing an ontology, while following these best practices, is a very difficult and time-consuming process, especially without proper tooling and

supporting infrastructure. As such, this PhD project explores the development of a comprehensive, modular ontology integrated development environment (IDE) in order to address the needs for ontology design pattern (ODP) discovery and modularization and engineering ontologies with best practices.

2 Relevancy

As described in the previous section, designing *modular* ontologies is an answer to some of the problems facing the Semantic Web community, namely those regarding the sharing and reuse of knowledge [3]. Thus, it seems particularly prudent to incentivize the adoption of this engineering paradigm.

Recently, there have been many attempts to do so, ranging from new visualizations, improved methodologies, and more accessible modeling tools. Unfortunately, these attempts are largely uncoordinated; each individual attempt is focused on improving a single aspect of the process. For example, they all do not share the same unifying platform nor necessarily work well together (and sometimes even at cross purposes). Further, some improvements may be strictly theoretical or methodological with no usable implementation.

In summary, there is currently no comprehensive nor integrated approach for developing modular ontologies according to best practices. However, there is now a critical mass of individual, tools with specialized functionalities. We believe the best approach for moving forward is to increase the support (e.g. tooling and infrastructure) available to ontology engineers by fusing together existing support. Thus, developing a comprehensive, modular ontology IDE will help increase the adoption of the modular ontology engineering paradigm and is thus very relevant to the Semantic Web community.

3 Research Questions

There are a number of open research questions regarding the future of modular ontology development, especially regarding tooling and infrastructure, as outlined by the Semantic Web community in [3]. We discuss the most relevant of them: "Which kind of tools are needed and best suited for ODP development and use?"

As a first approximation, Hammar et al. describe the need for a "pattern-capable ontology IDE." That is, an ontology IDE that is capable of using ontology design patterns as primitives, as well as having some mechanism for pattern discovery. A modular ontology IDE takes this concept a step further and would allow users to import and modularize ODPs to also be used as primitives. Thus, we seek to answer the following questions.

RQ1. What foundational support is needed to realize such an IDE?
RQ2. How can new and existing tools be combined to form a cohesive and useful whole, while still leaving room for extensibility?

Our attempts to answer RQ1 are described in Section 6. For RQ2, we describe some existing tools and methods in Section 5.

4 Hypotheses

In line with the previous section, we also want to show that a comprehensive, modular ontology IDE is, in fact, effective. Thus, we will attempt to confirm the following hypotheses.

> A comprehensive, modular ontology IDE will allow an ontology engineer to develop ontologies
>
> 1. more quickly than when not using such an IDE.
> 2. that adhere to best practices for modularity, documentation, and annotation.

How we will acheive this and how we determine success are discussed in Sections 6 and 7, respectively.

5 Related Work

Overall, there are many existing tools and methods for helping develop ontologies. As this PhD project is specifically concerned with *modular* ontology engineering, we most carefully consider those related tools and methods. In this section, we provide broad descriptions of related work that this PhD project will expand, adapt, or otherwise utilize. We may partition the related works into three categories, based on how they intersect with the ontology engineering process: Methodology, Visualization and Rendering, and Tools and Infrastructure.

Methodology

By methodology, we refer to those related works that deal with guidelines and principles for engineering ontologies. Of particular interest is the eXtreme Design (XD) Methodology and methods for documenting ontology design patterns.

The eXtreme Design (XD) Methodology is a "family of methods and associated tools... for for solving ontology development issues" [1]. The XD Methodology is in many ways the core methodology for modular ontology design and development. It outlines how to identify the need for patterns, how to utilize content information, and guidelines for different modelling approaches. We do not intend to replace XD, but instead use its principles. For example, we ma use its different design approaches to inform different content pattern suggestions during development.

Karima et al. give a thorough walkthrough on how to document ontology design patterns [5]. It provides key components of patterns that should be documented as well as criteria for measuring how well an ontology is documented. Thus, we may leverage these guidlines in order to provide tooling that prompts users to "document as they go," ultimately reducing documentation overhead.

Visualization & Rendering

This category refers to those tools and methods that facilitate alternative views of an ontology. For example, functional syntax for OWL or Turtle or Manchester Syntax are alternate renderings of the same information. As the semantic web community more deeply and consistently interacts with domain experts, it is very important to find vehicles for representing an ontology that is easy for people to intuit. Below, we describe two recent tools for doing this. As part of the evaluation of this PhD project, we will also evaluate the efficacy of presenting ontological information in this manner.

OWL2Rules is an augmentation[1] to the OWLAPI's LaTeX rendering framework. This tool is capable of representing the axioms of an ontology as First Order Predicate Logic Existential Disjunctive Rules. It also incorporates the improved LaTeX formatting from [10].

SDont is a tool[2] for creating the schema diagrams for ontologies. While there are other visualization tools (e.g. OWLgred and VOWL), SDont has been engineered to generate schema diagrams that are maximally similar to human curated schema diagrams. We intend to incorporate this tool in order to provide a more comfortable vehicle for representing the structure of a TBox.

Tools & Infrastructure

Tools and Infrastructure refers to the tools and methods that assist in the ontology engineering process. On their own, each of these tools is enormously helpful. However, a platform utilizing them will be greater than the sum of its components.

OPLa is an Ontology Design Pattern Representation Language [4]. This will enable ontology engineers to leverage OWL annotations to describe the ontological entities within a patern. That is, the annotations can be used to show from where properties are inherited, show which patterns were used to create modules, or show which concepts in the pattern can be used as hooks for external patterns. There is an existing plugin[3] that guides users through annotating an ontology design pattern.

ROWL & OWLAx are Protégé plugins [8, 9]. RowlTab is used for creating owl axioms (or the appropriate SWRL rule) from first order predicate logic rules. OWLAx is a graphical tool that generates the owl axiom (or appropriate SWRL rule) from schema diagram like representations. As well as generating the scoped domain and range axioms, and disjointness axioms.

[1] `https://github.com/cogan-shimizu-wsu/Logician`
[2] `https://github.com/cogan-shimizu-wsu/SDont`
[3] `https://github.com/cogan-shimizu-wsu/OPLaPlugin`

XD for Protégé is a first approximation of a pattern-capable ontology IDE implemented in WebProtégé [2]. Among other XD motivated functionalities, the following are of particular note: composite search engine, ODP specialization stragety importing, and ODP specialization alignment suggestions. This tool is the main source of inspiration for this PhD project and will act as a foundation upon which to build a CoModIDE.

6 Approach

In aggregate, our approach is to tie together multiple existing tools (e.g. XD for WebProtégé and OWLAx), develop or enhance new tools (e.g. SDont) and combine them into a comprehensive, modular ontology IDE (CoModIDE– pronounced 'commodity'). In addition, CoModIDE would be tightly integrated with a central, "smart" repository that will facilitate ODP discovery and importing. To address our research questions, we have split the approach into two distinct phases: foundational work and IDE development.

Phase I: Foundational Work

This phase largely addresses RQ1: "What foundational support is needed to realize CoModIDE?" Phase I will have the following trajectory.

Step 1: **Functionality Solicitation.**
We must first determine exactly which functionalities are most useful and helpful to the Semantic Web community. For this step, we will solicit suggestions from the community. In addition, we will use the suggestions from [3].

Step 2: **Discovery of Existing Tools.**
For those functionalities that have been determined to be integral to the community, we will need to discover any tools that already implement that functionality. If we cannot find an existing tool that covers that functionality, then a new tool or method will need to be developed to cover that gap.

Step 3: **Individual Tool Evaluation.**
Now, for each of the discovered or developed tools, there must be an individual evaluation in order to determine efficacy and correctness of the tool.

Step 4: **Extend and Enhance the Repository.**
Currently, `www.ontologydesignpatterns.org` is the central repository for ODP sharing and reuse. With the recent development of OPLa [4], we may further enhance the functionality of the site. In addition, we will need to ensure that there is a so-called critical mass of usable, well-documented, and thoroughly annotated ODPs available for the community to use.

Phase II: IDE Development

After we have identified, developed, evaluated individual useful tools, what remains is to integrate them into a comprehensive, modular ontology IDE; we call it CoModIDE. Overall, we foresee the trajectory to be as follows.

Step 1: **Choose a Platform.**
We will need to choose a platform upon which to build the IDE. At the time of this writing, we believe that Desktop Protégé is the best choice. It offers built-in plugin support and very tight integration with the OWLAPI.

Step 2: **Integrate Tools.**
Once a platform has been chosen, we may need to re-implement the functionality of existing tools for that platform. For example, XD for WebProtégé is *only* implemented for WebProtégé and itself has several pieces of our desired functionality.

Step 3: **Evaluate CoModIDE.**
After all the tools have been integrated so that they work together, we must evaluate whether or not the CoModIDE is achieving the desired purpose, i.e. does it help users design better ontologies more quickly? We describe this step in more detail in the next section.

Additionally, we must consider overall design.

- **UX and Workflow:** There are some functionalities that can only be considered once the IDE has been developed, such as determining the best way to guide users to document and annotate the ontology as its being designed. Additionally, we would like to implement a graphical plug'n'play interface. That is, using ODPs as primitives (similar to puzzle-pieces), connect the ODPs to form a first-pass ontology.
- **Extensibility:** The "modular" in CoModIDE need not only apply to the ontology design. In fact, it would behoove us to ensure that the IDE itself is modular, in order to continue adding functionality as modular ontology design evolves in the future.
- **Repo interfacing:** It is completely necessary that the IDE support interfacing with a central repository and a pattern representation language. At this time, we expect that to be ontologydesignpatterns.org and OPLa, respectively.

7 Evaluation Plan

We have posited that a comprehensive, modular ontology IDE will allow engineers to more quickly design ontologies, while following best practices. To determine if we have successfully done so, we will conduct evaluations in two parts.

First, each individual functionality of the IDE must be tested. For some of these tools (e.g. ROWL [8], OWLax [9], XD for WebProtégé[2]), these tools

have already been evaluated; their evaluations are available in the respective references. However, for those tools we have not yet written, or need to discover, their evaluations will necessarily be tailored to the functionality and cannot be described here.

After all the tools and desired functionalities have been implemented and evaluated, we will evaluate the sum-total: CoModIDE. This will necessarily be a user evaluation.

We will have a test group and control group. Each group will be given a moderately compex modeling task. The test group will be asked to design the ontology using CoModIDE; the control group will not. We will then measure the amount of time it took to complete the design task, measure its efficacy at providing answers to pre-selected compentency questions, and identify if the ontology has been designed modularly and with best practices, as according to [1, 5, 6].

8 Preliminary Results

At the time of this writing, we have only implemented the very beginning stages of our proposed approach and do not have any results to report. However, for the established tools (e.g. ROWL or XD for WebProtégé), see their respective references.

9 Reflections

It is not that others have failed, but that there is finally a critical mass in successful, existing tools and methods for modular ontology design. Thus, we believe that unifying them will result in a better tool for producing better ontologies.

By unifying the tools, we believe that we will decrease the time spent switching between tools, formats, and the like. The ability to communicate with a central repository, such as `ontologydesignpatterns.org` will greatly facilitate pattern discovery and utilizing a standardized pattern representation language, such as OPLa, will allow engineers to make better decisions more quickly.

Acknowledgement. The author acknowledges funding from the Dayton Area Graduate Studies Institute (DAGSI) and thanks Pascal Hitzler for his significant input.

References

1. E. Blomqvist, K. Hammar, and V. Presutti. Engineering ontologies with patterns - the extreme design methodology. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns - Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 23–50. IOS Press, 2016.

2. K. Hammar. Ontology design patterns in webprotege. In S. Villata, J. Z. Pan, and M. Dragoni, editors, *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.*, volume 1486 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.
3. K. Hammar, E. Blomqvist, D. Carral, M. van Erp, A. Fokkens, A. Gangemi, W. R. van Hage, P. Hitzler, K. Janowicz, N. Karima, A. Krisnadhi, T. Narock, R. Segers, M. Solanki, and V. Svátek. Collected research questions concerning ontology design patterns. In P. Hitzler et al., editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 189–198. IOS Press, 2016.
4. P. Hitzler, A. Gangemi, K. Janowicz, A. A. Krisnadhi, and V. Presutti. Towards a simple but useful ontology design pattern representation language. In E. Blomqvist et al., editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
5. N. Karima and P. Hitzler. How to document ontology design patterns. In K. Hammar, P. Hitzler, A. Lawrynowicz, A. Krisnadhi, and V. Presutti, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 97–104. IOS Press, 2017.
6. A. Krisnadhi and P. Hitzler. Modeling with ontology design patterns: Chess games as a worked example. In P. Hitzler et al., editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 3–21. IOS Press, 2016.
7. A. Krisnadhi, Y. Hu, K. Janowicz, P. Hitzler, R. A. Arko, S. Carbotte, C. Chandler, M. Cheatham, D. Fils, T. W. Finin, P. Ji, M. B. Jones, N. Karima, K. A. Lehnert, A. Mickle, T. W. Narock, M. O'Brien, L. Raymond, A. Shepherd, M. Schildhauer, and P. Wiebe. The geolink modular oceanography ontology. In M. Arenas et al., editors, *The Semantic Web – ISWC 2015 – Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 301–309. Springer, 2015.
8. M. K. Sarker, A. Krisnadhi, D. Carral, and P. Hitzler. Rule-based OWL modeling with rowltab protégé plugin. In E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, and O. Hartig, editors, *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science*, pages 419–433, 2017.
9. M. K. Sarker, A. A. Krisnadhi, and P. Hitzler. Owlax: A protege plugin to support ontology axiomatization through diagramming. In T. Kawamura and H. Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016.*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.
10. C. Shimizu, P. Hitzler, and M. Horridge. Rendering OWL in description logic syntax. In E. Blomqvist, K. Hose, H. Paulheim, A. Lawrynowicz, F. Ciravegna, and O. Hartig, editors, *The Semantic Web: ESWC 2017 Satellite Events - ESWC 2017 Satellite Events, Portorož, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, volume 10577 of *Lecture Notes in Computer Science*, pages 109–113. Springer, 2017.