# Advances in modular ontology engineering: methodology and infrastructure

by

Cogan Matthew Shimizu

B.S., Wright State University, 2016

M.S., Wright State University, 2017

---

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

# Abstract

Modular ontology engineering is a methodology for producing highly reusable knowledge graph schema. Over the course of this dissertation, we outline a number of contributions that have improved the process to what we see today. These contributions fall within four categories: conveying meaning through schema diagrams, the composition of a modular ontology, the modular ontology engineering methodology, and modular graphical modeling.

First, we created an improved method and tool for generating schema diagrams similar to those manually generated by humans and show that most of OWL, as it is used in real world ontologies, are expressible in this format.

Next, we examined and improved the ontology design pattern development process. This was accomplished through the development of both patterns and modules, extensions to the ontology design pattern representation language, and a tool that significantly improves the usability of these annotations. This work culminated in MODL: a modular ontology design library, which is a distributable set of curated, well-documented ODPs, both novel and drawn from the ontology design pattern portal.

These advances were combined, and building upon the state of the art, to create the Comprehensive Modular Ontology Design IDE (CoModIDE), which is a plugin for the industry-standard ontology editor, Protégé.

Finally, as a culmination of the tool and the methodology, we evaluated CoModIDE, where it was shown to significantly improve outcomes for experienced and new ontology developers when developing modular ontologies.

Altogether, these research topics, resulted in a methodology, that when executed, produced actually reusable, extendable, and adaptable ontologies.

# Advances in modular ontology engineering: methodology and infrastructure

by

Cogan Matthew Shimizu

B.S., Wright State University, 2016

M.S., Wright State University, 2017

---

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

Approved by:

Major Professor
Pascal Hitzler

# Copyright

# Abstract

Modular ontology engineering is a methodology for producing highly reusable knowledge graph schema. Over the course of this dissertation, we outline a number of contributions that have improved the process to what we see today. These contributions fall within four categories: conveying meaning through schema diagrams, the composition of a modular ontology, the modular ontology engineering methodology, and modular graphical modeling.

First, we created an improved method and tool for generating schema diagrams similar to those manually generated by humans and show that most of OWL, as it is used in real world ontologies, are expressible in this format.

Next, we examined and improved the ontology design pattern development process. This was accomplished through the development of both patterns and modules, extensions to the ontology design pattern representation language, and a tool that significantly improves the usability of these annotations. This work culminated in MODL: a modular ontology design library, which is a distributable set of curated, well-documented ODPs, both novel and drawn from the ontology design pattern portal.

These advances were combined, and building upon the state of the art, to create the Comprehensive Modular Ontology Design IDE (CoModIDE), which is a plugin for the industry-standard ontology editor, Protégé.

Finally, as a culmination of the tool and the methodology, we evaluated CoModIDE, where it was shown to significantly improve outcomes for experienced and new ontology developers when developing modular ontologies.

Altogether, these research topics, resulted in a methodology, that when executed, produced actually reusable, extendable, and adaptable ontologies.

# Table of Contents

# List of Figures

# Acknowledgments

During this journey, I have had the fortune to be supported by a number of truly incredible people. First, of course, I would like to thank my advisor, Pascal Hitzler, for encouraging me at every step to question and drive forward with my own ideas, and consistently (relentlessly, even) providing me with new opportunities to grow as both an individual and researcher. Also, to the many collaborators that have pushed my boundaries in so many new and interesting domains.

To all of my friends, for sticking with me even when I disappeared for weeks at a time. Thank you, in particular, to Sifis for reminding me to eat regularly.

A special thanks to my family, for giving me the foundation to see this through to the end and, in loving memory of William H. Shimizu, whose quiet support inspired me to make every step count, even when I faltered.

The author acknowledges partial support from the Dayton Area Graduate Studies Insitute (DAGSI) award RH17-13, the financial assistance award 70NANB19H094 from U.S. Department of Commerce, National Institute of Standards and Technology, the National Science Foundation under Grant No. 1936677, the Air Force Office of Scientific Research under award number FA9550-18-1-0386, and The Andrew W. Mellon Foundation through the Enslaved project (identifiers 1708-04732 and 1902-06575).

# Dedication

*To Riley Layne Chyoko Shimizu, who always wanted to call me 'doctor.'*

# Chapter 1

# Introduction

## 1.1 The Semantic Web and Data

The Semantic Web is both an active, growing area of research, as well as an expansive ecosystem for the delivery and linkage of machine-readable knowledge.

As a field of research, it has an immense breadth and depth of activity that draws from many different fields. This seems fitting for a field focused on the efficient representation of knowledge from any domain. The Semantic Web, as an artifact, is closely related to the World Wide Web (WWW). This is ultimately unsurprising as they share the same goal: to proliferate knowledge in a widely accessible manner. They simply differ for whom they emphasize accessibility. Just as the WWW is an ecosystem of technologies and standards for sharing data amongst its human users, the Semantic Web, both artifact and research field together, is an analogous ecosystem for machines. Fundamentally, the Semantic Web is a way to ascribe to web content meaning. That is, to carefully describe the semantics of the content in a machine-readable way. Consequently, this enables programmatic access, interpretation, and evaluation of knowledge previously encoded in an only human-readable format.

Ultimately, the Semantic Web, as a both field and artifact, drives how knowledge is linked, published, reused, and analyzed. One important, contemporary tool for doing so, is

the knowledge graph. While a more in-depth description is deferred to the next section, it is useful to also have a historical perspective of their origin: what standards constitute them and what is the context of their evolution.

As mentioned previously, the Semantic Web is concerned with the *meaning* of data. Over the years, there have been many different ways for accomplishing such attribution. Perhaps most prominently, and still in wide use today, is the Resource Description Framework (RDF). RDF is a W3C standard that allows for the annotation of data in a structured way.

One common way of annotating data is through the use of a vocabulary. This allows researchers and developers to connect their data to a standardized set of concepts or relationships that span domains. Some examples are VOID,[1] DCAT,[2], the Dublin Core,[3] and Schema.org. By agreeing on the meaning of these terms, applications can then parse through datasets and begin to understand the data by leveraging the annotations. This begets an interesting phenomenon: datasets linked through these vocabularies in a massive cloud of datasets spanning a myriad of domains.

Further building on top of RDF are so-called ontologies, which are "explicit specifications of conceptualizations" [8] which can be authored using an ontology language. For this dissertation, however, we focus on another W3C Standard, the Web Ontology Language (OWL) [16].[4]

OWL comes in several profiles, or species, that indicate its level of expressivity. OWL–Full, for example, is very expressive, but can be computationally intractable. Overall, the contributions of this dissertation focus on OWL–DL, which retains a high level of expressivity, but remains computationally tractable. More generally, OWL can be used model complex concepts in terms of simpler concepts. These simpler concepts are used as building blocks and are heavily enriched with meta-data that relate them to each other. In this way, we can represent a highly abstract and complex concept in a way that a machine can easily interpret.

[1] See https://www.w3.org/TR/void/.
[2] See https://www.w3.org/TR/vocab-dcat-2/.
[3] See https://dublincore.org/.
[4] This focus is, in turn, a result of the focus on OWL as a Semantic Web language.

The process by which an ontology is developed is called a methodology. Over the decades, several methodologies have been developed. The historical basis for modular ontology engineering, as discussed later in the dissertation, is founded in the eXtreme Design methodology, which is a pattern-based approach to ontology development [26, 3], which itself stems from the task-oriented, scenario-based, NeOn methodology [41].

An ontology provides structure to data, showing how it may be related and used, thus allowing to constrain properties, check for the consistency of the dataset, or derive new data. Linked Data shows how data may be reused across domains. Together, linked data and ontologies have paved the way for a new term: the knowledge graph.

## 1.2 Knowledge Graphs & Ontologies

A knowledge graph is a way of organizing information using a graph structure. In practice, there are many different sorts of graph structures, such as RDF graphs or labeled property graphs. For this dissertation we focus on RDF graphs, or at least RDF serializations of OWL. However, it is possible to move between these representations through modeling techniques or through reification. By choosing RDF graphs we have the added benefit of utilizing another W3C standard, SPARQL, for efficiently querying the graph. RDF graphs can be expressed using sets of triples. That is, a subject-predicate-object statement that captures some perception of a limited facet of reality. In contemporary Google search results, there are frequently "infoboxes" that appear on the right-hand side of the screen. Figure 1.1 shows how one could imagine the a knowledge graph constructed from this sort of data. Figure 1.2 provides the actual graph view.

Due to their inherently flexible nature, knowledge graphs are posed to be a significant disruptor in both the public and private sectors [25]. They have quickly become a major paradigm for data integration, communication, and visualization, supported by long-established W3C standards and recommendations [16, 4, 1, 21]. For Linked Data [2, 24] alone, which is but one form of a knowledge graph, a 2017 count showed more than 35 billion node-edge-node information triples freely available on the Web [23]. Schema.org data,

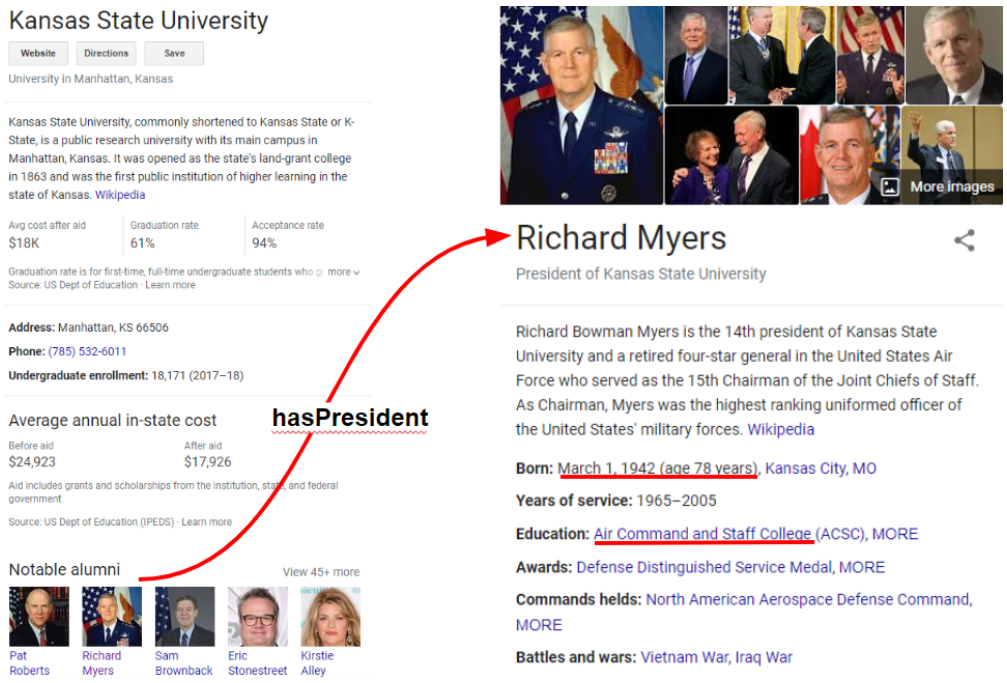



**Figure 1.1**: *Google Infocards. We can extract pertinent information from these infocards and construct a graph-like structure.*

which also constitutes a massive knowledge graph already in 2015, shows that over 30% of Web pages had corresponding markup [9]. As a repository of both declarative and procedural knowledge, a knowledge graph can be considered to be more than just a resource, but instead could be considered to be a platform upon which to build. However, the development of a such a knowledge graph, as with any complex system, is a costly endeavor, especially with respect to time and expertise. For small firms, building and maintaining a knowledge graph can be untenable.

Recently, there has been an emphasis on FAIR data principles (Findable, Accessible, Interoperable, and Reusable) [42]. Each of the above facets have a large impact on reducing the costs across the entire knowledge graph ecosystem by supporting the infrastructure of the ecosystem, as well as reducing the barrier of entry to engaging with the ecosystem.

To some extent, it is reasonable to group together, as complementary pairs, Findability and Accessibility; and Interoperability and Reusability, each speaking to half of the journey. An easily findable resource is worth little if it is inordinately difficult to reuse, and the perfectly reusable resource is hardly that if it cannot be found. The latter pair, interoperability

and reusability, is largely dictated by the knowledge graph's schema (assuming, of course, that the described data is of sufficient quality). As published in [42], interoperable and reusable have specific meanings:

- *Interoperable* – interoperates with applications or workflows for analysis, storage, and processing.
- *Reusable* – metadata and data should be well-described so that they can be replicated and/or combined in different settings.

However, the Reusability principle, and its corollaries, do not go far enough when applied to knowledge graphs. The reusability of a knowledge graph is more than its metadata on provenance or licensing. One of the biggest impediments to reusing a knowledge graph lies with the design of its schema [19, 27] – the commitments made as part of such a schema are sometimes referred to as *ontological commitments*. In addition to those principles outlined in [42], we append the following.

- *Reusable* – the design of the schema is amenable to adaptation, evolution, and follows best practices as outlined by the domain of interest.

Unfortunately, many knowledge graphs are not reusable in this sense [19, 27]. Large, monolithic knowledge graphs, designed with very strong – or very weak – ontological commitments in their schema are very difficult to reuse across the same domain, let alone across different domains. Strong ontological commitments lead to over-specification, to ontologies essentially being only fit for the singular purpose for which they were originally designed. Conversely, weak ones lead to ambiguity of the model, sometimes to the extent that is hard to grasp what is actually being modeled [17].

Ontologies, which may constitute a schema for a knowledge graph, are used at a large scale for many purposes. The schema is a mechanism by which the data can be organized: it can inform constraints on the data, and thus the shape of the graph; and provide a vehicle for exploring and navigating its contents. A schema may also act as built-in documentation for the knowledge graph, in the sense that it disambiguates meaning [14]. As such, having a schema is useful for nearly any use-case of a knowledge graph.

We posit that one effective way to obtain ontologies which are easier to reuse, especially

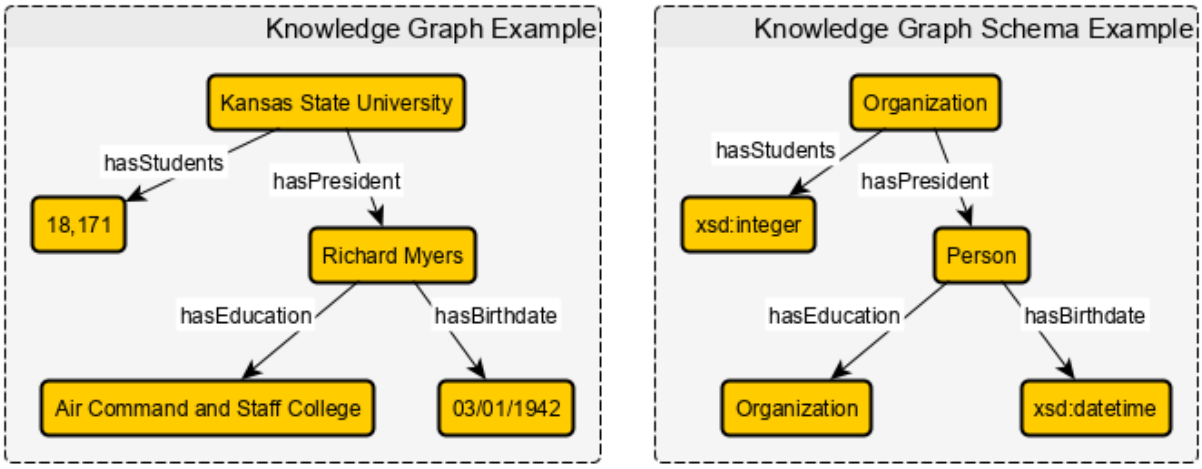



**Figure 1.2**: *An Example (Tiny) Knowledge Graph and its Schema. The raw data can be seen in Figure 1.1.*

with respect to the above definition of *Reusability*, is to build them in a modular fashion. A sufficiently modularized ontology is designed such that individual users can easily adapt an ontology to their use cases, while maintaining integration and relationships with other versions of the ontology [22]. Briefly, a modular ontology is constructed by piecing together so-called ontology modules. Ontology modules are created by adapting ontology design patterns to the domain and use-case [12]. Additional information can be found in Section 4.1. The process by which this is accomplished is eponymous: modular ontology engineering. This methodology is a pattern-based approach that stems from previous methodologies.

The current form of the methodology can be found in Figure 1.3. The methodology is addressed in detail in Chapter 4.

This dissertation describes the efforts made to mature this methodology, and focused on two primary tasks.

1. Improve the approachability of the methodology by improving tool support and infrastructure, as well as maturing the methodology and providing real-world examples of its execution.

2. Show that modular ontology engineering results in highly, reusable knowledge graphs by providing real world examples of its execution.

Step 1. Define use case or scope of use cases.

Step 2. Make competency questions while looking at possible data sources and continue scoping the problem and use-case(s).

Step 3. Identify key notions from the data and the use case and identify which pattern should be used for each. Use "stubs" as necessary.

Step 4. Instantiate these key notions from the pattern templates, then adapt the result as needed, to create modules. Develop the remaining modules from scratch.

Step 5. Systematically add axioms for each module.

Step 6. Assemble the modules and add axioms which involve several modules.

Step 7. Reflect on all entity names and possibly improve them. Check module axioms whether they are still appropriate after putting all modules together.

Step 8. Create OWL files.

**Figure 1.3**: *The Modular Ontology Engineering Methodology.*

The outline of the particular topics researched to accomplish these is provided in the next section.

## 1.3 Outline

This document is a cumulative dissertation that details the foundational research towards advancing the modular ontology engineering methodology. As mentioned in the above introduction, this is done to facilitate the development and usage of highly reusable knowledge graphs, in particular, through the use of modular ontologies as their schema. This work can be divided into four concrete research topics that incrementally build towards this goal. The remainder of this dissertation is outlined as follows:

Chapter 2 presents the first research topic: conveying meaning through schema diagrams. It provides the key research questions that this topic addresses, with discussion. The primary contributions referenced in this section are

- *A method for automatically generating schema diagrams for OWL ontologies* [31] and

- *Most of OWL is rarely needed* [5].

Chapter 3 discusses the usage of ontology design patterns and their history and role in the knowledge engineering community, as well as their development, documentation, and usage. Then, we will discuss the state of the art, as it pertains to their use and development. Finally, we present contributions to ontology design pattern community and refer to the respective use-cases and publications. These contributions are as follows:

- *An ontology design pattern for microblog entries* [30]
- *Ontology design patterns for Winston's taxonomy of part-whole relations* [37]
- *Towards a pattern-based ontology for chemical laboratory procedures* [39]
- *A Protégé plugin for annotating OWL ontologies with OPLa* [34]
- *Extensions to the ontology design pattern representation language* [11]
- *MODL: a modular ontology design library* [35]

Chapter 4 focuses on modular ontology engineering as a methodology for developing highly reusable knowledge graphs. First, we will relate it to earlier methodologies: how it differs from and improves upon the state of the art. Then, we present real-world examples of their use, alongside their respective publications. They are:

- *Modular ontologies as a bridge between human conceptualization and data* [17]
- *Modular ontology modeling: a tutorial* [38]
- *The Enslaved Ontology: peoples of the historic slave trade* [36]

Chapter 5 presents the conceptual next step from modular ontology engineering through the use of advanced tooling infrastructure: modular graphical modeling. We present the tool, CoModIDE, and its respective publications, as follows:

- *Towards a comprehensive modular ontology IDE and tool suite* [29]

- *CoModIDE–the Comprehensive Modular Ontology Engineering IDE* [32]
- *Modular Graphical Modeling Evaluated* [33]

Chapter 6 presents concluding remarks through a brief summary that highlights the overall contributions and how they fit together, especially with respect to the state of the art. Additionally, I provide an outlook on immediate, and further, future work.

# Chapter 2

# Conveying Meaning through Schema Diagrams

## 2.1 Overview

Communication between knowledge engineers or ontologists and domain experts is important during the development of an ontology. It is also a key component for determining the reusability of an ontology—an ontology that is poorly documented makes it difficult to understand its commitments and will thus likely not be reused. A popular mechanism for quickly conveying meaning or structure of an ontology is the schema diagram.

**Definition 2.1.1.** A *Schema Diagram* is an informal, but intuitive, graphical representation that depicts how classes in an ontology may be related. Figure 2.1 shows an example of a schema diagram that uses a visual syntax we have adapted for modular ontologies. Orange boxes indicate classes. Dashed blue boxes indicate classes which are part of another module. White-headed arrows indicate subclass relationships, while black arrows are object or data properties. Purple dashed boxes indicate the presence of a controlled vocabulary. Finally, yellow ovals denote datatypes. Titled, gray dashed boxes indicate that the enclosed items are part of a module.

However, schema diagrams are ambiguous by nature. That is, any particular node-edge-

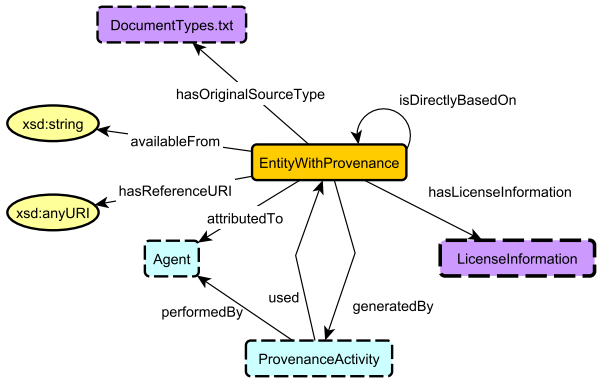



**Figure 2.1**: *The Provenance module for the Enslaved Ontology [36] ontology design pattern. The visual syntax for this schema diagram is detailed in Definition 2.1.1.*

node in the diagram can have one or more axiomatic meanings. Thus, when translating between OWL files containing the formally conceptualized knowledge and the schema diagram (or vice versa) can be an error prone process. As schema diagrams are the primary method of conveying meaning during the modular ontology engineering process, it is important to determine how they may be improved themselves, or by reducing the error rate in the generation process.

For this topic, I formulated the following research questions considering these concerns and are addressed in the next section.

**Q1.** *Can we develop and implement an algorithm for generating schema diagrams similar in form to those manually generated by unguided humans?*

**Q2.** *How can we reduce ambiguity in schema diagrams, but retain their intuitive traits?*

## 2.2 Contributions

This section connects the individual contributions to the above research questions.

[31] Cogan Shimizu, Aaron Eberhart, Nazifa Karima, Quinn Hirt, Adila Krisnadhi, and Pascal Hitzler. A method for automatically generating schema diagrams for OWL ontologies. In Boris Villazón-Terrazas and Yusniel Hidalgo-Delgado, editors, *Knowledge Graphs and Semantic Web - First Iberoamerican Conference, KGSWC 2019, Villa Clara, Cuba, June*

*23-30, 2019, Proceedings*, volume 1029 of *Communications in Computer and Information Science*, pages 149–161. Springer, 2019

This contribution addresses Research Question Q1 and describes such a tool and its use in checking if the formal knowledge in an OWL file mirrors the informal schema diagram through a principle of *schematic equivalence*. The implemented tool was found to produce significantly more accurate schema diagrams than other tools in the state of the art.

The original algorithm for generating a schema diagram was outlined by Nazifa Karima in [20]. In *A method for automatically generating schema diagrams for OWL ontologies* [31], Cogan Shimizu improved the algorithm, re-implemented it, conducted the programmatic evaluation, and analysis. Aaron Eberhart and Quinn Hirt assisted with creating the ground truth dataset from the Ontology Design Patterns portal for a significantly expanded evaluation. Adila Krisnadhi and Pascal Hitzler provided feedback.

[5] Aaron Eberhart, Cogan Shimizu, Sulogna Chowdhury, Md. Kamruzzaman Sarker, and Pascal Hitzler. Most of OWL is rarely needed. In *19th International Semantic Web Conference*, 2020. Under review

In the second publication, *Most of OWL is rarely needed* [5], Aaron Eberhart took the lead in implementation of the analysis software and evaluation. Cogan Shimizu assisted with the software and both he and Pascal Hitzler assisted with the evaluation design and results analysis. Sulogna Chowdury and Md. Kamruzzaman Sarker provided feedback. This contribution begins to address Research Question Q2 in determining how certain simple axioms, strictly those that can be represented using an individual node-edge-node construct in a schema diagram, are already used in existing ontologies. The evaluation was conducted on a sample of 280 ontologies from a variety of domains (e.g. bio-ontology and patterns). The results show that over 90% of axioms in the sample are expressible using a schema diagram. Indeed, the great majority are simple subclass, range, and domain axioms. This analysis will help to make future improvements to our graphical syntax and improve current and future tools that graphically represent axioms, but also reinforces that the schema diagram is a useful tool for expressing the structure and meaning of an ontology, at a broad level.

# Chapter 3

# The Composition of a Modular Ontology

## 3.1 Overview

In order to begin improving the modular ontology engineering methodology, it was important to understand the existing components that comprised a modular ontology. As stated in the introduction, modular ontologies are composed of modules, which are instantiated from ontology design patterns.

**Definition 3.1.1.** An *Ontology Design Pattern* (ODP) is a tiny, self-contained ontology that solves a domain-invariant modeling problem that draw on community-identified best practices.

They draw their inspiration from Software engineering design patterns, such as Factories or Model-View-Controller. Essentially, many different domains have similar conceptualizations for concepts. ODPs leverage this to improve reusability by preventing ontology engineers from "reinventing the wheel" for common modeling tasks. Figure 3.1 shows and example ontology design pattern: the trajectory pattern [18]. The trajectory design pattern aims to model things that move discretely move through a physical or conceptual space.

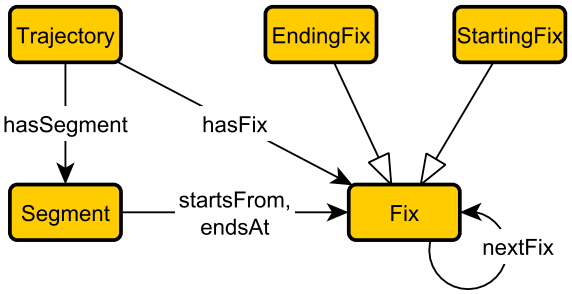



**Figure 3.1**: *The Semantic Trajectory [18] ontology design pattern. The visual syntax for this schema diagram is detailed in Definition 2.1.1.*

As such, the modular ontology engineering methodology promotes the use of ontology design patterns whenever possible. In particular, in Step 3 of the methodology (Figure 1.3), an ontology design pattern should be selected for each identified key notion. Ontology design patterns, however, are rarely immediately in a usable state. This means that they need to be modified in some way, in order to fit what is actually being modeled. This modification process is called template-based instantiation and results in a module.

**Definition 3.1.2.** An *Ontology Module* is a pattern that has been instantiated via template-based instantiation.

Template-based instantiation is the act of replacing class and property names in the general pattern with labels more applicable to the chosen domain [10]. This is a departure from other pattern usage techniques, where a developer may choose to view the ontology design pattern as an upper ontology and use subclass and subproperty axioms to connect the pattern to the domain ontology. Figure 3.2 is an example of this process. The Chemical Laboratory Procedure module (right) has been instantiated by using the State Transition Pattern (left), which in turn is a specialization of the Trajectory pattern (Figure 3.1).

Indeed, tracking this provenance for both modules and patterns is beneficial. In the case of patterns, it allows for attribution or, perhaps, metadata on how to utilize the pattern or the design considerations for it. To annotate patterns and modules as such, the use of the Ontology Design Pattern Representation Language (OPLa) is recommended. An indepth examination of OPLa and it's extensions can be found in [13, 11].

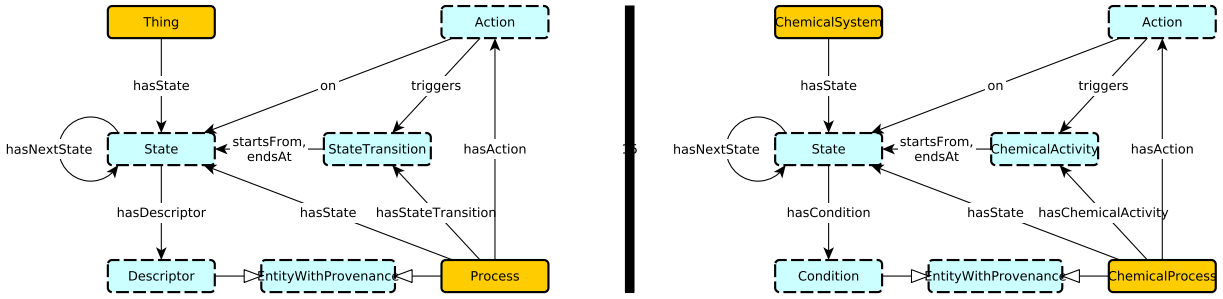



**Figure 3.2**: *The chemical laboratory procedure module (right) [39] next to the State Transition ontology design pattern (left), which is, in turn, a pattern derived from the Trajectory pattern (3.1). The visual syntax for this schema diagram is detailed in Definition 2.1.1.*

Thus, a modular ontology is composed of modules, that patterns from which they were instantiated, and the annotations that interrelate them. As such, this research topic is concerned with the following questions.

**Q3 .** *What are ontology design patterns and how are they developed?*

**Q4 .** *How are ontology design patterns (re)used and how can we improve this process?*

## 3.2 Contributions

The first three contributions, below, all correspond to efforts in answering Research Question Q3. Essentially, in order to understand what ODPs are and how they are developed, it was necessary to work closely with domain experts and attempt to create relevant patterns. These culminated in the below publications. However, and perhaps more importantly, they provided valuable insight into the creation process, which in turn allowed us to advance additional collaborative efforts, such as VoCamps.[1]

To be clear, RQ3 is very broad and can be attempted to be answered in many ways. For the purposes of this dissertation, the corresponding efforts below all lent towards gaining hands on experience in the development of the ODPs. This resulted in many lessons learnt,

[1]See http://vocamp.org/wiki/Main_Page.

both in terms of development and modeling expertise (e.g. how are axioms used and annotated) as well as soft knowledge (e.g. how can formal knowledge be communicated between parties or what are useful ways of iterating over schema diagrams). The former informs the rest of the dissertation, especially the tooling implementations and improvements. The latter is not often mentioned in the below contributions, yet it was a critical aspect for *enabling* downstream research and increasing productivity.

[30] Cogan Shimizu and Michelle Cheatham. An ontology design pattern for microblog entries. In Eva Blomqvist, Óscar Corcho, Matthew Horridge, David Carral, and Rinke Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017

[37] Cogan Shimizu, Pascal Hitzler, and Clare Paul. Ontology design patterns for Winston's taxonomy of part-whole relations. In Elena Demidova, Amrapali Zaveri, and Elena Simperl, editors, *Emerging Topics in Semantic Technologies – ISWC 2018 Satellite Events [best papers from 13 of the workshops co-located with the ISWC 2018 conference]*, volume 36 of *Studies on the Semantic Web*, pages 119–129. IOS Press, 2018

[39] Cogan Shimizu, Leah McEwen, and Quinn Hirt. Towards a pattern-based ontology for chemical laboratory procedures. In Martin G. Skjæveland, Yingjie Hu, Karl Hammar, Vojtech Svátek, and Agnieszka Lawrynowicz, editors, *Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 9th, 2018*, volume 2195 of *CEUR Workshop Proceedings*, pages 40–51. CEUR-WS.org, 2018

In the first publication, *An ontology design pattern for microblog entries* [30], the conceptual modeling was performed by Cogan Shimizu with feedback from Michelle Cheatham. In *Ontology design patterns for Winston's taxonomy of part-whole relations* [37], the conceptual modeling was done by both Cogan Shimizu and Pascal Hitzler. A use-case in the materials science domain was provided by Clare Paul. In the third publication, *Towards a pattern-based ontology for chemical laboratory procedures* [39], the conceptual modeling

was performed by Cogan Shimizu, with significant feedback from Leah McEwen, who also provided the chemical laboratory procedures for explicit use cases. Quinn Hirt provided the implementation and accompanying annotations.

The next two contributions address both research questions. The former is a tool that assists users in annotating their patterns (or modules and ontologies) with annotations from the OPLa. The latter is a contribution that extends the usefulness of OPLa so that discovering users can gain further insight into descriptions, use-cases, and the scenarios that drove the pattern's or module's development.

[34] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. A Protégé plug-in for annotating OWL ontologies with OPLa. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z. Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, volume 11155 of *Lecture Notes in Computer Science*, pages 23–27. Springer, 2018

[11] Quinn Hirt, Cogan Shimizu, and Pascal Hitzler. Extensions to the ontology design pattern representation language. In Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda Villalón, Karl Hammar, and Cogan Shimizu, editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 76–75. CEUR-WS.org, 2019

In the first publication, *A Protégé plug-in for annotating OWL ontologies with OPLa* [34], Cogan Shimizu designed and implemented the annotation software. Quinn Hirt assisted with implementation and Pascal Hitzler provided feedback. The second publication, *Extensions to the ontology design pattern representation language* was authored by Quinn Hirt with feedback from Cogan Shimizu and Pascal Hitzler.

The culmination of this research was the development and release of MODL: A Modular ontology design library. The library is a distributable artifact that contains some of the most frequently used patterns. Their documentation and diagrams were updated with the visual schema diagram syntax from Section 2.1 and were all annotated with OPLa. It also

contains two novel patterns for solving spatial and temporal patterns. Ultimately, the goal is to enhance discovery and trust of ontology design patterns.

[35] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. MODL: A modular ontology design library. In Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda Villalón, Karl Hammar, and Cogan Shimizu, editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 47–58. CEUR-WS.org, 2019

The patterns in *MODL: a modular ontology design library* were collected, curated, and documented by Cogan Shimizu. Quinn Hirt assisted with annotating the patterns. Pascal Hitzler provided feedback.

# Chapter 4

# Modular Ontology Engineering

## 4.1 Overview

After taking the time to learn and understand the component pieces of the methodology, as outlined in the previous chapters, it was then time to execute the methodology in its entirety. The methodology's steps are shown in Figure 1.3. An in-depth examination of the methodology can be found in [38], which is based off of [15, 22]. Briefly, these are some the key attributes.

Modular ontology engineering is use-case driven and assumes an empirical or data reality. This ultimately means that we assume that we have an understanding of how the knowledge graph will be used, as well as what sort of data we will have available (or will be collected) that the knowledge graph will capture.

Modular ontologies are pattern-based, as described in the previous chapter. That is, we generally try to re-use, as frequently as possible, so-called ontology design patterns during our ontology engineering process. By using these patterns as templates, we can create modules, which we then, in turn, hook together to create an Modular ontology.

By examining the process itself, as well as its outcomes, with respect to advancing the methodology, we obtain the following research questions.

**Q5 .** *What steps of the methodology are ambiguous or difficult to follow?*

**Q6 .** *Does the methodology, when executed, produce a reusable knowledge graph?*

## 4.2 Contribution

[17] Pascal Hitzler and Cogan Shimizu. Modular ontologies as a bridge between human conceptualization and data. In Peter Chapman, Dominik Endres, and Nathalie Pernelle, editors, *Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018

[38] Cogan Shimizu, Adila Krisnadhi, and Pascal Hitzler. Modular ontology modeling: a tutorial. In Giuseppe Cota, Marilena Daquino, and Gian Luca Pozzato, editors, *Applications and Practices in Ontology Design, Extraction, and Reasoning*, Studies on the Semantic Web. IOS Press, 2020. Under review

[36] Cogan Shimizu, Pascal Hitzler, Quinn Hirt, Dean Rehberger, Seila Gonzalez Estrecha, Catherine Foley, Alicia M. Sheill, Walter Hawthorne, Jeff Mixter, Ethan Watrall, Ryan Carty, and Duncan Tarr. The Enslaved Ontology: Peoples of the historic slave trade. *Journal of Web Semantics*, 63:100567, 2020

In the first publication, *Modular ontologies as a bridge between human conceptualization and data*, Pascal Hitzler and Cogan Shimizu put forward the position for using modular ontologies as schema for highly reusable knowledge graphs. This contribution frames the answer space for research question Q6.

The second publication, *Modular Ontology Modeling: a Tutorial* [38], was written by Cogan Shimizu based off material and feedback from Adila Krisnadhi and Pascal Hitzler. This contribution addresses Research Question Q5, in particular. By closely examinining each step in the methodology and attempting to present it in a salient and cogent manner, we reduce the ambiguity of the steps, while also providing an in-depth tutorial for new developers to follow.

In the third publication, *The Enslaved Ontology: Peoples of the historic slave trade* [36], Cogan Shimizu drove the conceptual modeling, with significant feedback from Pascal Hitzler.

Quinn Hirt provided documentation and annotation assistance. The following authors: Dean Rehberger, Seila Gonzalez Estrecha, Catherine Foley, Alicia M. Sheill, Walter Hawthorne, Ethan Watrall, Ryan Carty, and Duncan Tarr served as the domain experts, which were close collaborators for the document. Jeff Mixter consulted on infrastructure and content. This contribution describes a full implementation of the modular ontology methodology that resulted in a modular ontology that is used in the real-world; in post-publication, it has been found to be adaptable, extensible, and reusable. Since its conception, we have further worked to extend its provenance module and evolve its place module, which will result in forthcoming version 2.0 documents.[1] It is also being successfully adapted to the WikiBase schema for the Enslaved Hub knowledge graph. This contribution fully addresses Research Question Q6.

[1]See https://enslaved.org/docs/.

# Chapter 5

# Modular Graphical Modeling

## 5.1 Overview

Modular graphical modeling is the conceptual next step with respect to the modular ontology engineering methodology. By realizing the potential of the methodology through increased tool support, we posit that we can improve outcomes for both experienced and new ontology developers. That is, we ask the following research questions.

**Q7 .** *Does the use of graphical tools improve outcomes of modular ontology development?*

**Q8 .** *Does modular graphical modeling improve accessibility to modular ontology engineering?*

This effort culminated in a tool called the Comprehensive Modular Ontology Design IDE, which is a plugin for an industry standard ontology editor, Protegé. Figure 5.1 shows a screenshot of version 1.0 of the tool. CoModide was initially described in [29] and the preliminary implementation results are presented in [29].

Through the use CoModIDE, we did find that our hypothesis held true, after controlling for a number of variables, thus addressing Research Questions Q7 and Q8. A thorough analysis and comparison to the state of the art, description of the experiment design, with discussion of the results can be found in the contribution [33].

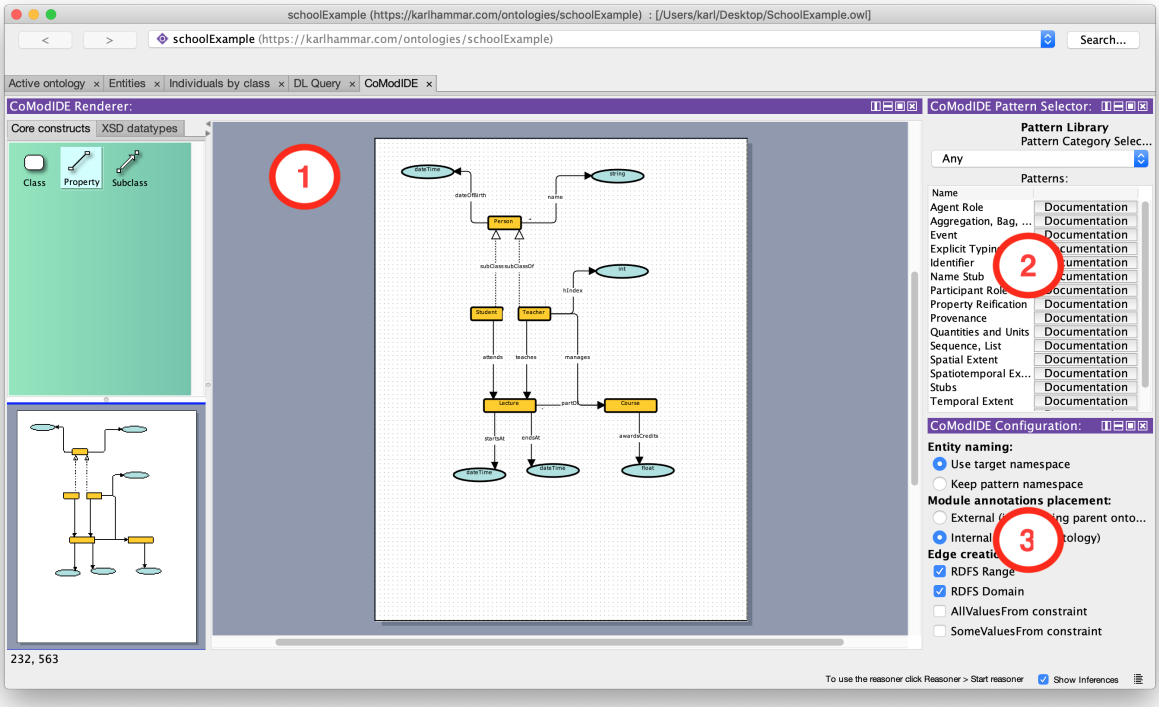



**Figure 5.1**: *A screenshot of CoModIDE. The area marked (1) is the palette where the visual primitives may be selected. (2) is the graphical canvas where developers may construct schema diagrams. (3) is a pattern library from which a user may drag-and-drop ODPs from MODL onto the graphical canvas.*

## 5.2 Contribution

[29] Cogan Shimizu. Towards a comprehensive modular ontology IDE and tool suite. In Sabrina Kirrane and Lalana Kagal, editors, *Proceedings of the Doctoral Consortium at ISWC 2018 co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th–12th, 2018*, volume 2181 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2018

[32] Cogan Shimizu and Karl Hammar. CoModIDE – the Comprehensive Modular Ontology Engineering IDE. In Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet, and Abraham Bernstein, editors, *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland,*

*New Zealand, October 26-30, 2019*, volume 2456 of *CEUR Workshop Proceedings*, pages 249–252. CEUR-WS.org, 2019

[33] Cogan Shimizu, Karl Hammar, and Pascal Hitzler. Modular graphical ontology engineering evaluated. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web – 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2020

In the first publication, *Towards a comprehensive modular ontology IDE and tool suite* [29], Cogan Shimizu puts forward a comprehensive description of the CoModIDE vision. As it was a doctoral consortium paper, it necessitated a single author requirement of the PhD candidate. The second publication, *CoModIDE – the comprehensive modular ontology engineering IDE* [32], is a demonstration paper that put forward the first version of CoModIDE. Cogan Shimizu and Karl Hammar co-developed the software platform.

In this final publication, *Modular graphical modeling evaluated* [33], CoModIDE and the methodology were evaluated. Cogan Shimizu designed the experiment, conducted a local instance of the experiment, and provided the experiment analysis. Karl Hammar provided related work, software description, and also conducted a local experiment. Pascal Hitzler provided feedback over the course of the work, in particular, the experiment design.

# Chapter 6

# Conclusion

## 6.1 Summary

Modular ontology engineering is a methodology for producing highly reusable knowledge graph schema. Over the course of this dissertation, we have outlined a number of contributions that have improved the process to what we see today.

First, we created an improved method and tool for generating schema diagrams similar to those manually generated by humans [31]. We furthermore show that over 80% of all axioms in a set of 280 real-world ontologies are expressible using this graphical representation.

Next, we examined and improved the ontology design pattern development process. This was accomplished through the development of relevant ODPs [30, 39, 37]. We furthermore developed extensions to OPLa [11] and a tool that significantly improves the usability of these annotations [34]. This work culminated in MODL: a modular ontology design library, which is a distributable set of curated, well-documented ODPs, both novel and drawn from the ontology design pattern portal [35].

These advances were combined, with inspiration from [28], to create the Comprehensive Modular Ontology Design IDE (CoModIDE), which is a plugin for the industry-standard ontology editor, Protégé [29, 32]. This tool was shown to significantly improve outcomes for experienced and new ontology developers when developing modular ontologies [33].

Altogether, these research topics, resulted in result in an actually reusable, extendable, and adaptable ontologies, in the forms of the The Enslaved Ontology [36] and the Domain Ontology for Task Instructions [6].

## 6.2 Future Work

There are two immediate next steps to address.

1. How can modular ontologies be leveraged in other subfields of knowledge representation and reasoning?

2. How can CoModIDE be improved as both a research and development platform?

Two examples of how modular ontologies may be leveraged elsewhere are the tasks of complex ontology alignment and ontology learning.

Ontology alignment is the science of finding correspondences between two different ontologies that may be conceptualizing the same concepts, albeit differently [7]. Complex ontology alignment (COA) is an extension of that, but examining more complex correspondences than, e.g., string matching concept names [44]. We have reason to believe that the modular structure of a modular ontology could assist in this task. By leveraging the embedded patterning information, we have more information on how individual axioms contribute to a module, thus providing insight to a COA system on possible complex correspondences.

With respect to ontology learning, we envision a more principled approach to pattern-mediated ontology learning. Our advances in understanding patterns and modules coupled with recent advances in natural language processing, we have reason to believe that new ground can be broken.

For Question 2, CoModIDE has potential to spur additional development of tooling infrastructure for modular ontology development. Recent updates to the software have seen the inclusion of telemetry software, which can be used to improve user-workflows, as well as more rapid detection and fixing of bugs in the platform. Furthermore, it has its own

underlying message bus that other developers may code against. Thus, CoModIDE is open to be improved or enhanced through the efforts of others outside of the current developers. For example, we foresee collaboration with different pattern communities (e.g. OTTR [40]) or other tools (e.g. ODPReco [43]).

All in all, modular ontologies have a healthy outlook and can have a considerable impact on the state of the art, moving forward.

# Bibliography


[1] Thomas Baker and Eric Prud'hommeaux, editors. *Shape Expressions (ShEx) 2.1 Primer*. Final Community Group Report 09 October 2019, 2019. http://shex.io/shex-primer/index.html.

[2] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data – the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[3] Eva Blomqvist, Karl Hammar, and Valentina Presutti. Engineering ontologies with patterns – the eXtreme Design methodology. In Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina 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.

[4] Richard Cyganiak, David Wood, and Markus Lanthaler, editors. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation 25 February 2014, 2014. Available from http://www.w3.org/TR/rdf11-concepts/.

[5] Aaron Eberhart, Cogan Shimizu, Sulogna Chowdhury, Md. Kamruzzaman Sarker, and Pascal Hitzler. Most of OWL is rarely needed. In *19th International Semantic Web Conference*, 2020. Under review.

[6] Aaron Eberhart, Cogan Shimizu, Christopher Stevens, Pascal Hitzler, Christopher W. Meyers, and Benji Maruyama. A domain ontology for task instructions. In *Knowledge Graphs and Semantic Web - Second Iberoamerican Conference, KGSWC 2020*, Communications in Computer and Information Science. Springer, 2020. Under review.

[7] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013.

[8] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[9] Ramanathan V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: evolution of structured data on the web. *Commun. ACM*, 59(2):44–51, 2016.

[10] Karl Hammar and Valentina Presutti. Template-based content ODP instantiation. In Karl Hammar, Pascal Hitzler, Adila Krisnadhi, Agnieszka Lawrynowicz, Andrea Giovanni Nuzzolese, and Monika Solanki, editors, *Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the 7th edition of the Workshop on Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, 18th October 2016]*, volume 32 of *Studies on the Semantic Web*, pages 1–13. IOS Press, 2016.

[11] Quinn Hirt, Cogan Shimizu, and Pascal Hitzler. Extensions to the ontology design pattern representation language. In Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda Villalón, Karl Hammar, and Cogan Shimizu, editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 76–75. CEUR-WS.org, 2019.

[12] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti, editors. *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*. IOS Press, 2016.

[13] Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Alfa Krisnadhi, and Valentina Presutti. Towards a simple but useful ontology design pattern representation language. In Eva Blomqvist, Óscar Corcho, Matthew Horridge, David Carral, and Rinke Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017),*

*Vienna, Austria, October 21, 2017.*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[14] Pascal Hitzler and Adila Krisnadhi. On the roles of logical axiomatizations for ontologies. In Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti, editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 73–80. IOS Press, 2016.

[15] Pascal Hitzler and Adila Krisnadhi. A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. *CoRR*, abs/1808.08433, 2018.

[16] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation 11 December 2012, 2012. Available from http://www.w3.org/TR/owl2-primer/.

[17] Pascal Hitzler and Cogan Shimizu. Modular ontologies as a bridge between human conceptualization and data. In Peter Chapman, Dominik Endres, and Nathalie Pernelle, editors, *Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018.

[18] Yingjie Hu, Krzysztof Janowicz, David Carral, Simon Scheider, Werner Kuhn, Gary Berg-Cross, Pascal Hitzler, and Mike Dean. A geo-ontology design pattern for semantic trajectories. In Thora Tenbrink, John G. Stell, Antony Galton, and Zena Wood, editors, *Spatial Information Theory – 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings*, volume 8116 of *Lecture Notes in Computer Science*, pages 438–456, Heidelberg, 2013. Springer.

[19] Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheth. Linked data is merely more data. In *Linked Data Meets Artificial Intelligence, Papers from the*

*2010 AAAI Spring Symposium, Technical Report SS-10-07, Stanford, California, USA, March 22-24, 2010*. AAAI, 2010.

[20] Nazifa Karima. Automated rendering of schema diagram for ontologies. Master's thesis, Wright State University, 2017.

[21] Holger Knublauch and Dimitris Kontokostas, editors. *Shapes Constraint Language (SHACL)*. W3C Recommendation 20 July 2017, 2017. https://www.w3.org/TR/shacl/.

[22] Adila Krisnadhi and Pascal Hitzler. Modeling with ontology design patterns: Chess games as a worked example. In Pascal Hitzler, Aldo Gangemi, Krzysztof Janowicz, Adila Krisnadhi, and Valentina Presutti, editors, *Ontology Engineering with Ontology Design Patterns – Foundations and Applications*, volume 25 of *Studies on the Semantic Web*, pages 3–21. IOS Press, 2016.

[23] LOD Laundromat. http://lodlaundromat.org/. Accessed November 7, 2017.

[24] http://lod-cloud.net/. Retrieved October 26, 2016.

[25] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.

[26] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. eXtreme Design with content ontology design patterns. In Eva Blomqvist, Kurt Sandkuhl, François Scharffe, and Vojtech Svátek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009) , collocated with the 8th International Semantic Web Conference ( ISWC-2009 ), Washington D.C., USA, 25 October, 2009.*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[27] Víctor Rodríguez-Doncel, Adila Alfa Krisnadhi, Pascal Hitzler, Michelle Cheatham, Nazifa Karima, and Reihaneh Amini. Pattern-based linked data publication: The

linked chess dataset case. In Olaf Hartig, Juan F. Sequeda, and Aidan Hogan, editors, *Proceedings of the 6th International Workshop on Consuming Linked Data co-located with 14th International Semantic Web Conference (ISWC 2105), Bethlehem, Pennsylvania, USA, October 12th, 2015.*, volume 1426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[28] Md. Kamruzzaman Sarker, Adila Alfa Krisnadhi, and Pascal Hitzler. OWLAx: A Protégé plugin to support ontology axiomatization through diagramming. In Takahiro Kawamura and Heiko 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*, 2016.

[29] Cogan Shimizu. Towards a comprehensive modular ontology IDE and tool suite. In Sabrina Kirrane and Lalana Kagal, editors, *Proceedings of the Doctoral Consortium at ISWC 2018 co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th–12th, 2018*, volume 2181 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2018.

[30] Cogan Shimizu and Michelle Cheatham. An ontology design pattern for microblog entries. In Eva Blomqvist, Óscar Corcho, Matthew Horridge, David Carral, and Rinke Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[31] Cogan Shimizu, Aaron Eberhart, Nazifa Karima, Quinn Hirt, Adila Krisnadhi, and Pascal Hitzler. A method for automatically generating schema diagrams for OWL ontologies. In Boris Villazón-Terrazas and Yusniel Hidalgo-Delgado, editors, *Knowledge Graphs and Semantic Web - First Iberoamerican Conference, KGSWC 2019, Villa*

*Clara, Cuba, June 23-30, 2019, Proceedings*, volume 1029 of *Communications in Computer and Information Science*, pages 149–161. Springer, 2019.

[32] Cogan Shimizu and Karl Hammar. CoModIDE – the Comprehensive Modular Ontology Engineering IDE. In Mari Carmen Suárez-Figueroa, Gong Cheng, Anna Lisa Gentile, Christophe Guéret, C. Maria Keet, and Abraham Bernstein, editors, *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*, volume 2456 of *CEUR Workshop Proceedings*, pages 249–252. CEUR-WS.org, 2019.

[33] Cogan Shimizu, Karl Hammar, and Pascal Hitzler. Modular graphical ontology engineering evaluated. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web – 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2020.

[34] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. A Protégé plug-in for annotating OWL ontologies with OPLa. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z. Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, volume 11155 of *Lecture Notes in Computer Science*, pages 23–27. Springer, 2018.

[35] Cogan Shimizu, Quinn Hirt, and Pascal Hitzler. MODL: A modular ontology design library. In Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda Villalón, Karl Hammar, and Cogan Shimizu, editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 47–58. CEUR-WS.org, 2019.

[36] Cogan Shimizu, Pascal Hitzler, Quinn Hirt, Dean Rehberger, Seila Gonzalez Estrecha, Catherine Foley, Alicia M. Sheill, Walter Hawthorne, Jeff Mixter, Ethan Watrall, Ryan Carty, and Duncan Tarr. The Enslaved Ontology: Peoples of the historic slave trade. *Journal of Web Semantics*, 63:100567, 2020.

[37] Cogan Shimizu, Pascal Hitzler, and Clare Paul. Ontology design patterns for Winston's taxonomy of part-whole relations. In Elena Demidova, Amrapali Zaveri, and Elena Simperl, editors, *Emerging Topics in Semantic Technologies – ISWC 2018 Satellite Events [best papers from 13 of the workshops co-located with the ISWC 2018 conference]*, volume 36 of *Studies on the Semantic Web*, pages 119–129. IOS Press, 2018.

[38] Cogan Shimizu, Adila Krisnadhi, and Pascal Hitzler. Modular ontology modeling: a tutorial. In Giuseppe Cota, Marilena Daquino, and Gian Luca Pozzato, editors, *Applications and Practices in Ontology Design, Extraction, and Reasoning*, Studies on the Semantic Web. IOS Press, 2020. Under review.

[39] Cogan Shimizu, Leah McEwen, and Quinn Hirt. Towards a pattern-based ontology for chemical laboratory procedures. In Martin G. Skjæveland, Yingjie Hu, Karl Hammar, Vojtech Svátek, and Agnieszka Lawrynowicz, editors, *Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 9th, 2018*, volume 2195 of *CEUR Workshop Proceedings*, pages 40–51. CEUR-WS.org, 2018.

[40] Martin G. Skjæveland, Daniel P. Lupp, Leif Harald Karlsen, and Henrik Forssell. Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018 – 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2018.

[41] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The neon methodology framework: A scenario-based methodology for ontology development. *Applied Ontology*, 10(2):107–145, 2015.

[42] Mark D. Wilkinson, Michel Dumontier, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3, 2016.

[43] Maleeha Arif Yasvi and Raghava Mutharaju. Odpreco - A tool to recommend ontology design patterns. In Krzysztof Janowicz, Adila Alfa Krisnadhi, María Poveda Villalón, Karl Hammar, and Cogan Shimizu, editors, *Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019*, volume 2459 of *CEUR Workshop Proceedings*, pages 71–75. CEUR-WS.org, 2019.

[44] Lu Zhou, Élodie Thiéblin, Michelle Cheatham, Daniel Faria, Catia Pesquita, Cássia Trojahn dos Santos, and Ondrej Zamazal. Towards evaluating complex ontology alignments. *Knowledge Eng. Review*, 35:e21, 2020.

# Appendix A

# Contributions

In the following pages, all contributions made for this dissertation are listed in order of appearance in this document.

# A Method for Automatically Generating Schema Diagrams for Modular Ontologies

Cogan Shimizu[1], Aaron Eberhart[1], Nazifa Karima[1], Adila Krisnadhi[2], and Pascal Hitzler[1]

[1] Data Semantics Laboratory, Wright State University, Dayton, OH, USA
[2] Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

**Abstract.** Interest in Semantic Web technologies, including knowledge graphs and ontologies, is increasing rapidly in industry and academics. Developing large onlologies is a difficult task that is often aided by modular development practices. Our method for generating schema diagrams supports ontology engineers working with this strategy. In order to evaluate this method, we provide a prototype tool, SDOnt, and examine its ability to generate schema diagrams similar to manually drawn schema diagrams and show that it outperforms the popular tools VOWL and OWLGrEd.

## 1 Introduction

Modular ontology development presents unique challenges to visualization that commonly-used software does not satisfy. Along with other types small ontologies, like ontology design patterns, the information necessary for understanding these models is often a fraction of what is required for a larger ontology with a complete diagram. Sometimes the over-inclusion of information can lead to densely unreadable or even subtly incorrect representations. Modular ontologies are and often changed, reworked, or otherwise re-positioned in reference to static superstructure during development. These and other reasons demonstrate that a different type of visualization is necessary to adequately capture the fundamental semantics of modular ontologies.

We present a novel algorithm that is designed to satisfy the specific requirements of modular ontologies. In particular, we show that in modular development often the most effective representation strategy is that of a schema diagram. A schema diagram is a widespread and invaluable tool for both understanding and developing ontologies. A survey we conducted shows that it ranks among the most important components in the documentation of an ontology [11]. Schema diagrams provide a view, albeit limited, of the structure of the relationships between concepts of an ontology. Frequently, a schema diagram is generated manually during the design phase of the engineering process. At that time, the diagram is a mutable, living document. After the schema diagram has been created, the OWL file is created in the likeness of the diagram by means of OWL axioms which precisely capture the underlying intention of the possibly

ambiguous diagram. We may call this a *diagram-informed OWL file*. As we see in Section 5, this can lead to unforeseen problems, such as whether the OWL file truly represents the diagram. Thus, one possible, beneficial side effect of having a tool that can generate a schema diagram programmatically is that it allows ontology engineers to create an *OWL-informed diagram*. Additionally, it would provide a mechanism by which schema diagrams may be easily updated in the case of a newer versioned OWL file.

In this paper, we demonstrate our technique for automatically generating schema diagrams that are most helpful for development according to the strategies we outline. Next we evaluate our strategy in comparison with author-drawn diagrams as well as the popular OWL visualization tools WebVOWL and OWLGrEd. For the limited scope of this paper, we focus only on evaluations of semantic content and avoid discussions of layout, position, size, aesthetics, etc. These are important concerns which we intend to explore in future work.

The rest of the paper proceeds as follows. Section 2 describes existing visualization tools and how they differ from our method and tool. Section 3 describes the process of schema diagram generation. Section 4 gives a very brief description of our implementation and our method. Section 5 evaluates our method, details possible points of improvement, and discusses the results. Finally, in Section 6 we conclude and outline our next steps and future work.

## 2 Related Work

Visualization is a critical aspect to understanding the purpose (and content) of an ontology [4, 5, 11]. There are many tools that offer visualization capabilities. We are interested in how meaningfully they construct a visualization, rather than the details of their implementations. For example, many of these tools offer some sort of interactivity, such as drag and drop construction and manipulation or folding for dynamic exploration. This differs from our intent: to provide a method for constructing a diagram that portrays the relations between concepts. Additionally, our approach does not provide specific support for visualizing an ABox, as our emphasis is on supporting the creation and use of schemas.

Below, we have selected for comparison a few tools that are representative in their functionality. For a more complete survey, see [6]. As we have chosen VOWL (as implemented using the WebVOWL tool) and OWLGrEd for direct evaluation against our method, we describe them in Section 5.

*NavigOWL*[3] is a plugin for the popular tool Protégé[4]. NavigOWL provides a graph representation of the loaded ontology, such that the representation follows a power-law distribution, which is a type of force-directed graph representation. It also provides a mechanism for filtering out different relational edges while exploring an ontology [1]. This tool is not supported in the current version of

[3] `http://home.deib.polimi.it/hussain/navigowl/`
[4] `https://protege.stanford.edu/`

Protégé.[5] It is particularly well suited to visualizing the ABox, which is outside the scope of our intent and method.

*OWLviz* is also a Protégé plugin. It generates an IS-A hierarchy for the loaded ontology rooted with the concept owl:Thing. That is, OWLviz displays *only* subclass relations between concepts and does not extract properties from those axioms. Hovering over the nodes in the graph representation provides axioms related to the class represented by that node. This plugin is not supported by the current version of Protégé. The lack of relational specificity per edge is non-ideal for our purposes. Furthermore, information accessible only through interactivity is non-ideal for a reference diagram.

*TopBraid Composer* is a standalone tool similar in functionality to Protégé if it were augmented with OWLviz; it is developed, maintained, and sold by TopQuadrant, Inc.[6] There is no version for academic purposes.

*OntoTrack* is a standalone tool for visualizing the subsumption hierarchy of an ontology rooted at owl:Thing. Properties are not extracted from axioms and used to label edges. Further, the tool only supports ontologies in the deprecated OWL-Lite$^-$ and automatically augments the visualization with subsumptions found with the reasoner RACER[7] [13]. Between the limitations on OWL and the interactivity, this tool is not strictly suitable for creating schema diagrams.

*MEMO GRAPH* was developed to be a memory prosthesis for users suffering from dementia [6]. As such, it is particularly focused on representing the relations between family members. It is not currently available for public use.

*RDF Gravity* is a standalone tool that provides a visualization for an ontology via graph metrics. The tool generates a force-directed graph representation of the underlying ontology. We could not find any data on how it handles blank nodes, represents class disjointness, and other non-graph metrics, as, at time of this writing, the tool is unavailable, no publication on its method can be found, and it seems to be survived only by screenshots. We include this entry for the sake of completeness. In these, we note that there are many tools for visualizing ontologies, however many are no longer supported or only provide limited support past graph-metric data.

Finally, we discuss the strategy as outlined in [18]. The authors describe a methodology similar in process and result to ours presented in the next section. Their method creates a so-called *navigation graph* by projecting an ontology into a graph-space. The navigation graph is a labelled graph that they purport to aid in end user understanding. This work was done in parallel to ours [10] and we were not aware of their line of work. While their proposed technique is very similar to ours, as detailed in the next section, they did not provide an implementation or evaluation of their method.

[5] `https://protegewiki.stanford.edu/wiki/NavigOWL`
[6] `https://www.topquadrant.com/products/`
[7] `https://www.ifis.uni-luebeck.de/index.php?id=385`

## 3 Method

A schema diagram does not necessarily aim to represent all information encoded in an ontology. As mentioned in Section 2, there are several tools that attempt to do so, in particular, VOWL and OWLGrEd. However, our ontology modeling experiences, with domain experts across wildly different fields, show that it is necessary to strike a good balance between complexity and understandability.

In fact, in these collaborations we gravitate towards diagrams that merely capture classes and the relationships between them. This omits most semantic aspects such as whether a relationship between classes does or does not indicate domain or range restrictions or even more complex logical axioms. We find that the exact semantics are better conveyed using either natural language sentences or logical axioms (preferably in the form of rules [16]) in conjunction with a very simplified diagram.

Typically, we create modular ontologies by first drawing schema diagrams with domain experts and then capturing the exact logical axioms that constitute the ontology. In this paper we reverse the process: start with the logical axioms and automatically derive the schema diagrams. We do this to help us to deal with ontologies constructed by others for which no suitable schema diagrams are provided. As we will see later in Section 5, our visualization approach can also be helpful in finding errors in OWL files or in manually drawn schema diagrams.

To maximize information and minimize clutter we define some guidelines:

- All classes inherit from owl:Thing, so it is unhelpful to clutter a diagram with subclass edges from concepts to owl:Thing.
- We do not represent any logical connectives or complex axioms, other than direct subClass relationships between named classes, since in our experience this type of information is better conveyed non-visually.
- Disjointness of classes does not need explicit graphical representation. In most cases, disjointness is immediately clear for a human with some knowledge about the domain.
- Inverse relations are not represented, as they are syntactic sugar for any relation.
- The ABox is disregarded; instances of classes are not represented.

With these assumptions in mind, we detail our method using the rules below:

Steps 3 and 4 may be omitted if there are no direct domain or range restrictions given. However, because there are multiple ways of expressing the same information in OWL, domain and range may appear in the declarations of the Object or the Datatype Properties.

In Step 5 it is important to note the differences between logical and schematic equivalence. *Schematic equivalence* between two ontologies means they share the same graphical representation. Consider, for example, the definitions for scoped domain and range restrictions:

$$\exists R.B \sqsubseteq A \tag{1}$$

$$A \sqsubseteq \forall R.B \tag{2}$$

1. Create a node for each class in the ontology's signature.
2. Create a node for each datatype in the ontology's signature.
3. Generate a directed edge for each Object Property based on its domain and range restrictions, if such are given. The source of the edge is the Property's domain and the target of the edge is the Property's range.
4. Generate a directed edge for each Datatype Property, in the same manner as for an Object Property, if domain and range restrictions are present.
5. For each other axiom in the TBox:
   - Case 1: if the subclass and superclass are atomic, generate a subclass edge between them.
   - Case 2: if the axiom is of the forms presented in (1) and (2) below, generate the associated directed edge.
   - Case 3: apply rules (3) to (6), as listed below, recursively until the resulting axiom sets can be handled by Cases 1 and 2.
6. Display.

Fig. 1: The algorithm for generating a schema diagram.

Logically, (1) and (2) convey two different meanings. Schematically, though, they may be represented by the same artifact in a graph: $A \xrightarrow{R} B$. Thus we consider them schematically equivalent. We may also break down more complex axioms using the rules defined in (3) through (6). These rules hold for both intersection ($\sqcap$) and union ($\sqcup$). We list only the union versions. Note that not all of these are logically equivalent transformations.

$$A \sqsubseteq \forall R.(B \sqcup C \sqcup \cdots) \Rightarrow \begin{cases} A \sqsubseteq \forall R.B \\ A \sqsubseteq \forall R.C \\ \quad \vdots \end{cases} \tag{3}$$

and

$$\exists R.(B \sqcup C \sqcup \cdots) \sqsubseteq A \Rightarrow \begin{cases} \exists R.B \sqsubseteq A \\ \exists R.C \sqsubseteq A \\ \quad \vdots \end{cases} \tag{4}$$

(5) and (6) are used only in the union case as shown here. (5) is not a logically equivalent transformation.

$$B \sqcup C \sqcup \cdots \sqsubseteq A \Rightarrow \begin{cases} B \sqsubseteq A \\ C \sqsubseteq A \\ \vdots \end{cases} \tag{5}$$

$$A \sqsubseteq B \sqcup C \sqcup \cdots \Rightarrow \begin{cases} A \sqsubseteq B \\ A \sqsubseteq C \\ \vdots \end{cases} \tag{6}$$

We may recursively apply (3) through (6) for non-atomic concepts $A, B, \cdots$ until we have reached axioms of the form (1) and (2) or atomic subclass relationships.

The time complexity for this process is minimal. If $c$ is the maximum number of concepts and datatypes in any axiom in the ontology then there are at most $\binom{c}{2}$ so-called "simple" axioms that together are schematically equivalent to the "complex axiom." Thus, there are at most $\binom{c}{2} \cdot n$ edges to parse per ontology, where $n$ is the number of axioms in the TBox, giving our method a time complexity of $O(n)$. This calculation ignores algorithms for the graph layout.

Because the goal of our approach is to generate static schema diagrams, there are practical limitations on the size of an ontology the program uses. Any schema diagram becomes essentially unreadable if it gets too large. Indeed, our approach is primarily meant for smaller OWL files such as ontology design patterns [7] or ontology modules [12]. Larger projects would first have to be broken down into modules to create separate smaller schema diagrams. In fact, this would likely result in a better engineered ontology [9].

## 4 Implementation

For the purposes of our evaluation, we have developed a prototype implementation. SDOnt is a three-part pipeline consisting of a GUI, a parser module, and a rendering module. SDOnt is developed in Java and provided as an executable JAR file. Ontology manipulations are done using the OWLAPI.[8] We provide the source code for SDOnt online, along with our test set, evaluation results, and a brief tutorial for the tool's use.[9]

The GUI is implemented using Java Swing and serves as an interface for navigating and loading ontologies into the program. The Ontology Parser is an implementation of our algorithm as described in Section 3. The parser provides a set of nodes to the rendering module that represent the classes and datatypes in the ontology's signature and the node-edge-node artifacts that represent their properties, domains, and ranges for the visualization.

[8] `https://github.com/owlcs/owlapi`
[9] `http://dase.cs.wright.edu/content/sdont`

The rendering module takes those node-edge-node artifacts and the node set and combines them to create the visualization and render it to the screen. The rendering module utilizes the library JGraphX[10] for generating, laying out, and displaying the schema diagram. JGraphX is an open source library written in Java for displaying and manipulating graphs. However, the SDOnt code-base has been written in such a way that any visualization library may be used; that is, an external developer may code against the SDOnt code-base with no changes necessary to the method.

## 5 Evaluation

In this evaluation we describe the closest alternatives to SDOnt and their methods in Section 5.1, the method by which we conduct our evaluation in Section 5.2, our choice of test set in Section 5.3, and discuss the results of our evaluation in Section 5.4.

### 5.1 Compared Tools

We evaluate SDOnt by comparison with author supplied visualizations, WebVOWL, and OWLGrEd.

Each of the ontologies in our test set, which is outlined in Section 5.3, has a general visualization provided by the authors. We use these diagrams as a baseline against which the tools can be judged and assume that these represent the authors' best attempt to generalize the semantics of the ontology. However, there may be some irregularities in the methodologies different authors use to produce these visualizations. Indeed, some of the methodologies are very distinct–some are very minimal; others take inspiration from UML. Some of the diagrams appear to be created automatically from Protégéor some other automated tool, while others are manually draw using a variety of graphing utilities. The variations in these sources may be partially responsible for suboptimal results, especially as none of the three compared tools use a UML-style visualization.

VOWL is a graphical notation tool for OWL. The specification can be viewed in detail in [14]. VOWL represents ontologies using detailed force-directed graphs. We use the web implementation WebVOWL[11] to generate visualizations of our ontologies. The website application has a high degree of potential customization. In the settings, we choose to filter out only class disjointness axioms and set the degree of collapsing to 0 since we use smaller ontologies.

WebVOWL is able to quickly produce a visualization for every valid OWL file that we analyze. Usually the output animation clearly represents the intent of the ontology. Occasionally, however, the result contains incomplete or missing information. In our experience WebVOWL is often ambiguous when it tries to display complex statements, which could be the cause of this.

[10] `https://github.com/jgraph/jgraphx`
[11] `http://www.visualdataweb.de/webvowl/`

OWLGrEd is a Graphical Ontology Editor that allows for interactive, drag-and-drop creation of ontologies [2]. It utilizes UML-like visualizations for displaying axioms associated to a class. In addition, it provides Manchester Syntax translations of axioms. OWLGrEd displays *all* axioms, sometimes as additional nodes. The visualization is hierarchical, so there is a subClass edge between an owl:Thing node and every un-subsumed concept in the ontology's signature. At the time of writing the web version of the OWLGrEd used in our study is not loading, though the desktop application is still available.

### 5.2 Comparison Scheme

The method for constructing schema diagrams for ontology patterns and modules, as introduced in the previous section, results in a diagram most similar to published reference diagrams which follow the visualization paradigm which we found most useful in interactive modeling sessions with domain experts. In order to provide a meaningful evaluation, we use as gold-standard reference the manually drawn diagrams which have been published in the papers or on the websites where the corresponding ontologies have been discussed by their authors. I.e., these diagrams have been designed with human understandability in mind, and their creation pre-dates our automated diagram generation method.

We will compare the diagrams generated by SDOnt, VOWL and OWLGrEd with the gold-standard diagrams taken from the respective publications.

In order to have some useful terminology, we say a *node* represents a class or concept. An *edge* represents a relationship or role, where the source of the edge is the relationship's domain and the head of the edge represents the relationship's codomain – domain and codomain are here not meant to be formal technical terms in the sense of OWL restrictions or RDFS domain/range declarations, but rather intuitive notions which are as ambiguous as a schema diagram: An edge from class A to class B in the diagram indicates that A is (informally) in the domain of the relation, and B is in the codomain of the relation. However, there may also be an edge with the same role label between two different classes C and D elsewhere in the diagram, without making the classes A and C (or B and D) identical, as would happen if these were formal domain or range declarations.

All three visualization tools generate directed edges. To conduct this comparison, we evaluate the following criterion for node-edge-node artifacts:

> For every node-edge-node artifact in the generated diagram, does it appear in the reference diagram, and vice-versa?

We state the results of each comparison as an $F_1$-score in Table **??** using the criteria below.

- True Positive: the artifact appears in both generated and reference diagrams
- False Positive: the artifact appears in the generated diagram, but not the reference diagram.

– False Negative: the artifact does not appear in the generated diagram, but does appear in the reference diagram.

### 5.3 Test Set

In order to test our process, we constructed a test set of ontology design patterns. Our process for selecting the patterns was simply searching the main publishing outlets and `ontologydesignpatterns.org` and choosing those patterns that had published diagrams, as well as unbroken links to their OWL files. All test data and the complete set of ontologies we used for our evaluation can be found on our tool's website.[12]

In some cases, it was necessary to make minor changes to the OWL files during this evaluation. Both OWLGrEd and WebVOWL produced errors on importing certain external resources.[13] Whenever removing an import allowed us to continue with the analysis we did so, using the new OWL file for both tools, even if the file worked for the others tools initially. However, there were still some patterns that failed to work for any tool. Our results report only on those OWL files that could be successfully processed by each of the compared tools. After these criteria were met, our test set contains 63 ontology design patterns.

### 5.4 Results & Discussion

The results of our evaluation are summarized in Tables 1 and 2. The node-edge-node triple sets are determined manually. During this process we take care to use a consistent naming format. This allows us to conduct our comparison programmatically. The code utilized for this comparison, with some documentation for its use, is also available in the online portal.

We see that in Table 1, the $F_1$–measures are very low. SDOnt has an $F_1$–measure of 0.465, OWLGrEd has an $F_1$–measure of 0.269, and WebVOWL has an $F_1$–measure of 0.163. However, we also note that there are many stylistic differences in the generation of diagrams. Many of the reference diagrams are presented in UML-esque manner. Comparing these to a force directed graph, such as those produced by WebVOWL would, of course, perform very poorly.

In this evaluation we do not encounter any false positives that are a misrepresentation of an axiom. Instead, false positives are strictly caused by the OWL file containing more information than expected. The exact reasons for this seem to vary from case to case. We speculate that in some cases the reason may be that the diagram may look more elegant or the name of a concept may imply its

[12] `http://www.dase.cs.wright.edu/content/sdont`

[13] We note that there has been an update to both of these web application since we conducted our evaluation. However, due to time constraints, we were unable to go back and re-run the evaluation. Fortunately, we do not punish any tool for being unable to render a document; we only compare performance against the subset of patterns that *every* tool successfully processed.

| | $F_1$–Measure |
|---|---|
| SDOnt | 0.465 |
| OWLGrEd | 0.269 |
| WebVOWL | 0.163 |

Table 1: $F_1$–Measure for each of the tool's performance.

Table 2: Significance of Pairwise comparison of the tools using the Wilcoxon signed rank test.

| SDOnt vs OWLGrEd | SDOnt vs WebVOWL | OWLGrEd vs WebVOWL |
|---|---|---|
| $p < 0.001$ | $p < 0.001$ | $0.05 < p < 0.01$[14] |

natural superclass. For example, in the Hazardous Event pattern [3], HazardousEvent is a subclass to Event, but this is not indicated in the reference diagram, leading to a false positive. In other cases, the OWL file could be malformed.

To formally compare the performances we run three Wilcoxon signed rank tests, with null hypothesis that there are no difference in performance. Our tests show that SDOnt performs significantly better than OWLGrEd ($p < 0.001$), and WebVOWL ($p < 0.001$). As well as that OWLGrEd performs significantly better than WebVOWL ($0.05 < p < 0.01$)

There are also motivating cases for using schema diagrams as error checkers during modular ontology development. The tools do poorly on many of the diagrams simply because the respective OWL files do not actually contain the information the diagram implies. This may mean that they contain *more* information (e.g. alignments to different patterns) or lack information (e.g. errors).

VOWL and OWLGrEd consistently performed worse than SDOnt for two reasons. First, WebVOWL had many duplicated edges for different functional properties, even after adjusting settings in an attempt to prevent them. Secondly, both OWLGrEd and VOWL had trouble extracting properties from complex axioms. For OWLGrEd these axioms were represented as anonymous nodes, leading to false positive artifacts. Set operation nodes in WebVOWL also lead to additional false positive artifacts.

## 6 Conclusions

Our results are promising even if they do not present as such. A lack of a consistent visual notation for diagrams in our test set and poor quality control in the OWL files definitely contribute to a poor, raw showing. However, we note that even given our low $F_1$–measures the results are consistent. And we see that SDOnt performs significantly better than both OWLGrEd and WebVOWL for generating schema diagrams that are most similar to the reference diagrams.

To be fair, the test set against which we evaluated contained many UML-esque diagrams, to which none of the evaluated tools are well-suited. VOWL and OWLGrEd were used for comparison simply because they are the current state-of-the-art for generalized ontology visualization and they are the tools which produced the most similar diagrams to the desired ones. Our results do not invalidate VOWL or OWLGrEd: they simply serve other purposes.

There are still many ways to improve our method and its implementation. Firstly, we see in many diagrams that namespaces are frequently color coded, as well as providing different node styles for external patterns. As ontology engineering practices mature, we expect to see these distinctions to be formally encoded in the ontology, e.g., according to the Ontology Design Pattern Representation Language (OPLa) as described in [8]. As such, once the necessary tooling support for OPLa has been realized, SDOnt will be able to leverage the annotations and inform style and placement of nodes for increased clarity in the schema diagram. We will also explore different styles of incorporating UML-like visualizations for datatypes. The manually created reference diagrams are fallible or simply unclear from the perspective of the OWL file which information is necessary to convey. We believe incorporating OPLa and augmenting SDOnt to account for these annotations will also help in this regard.

Secondly, we intend to investigate the most effective ways of creating a good layout beyond force-directed graphs and will explore the option of providing our work as an additional rendering capability for the OWLAPI.

Finally, we will integrate SDOnt with other existing Protégé plugins developed in our lab, including ROWL,[15] OWLax,[16] and OWL2DL[17] [15–17], in order to work towards a well-rounded ontology engineering suite which supports the Modular Ontology Modeling paradigm.

*Acknowledgement.* Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI).

## References

1. Scalable visualization of semantic nets using power-law graphs. In *Applied Math*, volume 8, pages 355–367, 01 2014.
2. J. Barzdins, G. Barzdins, K. Cerans, R. Liepins, and A. Sprogis. 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*, volume 614 of *CEUR-WS.org*, 2010.
3. M. Cheatham, H. Ferguson, I. Charles Vardeman, and C. Shimizu. A modification to the hazardous situtation ODP to support risk assessment and mitigation. In *Proceedings of WOP*, volume 16, 2016.
4. A.-S. Dadzie and M. Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, Apr. 2011.

[15] `http://dase.cs.wright.edu/content/modeling-owl-rules`
[16] `http://dase.cs.wright.edu/content/ontology-axiomatization-support`
[17] `http://dase.cs.wright.edu/content/owl2dl-rendering`

5. V. Geroimenko and C. Chen. *Visualizing the Semantic Web: XML-based Internet and Information Visualization*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
6. F. Ghorbel, N. Ellouze, E. Métais, F. Hamdi, F. Gargouri, and N. Herradi. MEMO GRAPH: An ontology visualization tool for everyone. *Procedia Computer Science*, 96(Supplement C):265–274, 2016.
7. 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*. IOS Press, 2016.
8. P. Hitzler, A. Gangemi, K. Janowicz, A. A. Krisnadhi, and V. Presutti. Towards a simple but useful ontology design pattern representation language. *Proceedings WOP 2017*, October 2017. To appear.
9. P. Hitzler and C. Shimizu. Modular ontologies as a bridge between human conceptualization and data. In P. Chapman, D. Endres, and N. Pernelle, editors, *Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018.
10. N. Karima. *Automated Rendering of Schema Diagram for Ontologies*. PhD thesis, Wright State University, 2017.
11. N. Karima, K. Hammar, and P. Hitzler. How to document ontology design patterns. In K. Hammar, P. Hitzler, A. Lawrynowicz, A. Krisnadhi, A. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 15–28. IOS Press, Amsterdam, 2017.
12. A. Krisnadhi and P. Hitzler. Modeling with ontology design patterns: Chess games as a worked example. 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 3–21. IOS Press, 2016.
13. T. Liebig and O. Noppens. Ontotrack: Combining browsing and editing with reasoning and explaining for OWL lite ontologies. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298 of *Lecture Notes in Computer Science*, pages 244–258. Springer, 2004.
14. S. Lohmann, S. Negru, F. Haag, and T. Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.
15. 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.
16. M. K. Sarker, A. Krisnadhi, and P. Hitzler. OWLAx: A Protégé 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.

17. 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.
18. A. Soylu and E. Kharlamov. Making complex ontologies end user accessible via ontology projections. In R. Ichise, F. Lécué, T. Kawamura, D. Zhao, S. Muggleton, and K. Kozaki, editors, *Semantic Technology - 8th Joint International Conference, JIST 2018, Awaji, Japan, November 26-28, 2018, Proceedings*, volume 11341 of *Lecture Notes in Computer Science*, pages 295–303. Springer, 2018.

# Most of OWL Is Rarely Needed

**Abstract.** The high expressivity of the Web Ontology Language (OWL) is a boon, in that it makes it possible to describe complex relationships between classes, properties, and individuals in an ontology. At the same time, however, this high expressivity can be an obstacle to correct usage and wide adoption. Past attempts to ameliorate this have included the development of specific, presumably human-friendly syntaxes, such as the Manchester syntax or graphical interfaces for OWL axioms, albeit with limited success. If modelers want to develop suitable OWL axioms it is important to study why managing ontology complexity is a difficult issue.

In this paper, we adopt an idea from the Protégé plug-in, OWLAx, which provides a simple, clickable interface to automatically input a very limited number of axioms of very limited expressivity. In particular, each of these axioms contains at most three classes or roles. We hypothesized that most of the axioms in existing ontologies can be expressed in terms of simple axiom patterns like these.

Our findings, based on an analysis of 280 ontologies from five public ontology repositories, confirm this hypothesis: Over 90% of class axioms are indeed expressible in this way. We provide a detailed analysis of our findings, and are also able to further reduce the number of axiom types which are needed to obtain coverage of this magnitude.

## 1 Introduction

Knowledge graph schema are complex artifacts that can be difficult and expensive to produce and maintain. This is perhaps especially true when encoding them in OWL (the Web Ontology Language) as ontologies. The high expressivity of OWL is a boon, in that it makes it possible to describe relationships between classes, properties, and individuals in an ontology. At the same time, however, this high expressivity is often an obstacle to its correct usage, which, in turn limits wide adoption. Past attempts to ameliorate this have included the development of specific, presumably human-friendly syntaxes, such as the Manchester syntax [9], or graphical interfaces for OWL axioms, albeit with limited success [14]. Additionally, certain engineering paradigms, or methodologies, have been developed, such as eXtreme Design [3] or Modular Ontology Modeling [6,17], that attempt to simplify the modeling process.

In general, these methodologies aim to guide ontology developers through the complex modeling process by either abstracting the complexity away (e.g. through the use of Ontology Design Patterns), or by limiting the scope of a complex landscape to something more immediately applicable and understandable. In this paper we are particularly interested in the latter, especially during

the axiomatization process. We believe that it is important to investigate new avenues for simplifying the creation of suitable OWL axioms.

One of the core tenets of the Modular Ontology Modeling methodology is to produce schema diagrams and then systematically axiomatize them, with the input of domain experts. This systematic axiomatization is inspired by the OWLAx plugin for Protégé,[1] which provides a simple, clickable interface to automatically input a very limited number of axioms of very limited expressivity [15]. In particular, each of these axioms contains at most three classes or roles. In [15], it was posited (but left unproven) that the 17 axiom patterns provided by the interface were sufficient for *most* modeling purposes.

In this paper, we test that hypothesis by analyzing 280 ontologies from five public ontology repositories. Concretely, we show the following:

**H1.** Almost all axioms in OWL ontologies are covered by the set of simple axiom patterns found in Table 1.

And indeed, as we will see, it holds for over 90% of class axioms using our relatively straightforward analysis. With a more thorough analysis, the percentage may even be higher.

The rest of this paper is organized as follows. In Section 2 we briefly describe literature related to our analysis. Section 3 presents our research method and then Section 4 presents our evaluation. The results are discussed in Section 5. Finally, in Section 6 we conclude.

## 2 Related Work

We are aware of only a very limited amount of research that specifically concerns the ideas addressed in this paper. Zhang et al. [22] look at ways to measure the design complexity of ontologies. Their work is focused more on ontology quality evaluation than ontology composition. Some have also attempted to measure the effect that axioms like existential quantifiers have on reasoning time, such as Kang et al. [10], although it is only tangentially related to the work that we are presenting.

There are also, as previously mentioned, tools that attempt to simplify OWL ontology development, such as Manchester Syntax [9], WebVOWL [12], CoModIDE [16], Graffoo [4], and ROWLTab [14]. However, these tools merely simplify the development process and do not measure whether OWL axioms actually *are* complex in everyday usage. It could very well be the case that OWL is necessarily complex and these tools are needed to deal with this complexity, although we believe our work demonstrates that this is usually not the case.

## 3 Methodology

Our hypothesis is that most ontologies either are, or could be expressed mostly with, simple axioms. In this section, we will define what we mean by simple

[1] See `https://protege.stanford.edu/`.

axioms, then give an example of a set of simple axioms, such as those used in the Protégé plugin, OWLAx. Following that, we will show how to determine simple axiom coverage for an ontology, and then describe some basic normalizations that can produce accurate coverage results.

### 3.1 Simple Axioms

The simple axioms we study in this paper are defined below. We consider description logic syntax for OWL DL, that is, we identify it with the description logic $\mathcal{SROIQ}(D)$ [7].

**Definition 1.** *A **Simple Axiom** is any axiom that contains at most three atomic concepts or roles, and is not a syntactic shortcut for a larger axiom. Any axiom which is not simple is a **Complex Axiom**.*

Our set of axiom patterns matches class axioms, so we restrict our focus to class axioms in the evaluation, although, in principle, the notion of a simple axiom could apply to the role axioms as well. The limitation of three atoms for simple axioms is an intuitive threshold, in terms of size, because it means that nesting in the expression is limited to at most one quantifier, yet the axiom can still participate in complex inferences in combination with other simple axioms. This would not be the case for axioms limited to size two, where one could only express $A \sqsubseteq B$ for concepts, or $R \sqsubseteq S$ for roles, which would radically limit the expressivity of the ontology. Axioms with more than three atomic concepts or roles may be more expressive, but are often equivalent through normalization to smaller axioms, so they do not make not good candidates for simple axioms.

### 3.2 OWLAx

OWLAx [15] is a Protégé plugin that allows users to automatically generate certain simple OWL axioms using a graphical interface. The set of axioms we study in this paper are fashioned after the axioms that OWLAx can create, and they are listed in Table 1.

| | | | |
|---|---|---|---|
| Subclass | $A \sqsubseteq B$ | Functional | $\top \sqsubseteq \leqslant 1R.\top$ |
| Disjoint Classes | $A \sqcap B \sqsubseteq \bot$ | Qualified Functional | $\top \sqsubseteq \leqslant 1R.B$ |
| Domain | $\exists R.\top \sqsubseteq B$ | Scoped Functional | $A \sqsubseteq \leqslant 1R.\top$ |
| Scoped Domain | $\exists R.A \sqsubseteq B$ | Qualified Scoped Functional | $A \sqsubseteq \leqslant 1R.B$ |
| Range | $\top \sqsubseteq \forall R.B$ | Inverse Functional | $\top \sqsubseteq \leqslant 1R^{-}.\top$ |
| Scoped Range | $A \sqsubseteq \forall R.B$ | Inverse Qualified Functional | $\top \sqsubseteq \leqslant 1R^{-}.B$ |
| Existential | $A \sqsubseteq \exists R.B$ | Inverse Scoped Functional | $A \sqsubseteq \leqslant 1R^{-}.\top$ |
| Inverse Existential | $A \sqsubseteq \exists R^{-}.B$ | Inverse Qualified Scoped Functional | $A \sqsubseteq \leqslant 1R^{-}.B$ |
| | | Structural Tautology | $A \sqsubseteq \geqslant 0R.B$ |

Table 1: OWLAx Axiom Patterns

The actual implementation details of the OWLAx plugin are not pertinent to what we want to discuss. Rather, we are interested in what it happens to contain: a set of only simple axioms. Since it was designed specifically to help create ontologies, we speculate that such axioms will appear in ontologies frequently. We now show how the *axiom patterns*, like those used in OWLAx, can be matched with axioms in an ontology. If a high percentage of axioms in an ontology are covered by the patterns from our table, then it must contain a high number of simple axioms, since the axioms in Table 1 are simple.

### 3.3 Coverage

To study whether axioms in an ontology match an axiom pattern, we require a precise definition of what axiom pattern coverage means. First, we define the term axiom pattern and then show what an evaluation of that axiom pattern's coverage looks like.

**Definition 2.** *An **Axiom Pattern** is a generic axiom that is structurally complete but may have uninstantiated terms that are used as variables in an axiom generator or pattern matching program.*

For example, the axiom pattern A $\sqsubseteq \exists$R.B for an existential axiom from Table 1, where A, B, R are pattern matching terms, matches the axiom Dog $\sqsubseteq \exists$chases.Squirrel, where Dog and Squirrel are concepts and chases is a role.

**Definition 3.** *The **Axiom Coverage** is the number of times an axiom in an ontology O is matched by an Axiom Pattern p as its most specific match, written $ac_p(O)$. We say most specific because it is possible for axioms to cover multiple patterns at once. If an axiom covers multiple patterns, the covered pattern is the pattern with the fewest variable terms for the axiom (i.e., the most* specific *of the covered axioms).*

A functional role axiom $\top \sqsubseteq \leqslant 1S.\top$, for instance, technically matches all of the functional patterns, because $\top$ is a class. However its most specific match is the functional pattern because there are fewer variable terms for the functional pattern than the scoped functional, qualified functional, and scoped qualified functional patterns. A range axiom matches the scoped range pattern for the same reason, but the range pattern has fewer variable terms, so the range pattern is its match, and so on.

**Definition 4.** *The **Ontology Coverage** for an ontology O and set of axiom patterns $\mathcal{P}$ is the sum of all of its Axiom Coverages divided by the number of axioms it contains $|O|$.*

$$oc_{\mathcal{P}}(O) = \frac{1}{|O|} \sum_{p \in \mathcal{P}} ac_p(O)$$

Because Axiom Coverage in $ac_{\mathcal{P}}(O)$ matches only the most specific pattern, the sum of the number of matches and non-matches is equal to the number of axioms (there are no duplicate matches).

And naturally we can define averages for these measures

**Definition 5.** *For a set of n ontologies $\mathcal{O}$ the* ***Average Ontology Coverage****, $\overline{oc}_{\mathcal{P}}(\mathcal{O})$, is given by*

$$\overline{oc}_{\mathcal{P}}(\mathcal{O}) = \frac{1}{n} \sum_{O \in \mathcal{O}} oc_{\mathcal{P}}(O)$$

*and the* ***Average Axiom Coverage****, $\overline{ac}_{\mathcal{P}}(\mathcal{O})$, is given by*

$$\overline{ac}_{\mathcal{P}}(\mathcal{O}) = \frac{\sum_{O \in \mathcal{O}} \sum_{p \in \mathcal{P}} ac_p(O)}{\sum_{O \in \mathcal{O}} |O|},$$

*where $|O|$ denotes the number of axioms in $O$.*

Though they are nearly the same and certainly related, it is important to note that the Average Ontology Coverage measures a set of ontologies, and the Average Axiom Coverage measures all the Axioms in a set of ontologies. Because the axiom measurement has fewer groupings it is in certain respects more precise, however we cannot produce a standard deviation since it simply has a single value.

It is also important to emphasize that the axiom patterns we use may have multiple matches, but in every case where there are multiple there is always a match with more instantiated terms than the others, constituting its most specific match. It might be possible in a different study to expand this definition for a more complex analysis, but for the purposes of this paper it is sufficient.

### 3.4 Normalization

We have discussed a method to evaluate coverage for an ontology. However, there remains an issue that ontologies may have been written for completely different purposes and at differing levels of complexity. For example, some ontologies are developed for complex reasoning applications, while others are used for more straightforward data integration. Even within a single ontology, different authors may express equivalent statements in non-equivalent ways based on personal preference or style. In order to evaluate a large number of ontologies uniformly, we therefore need at least a minimal normalization strategy taken from community standards that allows us to compare disparate sources without biasing the evaluation in favor of any particular style. For this, we use multiple strategies derived from common OWL tools and resources.

Our normalization begins by filtering out all axioms except class axioms and role axioms. This is necessary because there are many OWL axioms for which our pattern study will not apply. Included in this are assertion (ABox) axioms, since the notion of an axiom pattern has little relevance for a fact, but also axiom types such as annotation axioms, declaration axioms, and datatype definitions, that carry no or few formal semantics. The remaining class and role axioms are then transformed according to the following procedures.

The first transformation that we perform is an equivalence transformation based on the syntactic shortcuts defined in the OWL Structural Specification [13]. Whenever an axiom is found that has one of the forms in Column 1 of Table 2, we perform the designated substitution. This transformation is performed because we need a way to translate the axioms that OWLAPI [8] identifies as role axioms, but which are actually syntactic shortcuts for class axioms, into forms the program can match with. It is also possible that other simple transformations of class axioms according to the equivalences in the structural specification could reduce false negative matches. Thus for EquivalentClasses, DisjointClasses, and DisjointUnion we use the OWLAPI builtin transformations to obtain SubClass axioms.

| Ontology Axiom | Substituted Axiom |
|---|---|
| ReflexiveObjectProperty(R) | $\top \sqsubseteq \exists \mathrm{R}.\mathbf{Self}$ |
| IrreflexiveObjectProperty(R) | $\exists \mathrm{R}.\mathbf{Self} \sqsubseteq \bot$ |
| FunctionalObjectProperty(R) | $\top \sqsubseteq \leqslant 1\mathrm{R}.\top$ |
| FunctionalDataProperty(S) | $\top \sqsubseteq \leqslant 1\mathrm{S}.\top$ |
| InverseFunctionalObjectProperty(R) | $\top \sqsubseteq \leqslant 1\mathrm{R}^{-}.\top$ |
| ObjectPropertyRange(R C) | $\top \sqsubseteq \forall \mathrm{R}.\mathrm{C}$ |
| DataPropertyRange(S D) | $\top \sqsubseteq \forall \mathrm{S}.\mathrm{D}$ |
| ObjectPropertyDomain(R C) | $\exists \mathrm{R}.\top \sqsubseteq \mathrm{C}$ |
| DataPropertyDomain(S C) | $\exists \mathrm{S}.\top \sqsubseteq \mathrm{C}$ |

where R is a Role, S is a Data Property, C is a Concept, and D is a Data Range

Table 2: Axiom Transformations

The second transformation that we use is the application of OWLAPI builtin functions to obtain negation normal form (NNF) on all class axioms in an ontology. By using the standard OWLAPI NNF functions we can transform all of the class axioms in an ontology into simple forms that are stripped of semantically unnecessary information that might be due to coincidence rather than equivalence.

The last transformation we apply is splitting SubClass axioms with conjunctions in the consequent, or disjunctions in the antecedent, into separate axioms. This is a standard procedure in many normalizations, and we simply use the default functions in OWLAPI to add a set of axioms formed from the conjuncts or disjuncts whenever an axiom of this type is found. There is a special case which occurs only when the consequent is an ExactCardinality expression whose value is equal to 1. In this case, we do not use a MinCardinality 1 substitution but instead add an existential, since that is equivalent, and it is a more compact expression.

## 4 Evaluation

We analyze a set of 280 ontologies from various sources, normalizing them, and testing them for axiom pattern coverage according to the principles described in the previous section. Ontologies were selected from diverse sources with unique design requirements: benchmark ontologies, Ontology Design Patterns (ODPs), as well as medical domain ontologies. Average statistics about the original ontologies gathered before processing can be found in Table 3.

| | All Ontologies | Hydrography | Anatomy | Conference | ODP | Ontobee | Misc |
|---|---|---|---|---|---|---|---|
| classes | 616,912 | 495 | 6,048 | 498 | 594 | 528,481 | 80,796 |
| roles | 12,320 | 168 | 5 | 226 | 621 | 10,765 | 535 |
| axioms | 1,286,947 | 6,054 | 16,383 | 2,153 | 3,295 | 986,728 | 272,334 |
| ontologies | 280 | 5 | 2 | 7 | 87 | 175 | 5 |

Table 3: Ontology Statistics

In this section, we report the result for all ontologies we tested, then go into details about each source, reporting a separate evaluation for each. Then we break down the results by profile and report the numbers for those as well. In all cases, the coverage numbers are reported for All Axioms, Class Axioms, and Simple Axioms. Our axiom patterns can only match simple axioms, thus the values for Class Axioms and All Axioms are a reflection of the number of complex class axioms and the number of role axioms in ontologies that have no chance at matching. The last value we report, which is a byproduct of calculations that produce coverage numbers, is the percent subclass and percent existential, as well as their combination. By this we mean, what percent of all of the axioms in an ontology are matching the axiom patterns subclass, existential, or both. It will turn out in nearly every case that a surprisingly high proportion of most ontologies are expressible with just these two axiom patterns.

Resources that were used to produce the data and the code that performed the evaluation can be requested through the program chairs, as some files are very large and difficult to distribute in an anonymous fashion.

### 4.1 Overall Coverage

The average axiom coverage and the average ontology coverage for our simple axioms over all ontologies is included in Table 4, as well as the standard deviation for the ontology coverages.

For each ontology analyzed, Figure 1 shows the overall percent of axioms that are expressible as each pattern for the entire collection of ontologies. Simple subclass is 55.5%, and existential is 25.0%, totaling 80.5%. This is almost the same as the axiom coverage value for all axioms. A more detailed view of axiom type distributions can be found in the next section in Figure 2.

8

| | Axiom Coverage | Ontology Coverage | StdDev Ont. Cov. |
|---|---|---|---|
| All Axioms | 82.4% | 81.0% | 0.184 |
| Class Axioms | 83.4% | 90.3% | 0.136 |
| Simple Class Axioms | 99.8% | 97.9% | 0.109 |

Table 4: Average Overall Coverage

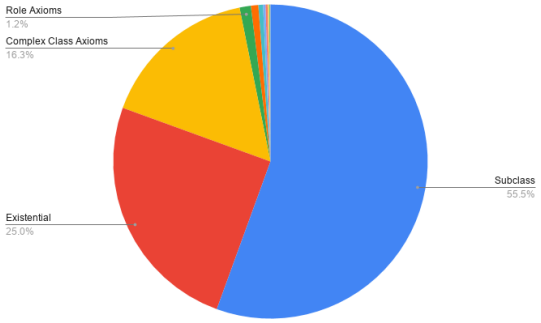



Fig. 1: Overall Axiom Percent

### 4.2 Source Coverage

As they are all from very different domains, each source was analyzed independently from the whole. We obtained benchmark ontologies that are used for ontology alignment evaluation. There are 5 ontologies from Hydrography, 2 from Anatomy, and 7 from the Conference domains, and each appear in their own column in Tables 5, 6, and 7. We also obtained and evaluated 87 ODPs[2] [18], as well as a collection of 175 OWL files that are mainly from the medical domain from the ontobee[3] [21] website. Additionally, we gathered some ontologies that did not fall neatly into any of these categories but nonetheless seemed better to include in the overall result than omit. These ontologies are General Formal Ontology [5], GeoLink Modular Ontology [11], Gene Ontology [1], GeoLink base ontology [11], and The Enslaved Ontology [19], and their average is labeled Misc in the tables.

The Gene Ontology tends to dominate the other sources in Misc due to its extremely large size. It also contains a much higher percentage of complex class axioms than any other ontology we tested, which likely accounts for the difference in Misc between simple class coverage and class coverage.

[2] `http://ontologydesignpatterns.org`.
[3] `http://ontobee.org`.

| | Hydrography | Anatomy | Conference | ODP | Ontobee | Misc |
|---|---|---|---|---|---|---|
| All Axioms | 75.0% | 99.9% | 89.2% | 85.2% | 88.6% | 62.4% |
| Class Axioms | 81.6% | 100% | 96.1% | 97.2% | 89.8% | 62.5% |
| Simple Class Axioms | 95.2% | 100% | 99.1% | 99.0% | 99.8% | 99.9% |

Table 5: Average Axiom Coverage By Source

| | Hydrography | Anatomy | Conference | ODP | Ontobee | Misc |
|---|---|---|---|---|---|---|
| All Axioms | 73.5% | 99.9% | 86.9% | 74.6% | 84.5% | 64.0% |
| Class Axioms | 84.5% | 100% | 95.3% | 94.9% | 88.0% | 80.6% |
| Simple Class Axioms | 98.8% | 100% | 98.8% | 97.6% | 97.9% | 99.3% |

Table 6: Average Ontology Coverage By Source

In Table 8, we see the range of percent subclass and existential among sources. Anatomy, Ontobee, and Misc all contain medical domain ontologies, which may account for the increase in percent existential if they contain more ontologies in the EL profile. Except for the Anatomy Benchmarks, which is actually only two ontologies so a disproportionately small sample size, it does not appear to be the case that any sources are entirely existential and subclass. Neither are any sources completely lacking the two axiom patterns. When we break the results down by profile in the next section, things will look quite a bit different.

In Figure 2, the previously mentioned high percentage of complex class axioms for Misc can be seen in the third column. Some other interesting percentage patterns also appear on close inspection. The two ontology sources with the highest percent subclass and existential are Anatomy and Ontobee, both medical type ontology sources. If we move farther down the chart to the less common axiom patterns, the larger ontology sources are less prevalent and now the benchmarks and ODPs start to dominate. The last three axiom patterns were never matched by our program. For the inverse functional axiom patterns it is conceivable that no-one had occasion to write axioms like this. Disjoint classes may seem surprising, however we speculate that, even though our pattern, $A \sqcap B \sqsubseteq \bot$, is expressible in profiles that do not contain negation, authors are using Protégé or OWLAPI to state disjoint classes axioms, which will normalize to a subclass axiom containing negation. This can cause disjoint classes to match the subclass pattern, so it is not a false negative, but it is a misclassification due to our differing terminologies.

10

| | Hydrography | Anatomy | Conference | ODP | Ontobee | Misc |
|---|---|---|---|---|---|---|
| All Axioms | 0.119 | 0.0 | 0.063 | 0.183 | 0.175 | 0.155 |
| Class Axioms | 0.112 | 0.0 | 0.030 | 0.077 | 0.158 | 0.154 |
| Simple Class Axioms | 0.039 | 0.0 | 0.013 | 0.067 | 0.130 | 0.008 |

Table 7: Average Ontology Coverage Standard Deviation By Source

| | Hydrography | Anatomy | Conference | ODP | Ontobee | Misc |
|---|---|---|---|---|---|---|
| Subclass | 40.1% | 66.8% | 57.6% | 55.4% | 60.0% | 41.1% |
| Existential | 07.4% | 33.1% | 06.7% | 05.7% | 26.4% | 20.7% |
| Subclass + Existential | 47.6% | 99.9% | 64.3% | 61.1% | 86.4% | 61.8% |

Table 8: Percent Subclass and Existential By Source

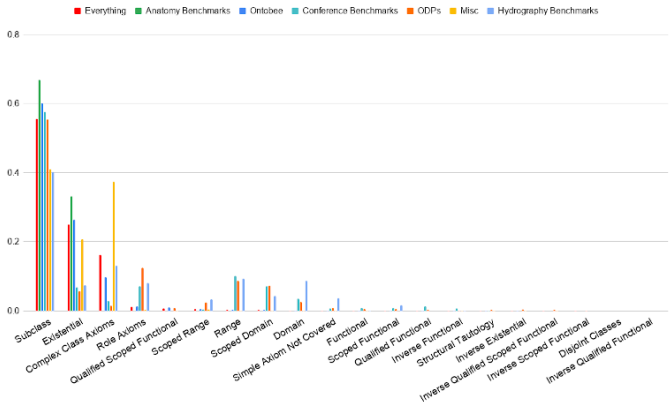



Fig. 2: Average Ontology Axiom Percent

Figure 3 shows the actual counts of each axiom pattern match in logarithmic scale. In this chart we can see how sources like Ontobee and Misc do contain some of the less common patterns. They are just so large that smaller sources, like ODPs and benchmarks, tend to have higher percentages.

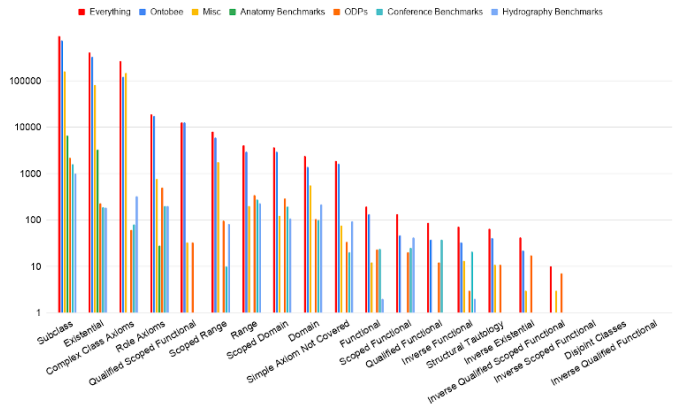



Fig. 3: Axiom Counts, Log Scale

### 4.3 Profile Coverage

During the analysis we also tested each ontology to see if it was in the OWL profiles EL, QL, RL, or DL, and computed the coverage information for each profile. In Tables 9, 10, and 11 we also reproduce the overall results in the column Full, since all ontologies will be in OWL Full. There were 8 ontologies that could be loaded, but the OWLAPI could not test their profile; these ontologies were not included in the profile results. Interestingly, for the EL, QL, and RL profiles we see around a ten percent coverage boost over the overall result. All three also have perfect coverage for simple class axioms, and nearly perfect coverage for all class axioms. The coverage numbers for OWL DL are also slightly higher than the overall numbers, though significantly less so than for the other profiles.

| | EL | QL | RL | DL | Full |
|---|---|---|---|---|---|
| All Axioms | 98.7% | 99.7% | 99.1% | 84.7% | 82.4% |
| Class Axioms | 98.8% | 100% | 100% | 85.7% | 83.4% |
| Simple Class Axioms | 100% | 100% | 100% | 99.8% | 99.8% |

Table 9: Average Axiom Coverage By Profile

Unlike the different sources, where the percent subclass and existential numbers were mostly near the average, we get a much more skewed result when we

| | EL | QL | RL | DL | Full |
|---|---|---|---|---|---|
| All Axioms | 96.6% | 90.1% | 92.2% | 84.4% | 81.0% |
| Class Axioms | 99.4% | 100% | 100% | 93.7% | 90.3% |
| Simple Class Axioms | 100% | 100% | 100% | 98.1% | 97.9% |

Table 10: Average Ontology Coverage By Profile

| | EL | QL | RL | DL | Full |
|---|---|---|---|---|---|
| All Axioms | 0.121 | 0.182 | 0.113 | 0.171 | 0.184 |
| Class Axioms | 0.022 | 0.0 | 0.0 | 0.112 | 0.136 |
| Simple Class Axioms | 0.0 | 0.0 | 0.0 | 0.094 | 0.109 |

Table 11: Average Ontology Coverage Standard Deviation By Profile

break the ontologies down by profile in Table 12. EL and DL ontologies seem to be expressible with a similar percent of subclass axioms as the overall result, though EL has many more existential expressions. QL ontologies, on the other hand, are eighty percent expressible as simple subclass. And the RL profile ontologies are almost entirely expressible as simple subclass axioms. It is no surprise, then, that EL, QL, and RL ontologies have such high coverage numbers, since subclass and existential are matched patterns.

| | EL | QL | RL | DL | Full |
|---|---|---|---|---|---|
| Subclass | 54.2% | 80.5% | 98.2% | 56.8% | 55.5% |
| Existential | 44.5% | 18.9% | 0% | 25.9% | 25.7% |
| Subclass + Existential | 98.7% | 99.5% | 98.2% | 82.8% | 80.5% |

Table 12: Percent Subclass and Existential By Profile

In Table 13, we mark which of our axiom patterns are expressible in each profile with an X symbol, using the OWL 2 Profiles [20] document as a reference. Comparing the different profiles with the coverage numbers, we can see how close their sums are, even though the expressible axioms are rather different. For instance, RL coverage is almost entirely subclass, and the existential pattern is inexpressible in that profile so the 0.0 value from Table 12 makes sense. EL seems to be evenly divided between subclass and existential, which again aligns with the types of statements permitted in the language. The DL profile allows all the types of expressions and it understandably has a similar result to the overall average.

For the EL profile we can also observe a unique result, because our axiom patterns have almost complete overlap with the 4 normal form class axioms defined for $\mathcal{EL}^{++}$ in [2], as shown in Table 14. The only exception is conjunction, which can only match our disjoint classes axiom pattern when the consequent is equal to $\bot$. If we were to define a conjunction axiom pattern, it might be possible

| | EL | QL | RL | DL |
|---|---|---|---|---|
| $A \sqsubseteq B$ | X | X | X | X |
| $A \sqcap B \sqsubseteq \bot$ | X | | X | X |
| $\exists R.\top \sqsubseteq B$ | X | X | X | X |
| $\exists R.A \sqsubseteq B$ | X | | X | X |
| $\top \sqsubseteq \forall R.B$ | | | | X |
| $A \sqsubseteq \forall R.B$ | | | X | X |
| $A \sqsubseteq \exists R.B$ | X | X | | X |
| $A \sqsubseteq \exists R^-.B$ | | X | | X |
| $\top \sqsubseteq \leqslant 1R.\top$ | | | | X |
| $\top \sqsubseteq \leqslant 1R.B$ | | | | X |
| $A \sqsubseteq \leqslant 1R.\top$ | | | X | X |
| $A \sqsubseteq \leqslant 1R.B$ | | | X | X |
| $\top \sqsubseteq \leqslant 1R^-.\top$ | | | | X |
| $\top \sqsubseteq \leqslant 1R^-.B$ | | | | X |
| $A \sqsubseteq \leqslant 1R^-.\top$ | | | X | X |
| $A \sqsubseteq \leqslant 1R^-.B$ | | | X | X |
| $A \sqsubseteq \geqslant 0R.B$ | | | | X |

Table 13: Profile Expressibility

to completely cover this profile for normalized class axioms. This could also be done for class axioms in the QL profile, where our axiom patterns would cover all simple axioms, and could cover any set of QL axioms that was normalized to remove nested quantifiers. Simple class axioms for the RL profile could be covered in much the same way as EL, missing only conjunction axioms that do not have $\bot$ in the consequent. With the addition of a conjunction pattern and by normalizing nested quantifiers we could also obtain complete class axiom coverage for RL.

| Axiom Pattern | $\mathcal{EL}^{++}$ Normal Class Axiom |
|---|---|
| $A \sqsubseteq B$ | $A \sqsubseteq B$ |
| $A \sqcap B \sqsubseteq \bot$ | $A \sqcap B \sqsubseteq C$, when $C = \bot$ |
| $\exists R.\top \sqsubseteq B$ | $\exists R.A \sqsubseteq B$, when $A = \top$ |
| $\exists R.A \sqsubseteq B$ | $\exists R.A \sqsubseteq B$ |
| $A \sqsubseteq \exists R.B$ | $A \sqsubseteq \exists R.B$ |

Table 14: $\mathcal{EL}^{++}$ Coverage

## 5 Discussion

Our motivation for this study is that we believe *simple axioms*, in general, are easier for non-logicians (e.g. domain experts) to understand and utilize for modeling. Alongside improved comprehension, they come with a number of added

benefits: attempting to measure the non-local effects of ontological commitments may be easier, they can be easily and automatically created by tools that allow users to specify statements in a graphical interface without a deep technical understanding of the inner-workings of OWL, and simple axioms often do not require normalization before being input to a reasoner. To support this, we determine the current usage characteristics of axioms in existing ontologies that are expressible as simple axioms, as well as how well this relates to the different OWL profiles.

To be specific, 120 of the ontologies we analyze are exclusively in the OWL Full profile, yet collectively they have a class axiom ontology coverage above 90% for our axiom patterns. There are exceptions, of course. Some ontologies have many more complex axioms than usual, such as the Gene Ontology, or consist primarily role axioms, so they are coverage outliers. However, in general, if even these more complex ontologies are for the most part expressible as simple axioms, then efforts to improve the modeling process should start with the simple axioms.

### 5.1 Future Work

In the future there are potentially many things we could do to improve on this study. One interesting approach would be to test different sets of simple axioms and see how the coverage numbers compare between them. OWLAx was a good basis to create a set of simple axioms but there are some obvious common ones that it lacks, for instance conjunction, disjunction, negation, as well as multiple variations on cardinality and role axioms.

We also admit that our definition of coverage is quite simple, intentionally kept this way for clarity. However it may be possible with some more comprehensive statistical tools that a better understanding of axiom usage in ontologies is possible. In a future study we may look into different evaluations besides coverage, perhaps it will be informative to compare.

Our method did normalize many axioms, however it is likely that complex axioms existed in the ontologies we studied that *could* have been normalized but *weren't* because our method only obtained NNF and then split up appropriate conjunction and disjunction axioms. By introducing new terms we might be able to even further increase the matching capability. Though, as previously mentioned, this would require the addition of new terms, so it would be equivalent but would also contain more entities, so the comparison would be less obviously appropriate.

Ultimately, though, we pursued this study to understand ontology axiomatization choices in practice. As such, this work can be built upon for improving ontology engineering tools, by providing particular support or focus on simple axioms.

## 6 Conclusion

In this paper we demonstrate that most class axioms in OWL ontologies are expressible with a small set of simple axioms. This has implications for how we can approach ontology management and development. If most ontologies are primarily simple then focusing on supporting and explaining these types of axioms can lead to easier adoption and maintenance. Complex axioms will of course always be a part of OWL, but we can improve our ontologies most easily by first making sure that the simple axioms are well understood and used correctly.

## References


1. Gene Ontology Consortium: The Gene Ontology (GO) database and informatics resource. Nucleic Acids Research **32**(Database-Issue), 258–261 (2004). https://doi.org/10.1093/nar/gkh036
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI. vol. 5, pp. 364–369 (2005)
3. 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, Studies on the Semantic Web, vol. 25, chap. 2, pp. 23–50. IOS Press (2016)
4. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL ontologies with Graffoo. In: European Semantic Web Conference. pp. 320–325. Springer (2014)
5. Herre, H.: General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In: Theory and applications of ontology: computer applications, pp. 297–345. Springer (2010)
6. Hitzler, P., Krisnadhi, A.: A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. CoRR **abs/1808.08433** (2018), `http://arxiv.org/abs/1808.08433`
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman and Hall/CRC Press (2010)
8. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proceedings of the 6th International Conference on OWL: Experiences and Directions – Volume 529. p. 49–58. OWLED'09, CEUR-WS.org, Aachen, DEU (2009)
9. Horridge, M., Patel-Schneider, P.F.: OWL 2 Web Ontology Language Manchester Syntax. W3C Working Group Note (2009)
10. Kang, Y.B., Li, Y.F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: International Semantic Web Conference. pp. 198–214. Springer (2012)
11. Krisnadhi, A., Hu, Y., Janowicz, K., Hitzler, P., Arko, R.A., Carbotte, S., Chandler, C., Cheatham, M., Fils, D., Finin, T.W., Ji, P., Jones, M.B., Karima, N., Lehnert, K.A., Mickle, A., Narock, T.W., O'Brien, M., Raymond, L., Shepherd, A., Schildhauer, M., Wiebe, P.: The GeoLink Modular Oceanography Ontology. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth,

P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9367, pp. 301–309. Springer (2015). https://doi.org/10.1007/978-3-319-25010-6_19
12. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 154–158. Springer (2014)
13. Parsia, B., Patel-Schneider, P., Motik, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). W3C recommendation, W3C (Dec 2012), http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/
14. Sarker, M.K., Krisnadhi, A., Carral, D., Hitzler, P.: Rule-based OWL modeling with ROWLTab Protégé plugin. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) The Semantic Web – 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 – June 1, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10249, pp. 419–433 (2017)
15. Sarker, M.K., Krisnadhi, A.A., Hitzler, P.: OWLAx: A Protégé 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 (2016)
16. 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 co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th to12th, 2018. CEUR Workshop Proceedings, vol. 2181, pp. 65–72 (2018)
17. Shimizu, C., Hammar, K., Hitzler, P.: Modular graphical modeling evaluated. In: Proceedings of ESWC (2020), to appear
18. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: A Modular Ontology Design Library. In: Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019). CEUR Workshop Proceedings, vol. 2459, pp. 47–58 (2019)
19. Shimizu, C., Hitzler, P., Hirt, Q., Shiell, A., Gonzalez, S., Foley, C., Rehberger, D., Watrall, E., Hawthorne, W., Tarr, D., Carty, R., Mixter, J.: The Enslaved Ontology 1.0: People of the historic slave trade. Tech. rep., Michigan State University, East Lansing, Michigan (April 2019)
20. Wu, Z., Fokoue, A., Grau, B.C., Horrocks, I., Motik, B.: OWL 2 Web Ontology Language Profiles (Second Edition). W3C recommendation, W3C (Dec 2012), http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/
21. Xiang, Z., Mungall, C., Ruttenberg, A., He, Y.: Ontobee: A linked data server and browser for ontology terms. In: ICBO (2011)
22. Zhang, H., Li, Y.F., Tan, H.B.K.: Measuring design complexity of semantic web ontologies. Journal of Systems and Software **83**(5), 803–814 (2010)

# An Ontology Design Pattern for Microblog Entries

Cogan Shimizu and Michelle Cheatham

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

**Abstract.** Due to the exponential growth of the Internet of Things and use of Social Media Platforms, observers have an unprecedented level of detailed information available on the behavior of communities. However, due to the highly heterogeneous nature and the immense volume of the data, a composite view is difficult to generate. Such a composite view would be exceptionally useful in the realms of insider threat detection, after-action forensics, and hazardous situation detection and avoidance. The Semantic Web, via ontology modeling, offers a powerful tool for fusing the disparate data sources and formats. To this end, we have created an ontology design pattern (ODP) for the modeling of a simple microblog entry. This ODP is intended to fit within an ecosystem for fusing social media, support advanced visualization, and provide a preliminary framework for trust assessment.

## 1 Motivation & Scope

In recent years, access to data has become increasingly trivial as Social Media Platforms and the Internet of Things (IoT) continue to grow. However, important latent or implicit information runs the risk of obfuscation simply by the sheer volume of collected data. Further, the data is presented and accessed via highly disparate vectors (e.g. microblog entries, visual media, and geotagged textual data). Thus, it is increasingly necessary to identify and develop methods for seamless fusion and visualization of information extracted from heterogeneous social media data.

Such methods are especially important for obtaining an accurate and comprehensive view of a crisis theater or battlespace (e.g. formulating a "Common Operating Picture"[1]). For these use cases, it is also important to take into account the provenance and trustworthiness of the acquired data and for any conclusions drawn from such data. To support the fusion of such heterogeneous data and the capture of its metadata, we will build an ecosystem of ontology design patterns [6]. ODPs enable sophisticated visualizations that leverage the inherent concept hierarchy, such as models displaying varying levels of granularity and interconnectedness. Figure 1, provides two examples of possible visualization methods that the microblog entry (MBE) will help support. We are currently

[1] A Common Operating Picture is a single identical display of relevant operational information on materiel shared by more than one Command. This term is frequently

investigating other visualizations in collaboration with domain experts from the United States Air Force. In this paper, we describe a pattern for a MBE as an entry point into developing the ecosystem.

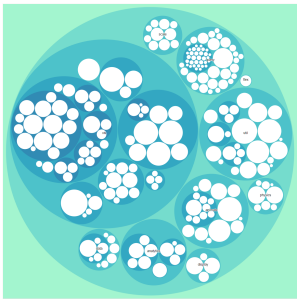
The MBE pattern is important for a number of reasons. First, microblog entries are representative of a fairly large subset of publicly available social media data. For example, Twitter[2] the popular, public-facing microblogging platform, allows a Tweet's payload to contain text, hyperlinks, images, or video. The entries may also be geotagged and may explicitly refer to other users. Additionally, there are many existing datasets that capture Tweets during natural disasters and humanitarian crises (e.g. CrisisLex[3]).

By definition and intent, microtext[4] is simple; its model is relatively straightforward and requires little of the complexity that OWL brings to the table. Regardless, it is important to note that this pattern is a fundamental building block of the intended ODP ecosystem. However, due to its simplicity, it is relatively straightforward to fit with many existing patterns. Specifically, we foresee easy integration with the ModifiedHazardousSituation Design Pattern [4] and ReportingEvent [7]. As the ecosystem matures, we also foresee including existing patterns regarding maps, climate, and public infrastructure.

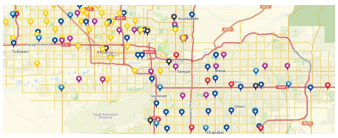
Finally, the MBE pattern has some components that allow for interesting interaction: spatiotemporal extent and author trustworthiness. Spatiotemporal extent of information is of particular interest to the modeling community as there are still many open questions on its handling. However, it is an integral part of any sort of response or intelligence operation. In a perfect world, we could assume that any author neither seeks to mislead nor propagate lies. However, in light of recent events, as well as the ODP's relevance to crisis and operational intelligence management, it is necessary to include a component for the trustworthiness of an author. Thus, the model for the microblog entry seeks to answer, at least, the following competency questions. Due to the strong emphasis on geospatial and temporal components of the fused data, we assume that these queries will be executed using geoSPARQL[5].

1. Who is the author of entry $x$?
2. What are all the entries authored by $y$?
3. What entries from time $A$ to time $B$ originate from region of interest $C$ with radius $D$?
4. What is the trust value $v$ for author $y$?
5. What is the trust value $v$ for entry $x$?
6. What entries from authors with a trust value greater than $v$ originate from a region of $C$ with radius $D$?
7. What entries relate to topic $T$?

[2] `https://twitter.com`
[3] `http://crisislex.org/`
[4] Microtext is any sufficiently short parcel of information in natural language. An MBE is an instance of microtext.
[5] `http://www.opengeospatial.org/standards/geosparql`



(a) A Circle Packing visualization generated by D3[6]. Smaller circles are related to the superimposed circle via subsumption and proximity in the same level of circle denotes a short semantic distance.



(b) A standard view of geographic information: pins on a map background. This visualization can be updated in real-time and allows the user to see incoming data.

Fig. 1: Both visualizations will utilize the MBE pattern at the most granular level (i.e. smallest circles and map pins).

Microtext is a valuable resource in the Semantic Web Community, as evidenced by [2, 9, 10, 8]. However, to our knowledge this is the first attempt at modeling an MBE as an entity, instead of only modeling extracted information.

The rest of the paper is organized as follows. Section 2 will address the design decisions in the structure of the pattern and accompanying axioms. Section 3 provides a motivating example and interaction with real data. Section 4 addresses future work and collaborations.

## 2 Pattern Overview

This pattern was directly informed by the competency questions in the preceding section; the competency questions are fairly straightforward and have a one to one correspondence with the concepts in the pattern. As such, the microblog entry pattern must capture both the entry's payload and its provenance. In addition, it must capture any information extracted from the payload and analysis of the author, such as answers to the questions: "To what is the microblog entry referring?" or "How trusted is the author by their peers?"

We will discuss the main design aspects of this pattern by referring its class diagram as depicted in Figure 2. Yellow boxes indicate datatypes, light blue boxes with dashed borders indicate external patterns. Purple is used for external

[6] Circle Packing is an arrangement of circles on a surface so that all circles touch one another. D3 is a powerful JavaScript library used for generating visualizations.

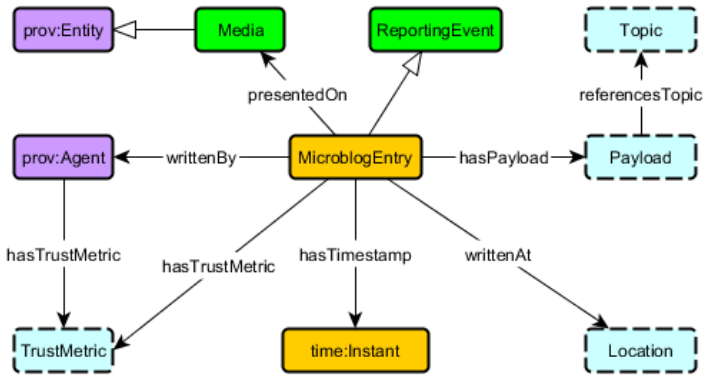



Fig. 2: A graphical representation of the microblog entry design pattern. Yellow boxes indicate datatypes, light blue boxes with dashed borders indicate external patterns. Purple is used for external classes belonging to PROV-O [5]. Green is used for external classes belonging to [7]. White arrowheads represent the owl:SubclassOf relation.

classes belonging to PROV-O [5]. Green depicts external classes belonging to [7]. White arrowheads represent the owl:SubclassOf relation.

By indicating several of the classes as "external," we intend to convey that the models for said classes are not indicative of the functionality of the MicroblogEntry pattern. For example, in our implementation[7] the light blue boxes are currently wrappers for datatypes. However, it is not hard to imagine increasingly complex models for each class. Below, we will discuss our implementation and future iterations. We will consider the pattern in the context of our use-case: event detection during a crisis. Furthermore, we assume that any microblog entry populating the ontology occurs within the time-frame and are shown to be relevant to the crisis situation.

**MicroblogEntry** The MicroblogEntry is the core class. Here, we will describe a few limitations placed upon its relations.

[7] The OWL file can be found at `https://raw.githubusercontent.com/cogan-shimizu-wsu/MicroblogEntryOWL/master/MicroblogEntry.owl`

$$\mathsf{MicroblogEntry} \sqsubseteq =1\mathsf{hasPayload}.\mathsf{Payload} \quad (1)$$
$$\mathsf{MicroblogEntry} \sqsubseteq =1\mathsf{hasAuthor}.\mathsf{Author} \quad (2)$$
$$\mathsf{MicroblogEntry} \sqsubseteq \leq 1\mathsf{hasLocation}.\mathsf{Location} \quad (3)$$

1. A MicroblogEntry may only have one Payload.
2. A MicroblogEntry may only have one Author.
3. A MicroblogEntry might not have a location attached to it.

**ReportingEvent** The ReportingEvent pattern is documented in [7]. This established pattern provides for a lot of interplay with MicroblogEntry, as well as providing structure for how information is shared.

As ReportingEvent is itself a subclass of Situation, it will be reasonably straightforward to integrate the ModifiedHazardousSituation [4] pattern to the MicroblogEntry. Additionally, ReportingEvent provides a framework for connecting the "report" to an ActualEvent; thus, along with Topic, ground the MicroblogEntry in reality. Finally, the fact that a ReportingEvent *isBasedOn* a Source, provides us a vehicle for capturing the fact that a MicroblogEntry has been re-Tweeted or shared (without modification).

**Media** The Media class allows us to represent the platform on which the MicroblogEntry was posted. In the case of our example in the next section, this would be Twitter. However, it is also conceivable that Media may represent CNN, Fox News, BBC, and so on. Obviously, these establishments are fairly complex in their own right.

Media is also drawn from [7], though is largely left for others to implement. Monitoring different Media will be very important in our use case scenario, especially when considering the TrustMetric for provenance and author. To this point, it seems reasonable to expect the trustworthiness of the platform and corporation to effect the trustworthiness of the reported data.

**Payload** The Payload is the content of the MicroblogEntry. In Figure 3, this is the content in Box 2. For the general pattern, we opted to leave this as an external pattern due to the expected heterogeneity of MBEs of different platforms and even high variance of content on the same platform. That is, Twitter allows for many different payloads: text, hyperlinks, images, and videos. Facebook, on the other hand, offers a superset of content types and no length restriction on text payloads.

In addition, we see the Payload playing a large role in defining how MBEs will interact with each other. In the case of Tweets, a Tweet may be "Retweeted," thus embedding a Tweet inside of a Payload. Furthermore, a Payload may "mention" another user or author. Our next steps will include ways to more accurately model these relationships between Authors, Payloads, and MicroblogEntries.

For our initial implementation, as our test sets do not include Tweets with pictures or hyperlinks, Payload wraps an xsd:string. Additionally, relevant MicroblogEntries must have a relevant Payload. That is, the Payload must refer to some Topic relevant to the crisis situation.

**Topic** In some cases, it may make sense to have Topic include a targeted list of terms from a controlled vocabulary. Or, instead, to have the Topic act as a category. For example, in [3], Tweets were partitioned into the following categories: affected individuals, infrastructures and utilities, donations and volunteer, caution and advice, sympathy and emotional support, useful information and unknown.

Our implementation currently wraps an xsd:string. This allows us to dynamically generate a Topic as Tweets are encountered. As the intended ODP ecosystem matures, it is conceivable that this Topic sub-pattern will be more fully fleshed out, allowing for more interesting interaction between MicroblogEntries referencing the same Topic.

**Location** There are many methods for representing location, e.g. the POI:Place [1] pattern or using WellKnownText (WKT) from OpenGIS, among others. To promote reusability, we do not constrain the top-level pattern to use one or another. In our implementation, however, we opted to use a WKT literal for simplicity's sake. In the future, we expect to be able to augment this part of the model by including relevant descriptors, such as the name of the location taken from a gazetteer.

**TrustMetric** The TrustMetric sub-pattern has the potential to be the most complex due to its far reaching effects on the interplay between Author, Payload, and Media. In addition, the actual metric for trust will need its own provenance and uncertainty measures. Until the system is actually implemented, it will be difficult to completely model. Thus, in our implementation, we assume we are getting a value between 0 and 1 from some black-box system. As such, we wrap xsd:double.

## 3 Example Triples

Figure 3 shows an example Tweet. The relevant data that will be extracted has been boxed in red.

```
kast:CarAccident            ## Extracted from Box 2
    rdf:type                t:Topic;
    t:hasName               "Car Accident"^^xsd:string;
.

kast:Evacuation             ## Extracted from Box 2
```


Fig. 3: An example Tweet with extracted data highlighted in red. Note, this example does not have a geolocation.

```
    rdf:type                t:Topic;
    t:hasName               "Evacuation"^^xsd:string;
.

kast:examplepayload         ## Extracted from Box 2
    rdf:type                pl:Payload;
    kast:hasvalue           "There is a car accident on 4th and
                            Main. Be careful out there!
                            #evac"^^xsd:string;
    kast:referencesTopic    kast:CarAccident, kast:Evacuation;
.

kast:cogantm                ## Note here that there are two trust metrics.
    rdf:type                tm:TrustMetric;
    tm:hasValue             .99^^xsd:double;
.

kast:mbetm                  ## As trust in author is distinct from trust in the MBE.
    rdf:type                tm:TrustMetric;
    tm:hasValue             .89^^xsd:double;
.

kast:CoganShimizu           ## Extracted from Box 1
   a prov:Person, prov:Agent;
   foaf:givenName           "Cogan Shimizu"^^xsd:string;
   kast:hasTrustMetric      kast:cogantm;
.

kast:Twitter
    rdf:type                pz:Media, prov:Entity;
.
```

```
kast:examplets              ## Extracted from Box 3
    rdf:type                time:Instant;
    time:inXSDDateTimeStamp "2017-07-12T10:01:00-5:00"^^xsd:dateTimeStamp;
.
```

And finally,

```
kast:exampletweet
    rdf:type                kast:MicroblogEntry, pz:ReportingEvent;
    kast:hasPayload         kast:examplepayload;
    kast:writtenBy          kast:CoganShimizu;
    kast:presentedon        kast:Twitter;
    kast:hasTrustMetric     kast:mbetm;
    kast:kastTimestamp      kast:examplets;
.
```

## 4 Conclusions and Future Work

The Microblog Entry Ontology Design Pattern is a useful model for a very commonplace structure, especially as the amount of social media data available for inspection continues to increase. The potential applications of this pattern are widespread, from determining public sentiment, measuring affect, or investigating community formation and evolution on social media networks.

The Microblog Entry pattern is foundational. On its own, it is not particularly remarkable. However, in the ecosystem it plays a fundamental role. In similar systems, it is analogous to entity extraction. Knowing the entities in play is important, but ultimately provides only a small facet of a crisis situation. The Microblog Entry pattern serves a similar role. It provides the threads to weave a more comprehensive picture. At this time, the pattern heavily relies on many external patterns, though many of them can be implemented as simple wrappers for datatypes. Future work will be focused on developing the ecosystem of ODPs for building a Common Operating Picture for a crisis situation. We will also investigate how the different visualizations can be effected by the trust metric. As the work progresses, we will be working closely with domain experts in the United States Air Force.

*Acknowledgement.* The authors acknowledge support by the Dayton Area Graduate Studies Institute (DAGSI) and input from Vincent Schmidt, Ph.D.

## References

1. A. Alves, B. Antunes, F. C. Pereira, and C. Bento. Semantic enrichment of places: Ontology learning from web. *Int. J. Know.-Based Intell. Eng. Syst.*, 13(1):19–30, Jan. 2009.

2. S. P. Bhatt, H. Purohit, A. Hampton, V. Shalin, A. Sheth, and J. Flach. Assisting coordination during crisis: A domain ontology based approach to infer resource needs from tweets. In *Proceedings of the 2014 ACM Conference on Web Science*, WebSci '14, pages 297–298, New York, NY, USA, 2014. ACM.
3. G. Burel, H. Saif, M. Fernandez, and H. Alani. On semantics and deep learning for event detection in crisis situations. 2017. Available from `http://semdeep.iiia.csic.es/files/SemDeep-17_paper_5.pdf` on September 6, 2017.
4. M. Cheatham, H. Ferguson, C. Vardeman, and C. Shimizu. Modified hazardous situation odp. 2017. Available from `http://www.michellecheatham.com/files/modification-hazardous-situation.pdf` on September 6, 2017.
5. P. Groth and L. Moreau, editors. *PROV-Overview: An Overview of the PROV Family of Documents*. W3C Working Group Note 30 April 2013, 2013.
6. P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors. *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*. Studies on the Semantic Web. IOS Press, Amsterdam/AKA Verlag, Heidelberg, 2016.
7. E. Kowalczuk and A. Lawrynowicz. The reporting event ontology design pattern and its extension to report news events. 2017. Available from `http://ontologydesignpatterns.org/wiki/images/a/ac/WOP2016_paper_18.pdf` on September 6, 2017.
8. M. B. Lazreg, M. Goodwin, and O. Granmo. Information abstraction from crises related tweets using recurrent neural network. In L. S. Iliadis and I. Maglogiannis, editors, *Artificial Intelligence Applications and Innovations - 12th IFIP WG 12.5 International Conference and Workshops, AIAI 2016, Thessaloniki, Greece, September 16-18, 2016, Proceedings*, volume 475 of *IFIP Advances in Information and Communication Technology*, pages 441–452. Springer, 2016.
9. R. Nithish, S. Sabarish, M. N. Kishen, A. M. Abirami, and A. Askarunisa. An ontology based sentiment analysis for mobile products using tweets. In *2013 Fifth International Conference on Advanced Computing (ICoAC)*, pages 342–347, Dec 2013.
10. P. Thakor and S. Sasi. Ontology-based sentiment analysis process for social media content. *Procedia Computer Science*, 53:199 – 207, 2015. INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015.

# Towards a Pattern-Based Ontology for Chemical Laboratory Procedures

Cogan Shimizu[1], Leah McEwen[2], and Quinn Hirt[1]

[1] Data Semantics Laboratory, Wright State University, Dayton, OH, USA
[2] Cornell University, Ithaca, NY, USA

**Abstract.** There is an increasing expectation in the academic sector for chemistry researchers to conduct risk assessment during experimental planning. However, information concerning laboratory scale chemical reactivity hazards can be difficult to parse despite ongoing efforts to compile from reported incidents. Laboratory procedures do not always directly flag possible incompatibilities among constituents or other process factors. In this paper, we present a pattern-based ontology for capturing multiple factors involved in laboratory procedures, including chemical properties, states, conditions, actions, and associated hazard classifications.



## 1 Motivation

Developing chemical safety risk assessment tools useful for the academic sector will necessitate tapping into digitally curated data in ways that are relevant to the decision-making processes of research chemists, safety professionals, institutional administration, and other stakeholders. For example, a researcher might be looking at two known chemicals in a proposed reaction scheme and want to know of any conditions that might trigger an adverse outcome, if there are any known procedures for minimizing the likelihood of these conditions, and how to mitigate potential harm if something untoward did occur. The relevant data and information may come from a diverse set of sources covering physical properties,[3] synthesis protocols,[4] and previously reviewed incidents,[5] among other information.

Some of the most relevant information for analyzing risk appears in reports of incidents where safe control was exceeded, and the influence of reactivity and process factors can be considered in retrospect. However, such reports are not the focus of normal research practice and tend to be exceedingly brief mentions found sporadically in letters to editors of journals,[6] or as news items,[7] or occasionally rephrased as caution statements in vetted procedures.[8] Some of these

[3] https://pubchem.ncbi.nlm.nih.gov
[4] https://www.orgsyn.org
[5] https://www.csb.gov/investigations
[6] http://pubs.acs.org/cen/safety
[7] https://dchas.org/the-dchas-l-list
[8] http://cenblog.org/the-safety-zone/2016/02/oprds-safety-notables-from-the-literature

reports have been collected into reference sources such as Bretherick's Handbook of Reactive Chemical Hazards, and the Pistoia Chemical Safety Library.[9] Much of this content has been further compiled into an API-processable data stream within the PubChem database, dynamically presented in the Laboratory Chemical Safety Summaries format (LCSS)[10] described by the US National Research Council (NRC) [3]. However, the meaning remains "locked" in unstructured text and not easily parsed for incorporation into digital information workflows.

The ability to make this information discoverable at the time of need will depend in part on more systematic description of these hazard scenarios. There are many factors at play in conducting a laboratory procedure that may contribute to the potential risk of a given situation. There is a body of research dedicated to analyzing the operations and conditions of large scale chemical processes in industrial settings, where these processes are well-defined and carefully specified as part of the planning process [11].[11] However, such analyses are rarely conducted for chemical procedures developed iteratively at the laboratory level as defined by OSHA regulations in the United States. Analyzing procedures and coupling these with incident data can potentially bring to light incompatible combinations and problematic operations, as well as aid in planning for adjustments to experimental parameters. Domain terminology that describes key factors can enable the systematic analysis of relationships, such as combinations of chemicals, or substances under different conditions. Such approaches have been used for single analysis of M/SDS documents,[12] and chemical procedures.[13] Developing ontology patterns for chemical processes can more systematically represent potential intersections with hazardous situations [10].

Chemical information is predominantly organized by chemical entity, which is a limited perspective for discerning relationships among multiple process factors. The safety literature is no exception, focusing on hazard-related properties of individual chemicals or substances without reference to specific experimental context or to the surrounding laboratory conditions. Scale, concentration, temperature, pressure, flow rate, and many other chemical, process, operator, and environmental factors have the potential to trigger "runaway" hazardous situations.[14] A more complete risk assessment process, as described by the RAMP model, involves a holistic, laboratory level approach to managing risks beyond hazard identification [13]. Complementing the "object-based" index of specific chemical entities with "process-based" modeling could help surface information and data buried in the published literature on how these chemicals are being used under various conditions and combinations, and the potential for subsequent unintentional interactions to arise [9].

[9] http://www.pistoiaalliance.org/projects/chemical-safety-library
[10] https://pubchem.ncbi.nlm.nih.gov/lcss
[11] www.acs.org/hazardassessment
[12] www.ilpi.com/msds/ref/demystify
[13] http://chemicaltagger.ch.cam.ac.uk
[14] https://dchas.org/2017/04/05/information-flow-in-environmental-health-safety

As such, we have begun the construction of a pattern ecosystem for capturing these chemical interactions and laboratory procedures. The foundational pattern is a chemical process pattern, which has been adapted from the State Transition pattern, which, in turn, is a generalization of the Semantic Trajectory pattern [7]. With the pattern, we hope to answer the following competency questions.

1. What substances appear in a particular action, together?
2. What substances are ever in the same container?
3. What temperatures or pressures are associated with these substances (conditions and/or changes)?
4. What apparatus or equipment is involved and associated with which substances (eg. glassware, stir-bars, glove-box)
5. What substances are co-located after some particular action?

## 2 Chemical Process Pattern

In this section, we detail the Chemical Process Pattern. A graphical overview of the pattern can be seen in Figure 1.

### 2.1 State Transition Pattern

The State Transition Pattern is a novel adaptation or *modularization* [5] of the Semantic Trajectory Pattern [7]. We provide a graphical representation of the pattern in Figure 1a.

The State Transition Pattern is a generalization of the Semantic Trajectory Pattern. The Semantic Trajectory deals with some Thing that moves through time and space which are captured as Fixes. In the State Transition Pattern, we have abstracted time and location to be Conditions of some State.

However, for our use case, we must further modularize the State Transition Pattern. At this time, the alignment is a set of subclass relations between the patterns, as follows.

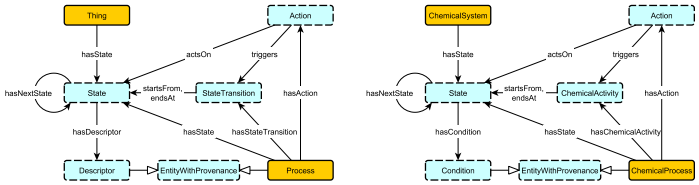
$$\mathsf{ChemicalSystem} \sqsubseteq \top$$
$$\mathsf{ChemicalActivity} \sqsubseteq \mathsf{StateTransition}$$
$$\mathsf{ChemicalProcess} \sqsubseteq \mathsf{Process}$$

Graphically, we see the results of these equivalences in Figure 1b.

### 2.2 Patterns Overview

**Scoped Domain and Range.** One of the primary goals of modelling with ontology design pattern is to lower the number of required ontological commitments required of an ontology engineer adopting the ontology. As such, we *scope* or *guard* many of the range and domain restrictions [6].

$$\mathsf{A} \sqsubseteq \forall \mathsf{R}.\mathsf{B} \tag{1}$$
$$\exists \mathsf{R}.\mathsf{B} \sqsubseteq \mathsf{A} \tag{2}$$




(a) A graphical representation of the State Transition Ontology Design Pattern.

(b) We *modularize* the State Transition ODP to construct the Chemical Process pattern.

Fig. 1: These two figures illustrate the modularization of the State Transition Pattern to Chemical Process Pattern.

Axiom (1) is a scoped range restriction. This allows us to say "when we relate A to something via R, that something must be a B." Axiom (2) follows the same for scoped domain restriction.

**Structural Tautologies.** These axioms are intended for human consumption; they do not add anything to the ontology. Essentially, these axioms, taking the below form, simply inform the reader of the intended use of a property [6].

$$\mathsf{A} \sqsubseteq \geq 0\mathsf{R}.\mathsf{B}$$

**OPLa Annotations.** The provided OWL file is annotated with the appropriate OPLa annoations [5]. We note, in particular, the classes marked as opla:ExternalClass: Action, Condition, and State. ChemicalActivity and EntityWithProvenance are defined later in the paper. The annotations were generated with the OPLa plugin for Protégé [12].

**Standard Disjointness.** In the following sections, all classes which are not in direct or inferred subclass relationship are declared to be mutually disjoint.

### 2.3 Action

Additionally, we provide graphical representations of the Stir Action and Heat Action subpatterns, as well as an expanded view of the Action Pattern in Figure 2. In the diagram, we use MethodTypes.txt and Apparatus.txt to denote that these values are *individuals* from a controlled vocabulary. An individual appearing the controlled vocabulary is an individual of type MethodType or Apparatus, for

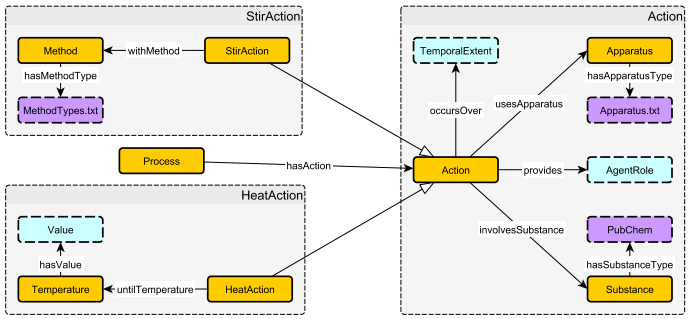



Fig. 2: Graphical overviews of the Action sub-patterns.

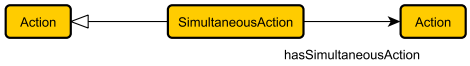



Fig. 3: Graphical overview of the Simultaneous Action Pattern.

example. The Simultaneous Action is shown in Figure 3.

$$\mathsf{Action} \sqsubseteq =1\mathsf{triggers}.\mathsf{ChemicalActivity} \tag{1}$$
$$\mathsf{Action} \sqsubseteq =1\mathsf{actsOn}.\mathsf{State} \tag{2}$$
$$\top \sqsubseteq \forall \mathsf{occursOver}.\mathsf{TemporalExtent} \tag{3}$$
$$\mathsf{Action} \sqsubseteq =1\mathsf{occursOver}.\mathsf{TemporalExtent} \tag{4}$$
$$\top \sqsubseteq \forall \mathsf{usesApparatus}.\mathsf{Apparatus} \tag{5}$$
$$\mathsf{Action} \sqsubseteq \geq 1\mathsf{usesApparatus}.\mathsf{Apparatus} \tag{6}$$
$$\top \sqsubseteq \forall \mathsf{hasApparatusType}.\mathsf{ApparatusType} \tag{7}$$
$$\forall \mathsf{hasApparatusType}.\top \sqsubseteq \mathsf{ApparatusType} \tag{8}$$
$$\mathsf{Action} \sqsubseteq =1\mathsf{provides}.\mathsf{AgentRole} \tag{9}$$
$$\top \sqsubseteq \forall \mathsf{involvesSubstance}.\mathsf{Substance} \tag{10}$$
$$\mathsf{Action} \sqsubseteq \geq 1\mathsf{involvesSubstance}.\mathsf{Substance} \tag{11}$$
$$\top \sqsubseteq \forall \mathsf{hasSubstanceType}.\mathsf{PubChem} \tag{12}$$
$$\forall \mathsf{hasSubstanceType}.\top \sqsubseteq \mathsf{Substance} \tag{13}$$

1. An Action triggers exactly one ChemicalActivity. However, we currently leave it to the ontology engineer to specify the exact complexity of a ChemicalActivity.
2. An Action acts on exactly one state.
3. The range of occursOver is strictly limited to TemporalExtent.
4. An Action occurs over exactly one TemporalExtent.
5. The range of usesApparatus is strictly limited to Apparatus.
6. An Action uses at least one Apparatus.
7. The range of hasApparatusType is strictly limited to ApparatusType.
8. The domain of hasApparatusType is strictly limited to Apparatus.
9. An Action provides exactly one AgentRole.
10. The range of involvesSubstance is strictly limited to Substance.
11. An Action always involves at least one Substance.
12. The range of hasSubstanceType is strictly limited to SubstanceType.
13. The domain of hasSubstanceType is strictly limited to Substance.

**StirAction**

$$\mathsf{StirAction} \sqsubseteq \mathsf{Action} \tag{14}$$
$$\top \sqsubseteq \forall \mathsf{withMethod}.\mathsf{Method} \tag{15}$$
$$\mathsf{StirAction} \sqsubseteq =1\mathsf{withMethod}.\mathsf{Method} \tag{16}$$
$$\top \sqsubseteq \forall \mathsf{hasMethodType}.\mathsf{MethodType} \tag{17}$$
$$\forall \mathsf{hasMethodType}.\top \sqsubseteq \mathsf{Method} \tag{18}$$

14. All StirActions are Actions.
15. The range of withMethod is strictly limited to Method.
16. A StirAction is completed with exactly one Method.
17. The range of hasMethodType is strictly limited to MethodType.
18. The domain of hasMethodType is strictly limited to Method.

**HeatAction**

$$\mathsf{HeatAction} \sqsubseteq \mathsf{Action} \tag{19}$$
$$\mathsf{HeatAction} \sqsubseteq =1\mathsf{untilTemperature}.\mathsf{Temperature} \tag{20}$$
$$\top \sqsubseteq \forall \mathsf{hasValue}.\mathsf{Value} \tag{21}$$
$$\mathsf{Temperature} \sqsubseteq =1\mathsf{hasValue}.\mathsf{Value} \tag{22}$$

19. All HeatActions are Actions.
20. A HeatAction has exactly one limiting Temperature.
21. The range of hasValue is strictly limited to Value.
22. A Temperature has exactly one Value.

**SimultaneousAction**

$$\mathsf{SimultaneousAction} \sqsubseteq \mathsf{Action} \tag{23}$$

$$\top \sqsubseteq \forall \mathsf{hasSimultaneousAction}.\mathsf{Action} \tag{24}$$

$$\top \sqsubseteq \forall \mathsf{hasSimultaneousAction}.\neg \mathsf{SimultaneousAction} \tag{25}$$

$$\forall \mathsf{hasSimultaneousAction}.\top \sqsubseteq \mathsf{SimultaneousAction} \tag{26}$$

$$\mathsf{hasSimultaneousAction} \circ \mathsf{occursOver} \sqsubseteq \mathsf{occursOver} \tag{27}$$

$$\mathsf{hasSimultaneousAction} \circ \mathsf{involvesSubstance} \sqsubseteq \mathsf{involvesSubstance} \tag{28}$$

23. All SimultaneousActions are Actions
24. The range of hasSimultaneousAction is strictly limited to Action.
25. A SimultaneousAction may not have another SimultaneousAction as a simultaneous action.
26. The domain of hasSimultaneousAction is strictly limited to SimultaneousAction.
27. The Actions that co-occur must, in fact, occur simultaneously.
28. Any Substance that is involved in a "subaction" is involved in the SimultaneousAction.

### 2.4 ChemicalActivity

$$\mathsf{ChemicalActivity} \sqsubseteq =1\mathsf{startsFrom}.\mathsf{State} \tag{1}$$

$$\mathsf{ChemicalActivity} \sqsubseteq =1\mathsf{endsAt}.\mathsf{State} \tag{2}$$

$$\top \sqsubseteq \forall \mathsf{startsFrom}.\mathsf{State} \tag{3}$$

$$\top \sqsubseteq \forall \mathsf{endsAt}.\mathsf{State} \tag{4}$$

$$\tag{5}$$

1. A ChemicalActivity always begins in some State and results in some State.
2. *supra*.
3. The range of startsFrom is strictly limited to States.
4. The range of endsAt is strictly limited to States.

### 2.5 ChemicalProcess

$$\top \sqsubseteq \forall \mathsf{hasAction}.\mathsf{Action} \tag{1}$$

$$\top \sqsubseteq \forall \mathsf{hasChemicalActivity}.\mathsf{ChemicalActivity} \tag{2}$$

$$\mathsf{ChemicalProcess} \sqsubseteq \geq 1\mathsf{hasAction}.\mathsf{Action} \tag{3}$$

$$\mathsf{ChemicalProcess} \sqsubseteq \geq 1\mathsf{hasChemicalActivity}.\mathsf{ChemicalActivity} \tag{4}$$

$$\mathsf{ChemicalProcess} \sqsubseteq \geq 1\mathsf{hasState}.\mathsf{State} \tag{5}$$

1. The range of hasAction is strictly limited to Activity.
2. The range of hasChemicalActivity is strictly limited to ChemicalActivity.
3. A ChemicalProcess must have at least one Action.
4. A ChemicalProcess must have at least one ChemialActivity.
5. A ChemicalProcess must have at least one State.

### 2.6 ChemicalSystem

$$\mathsf{ChemicalSystem} \sqsubseteq \geq 1 \mathsf{hasState}.\mathsf{State} \quad (1)$$
$$\top \sqsubseteq \forall \mathsf{hasState}.\mathsf{State} \quad (2)$$
$$\mathsf{State} \sqsubseteq \leq 1 \mathsf{hasState}^{-}.\top \quad (3)$$

1. A ChemicalSystem always has at least one State.
2. The range of hasState is strictly limited to State.
3. Any State is associated with exactly one Thing.

### 2.7 Condition

$$\mathsf{Condition} \sqsubseteq \mathsf{EntitywithProvenance} \quad (1)$$
$$\top \sqsubseteq \forall \mathsf{hasCondition}.\mathsf{Condition} \quad (2)$$
$$(3)$$

1. All Conditions must have provenance. In this use-case this is reasonable as every condition is measured by someone or some device.
2. The range of hasCondition is strictly limited to Conditions.

### 2.8 EntityWithProvenance

The EntityWithProvenance Pattern is extracted from the PROV-O ontology. At the pattern level, we do not want to make the ontological committment to a full-blown ontology. It suffices to align a sub-pattern to the core of PROV-O. Further discussion on the EntityWithProvenance pattern, as well as its specification (as below) in an OWL file may be found on the online portal.[15]

[15] `https://ontologydesignpatterns.org/wiki/Submissions:EntityWithProvenance`

$$\mathsf{EntityWithProvenance} \sqsubseteq \forall \mathsf{wasDerivedFrom.EntityWithProvenance} \quad (1)$$
$$\forall \mathsf{attributedTo.Agent} \sqsubseteq \mathsf{EntityWithProvenance} \quad (2)$$
$$\mathsf{EntityWithProvenance} \sqsubseteq \forall \mathsf{attributedTo.Agent} \quad (3)$$
$$\forall \mathsf{generatedBy.ProvenanceActivity} \sqsubseteq \mathsf{EntityWithProvenance} \quad (4)$$
$$\mathsf{EntityWithProvenance} \sqsubseteq \forall \mathsf{generatedBy.ProvenanceActivity} \quad (5)$$
$$\forall \mathsf{used.EntityWithProvenance} \sqsubseteq \mathsf{ProvenanceActivity} \quad (6)$$
$$\mathsf{ProvenanceActivity} \sqsubseteq \forall \mathsf{used.EntityWithProvenance} \quad (7)$$
$$\forall \mathsf{performedBy.Agent} \sqsubseteq \mathsf{ProvenanceActivity} \quad (8)$$
$$\mathsf{ProvenanceActivity} \sqsubseteq \forall \mathsf{performedBy.Agent} \quad (9)$$

1. The scoped range of wasDerivedFrom, scoped by EntityWithProvenance, is EntityWithProvenance.
2. The scoped domain of attributedTo, scoped by Agent, is EntityWithProvenance.
3. The scoped range of attributedTo, scoped by EntityWithProvenance, is Agent.
4. The scoped domain of generatedBy, scoped by ProvenanceActivity, is EntityWithProvenance.
5. The scoped range of generatedBy, scoped by EntityWithProvenance, is ProvenanceActivity.
6. The scoped domain of used, scoped by EntityWithProvenance, is ProvenanceActivity
7. The scoped range of used, scoped by ProvenananceActivity, is EntityWithProvenance.
8. The scoped domain of performedBy, scoped by Agent, is ProvenanceActivity.
9. The scoped range of performedBy, scoped by ProvenanceActivity, is Agent.

### 2.9 State

$$\top \sqsubseteq \forall \mathsf{hasNextState.State} \quad (1)$$
$$\mathsf{State} \sqsubset \leq 1 \mathsf{hasNextState.State} \quad (2)$$

1. The range of hasNextState is strictly limited to State.
2. A State will always follow at most one State.

## 3 Worked Example

The following incident report is extracted from [4, 1]. Formatting and language have been modified in order to make it clear exactly how the information was obtained. In the interest of brevity, we have used a simple incident report. However, even such a simple application of the pattern requires a high level of detail from the report. Thus, in our worked example, we aim to provide an illustration of the foundational concepts of our ontological ecosystem and note

certain aspects will be addressed in future work. In the following, we use the cpp: namespace as an abbreviation for "Chemical Process Pattern" in the URI https://daselab.org/chemicalprocesspattern/.

```
The Incident Report.

5-ethyl-2-methyl-pyridine and 70% nitric acid  were placed in
a small auto-clave.
They were heated and stirred for 40 minutes.
The emergency vent was opened due to a sudden pressure rise.
A violent explosion occurred 90 seconds later.
```

From the first statement, we extract the following triples regarding the substances and apparatus. The placement of the chemicals will also constitute an Action subclass, as it is developed.

```
cpp:sub1 rdf:type   cpp:Substance
         cpp:asText "5-ethyl-2-methyl-pyridine"  .
cpp:sub2 rdf:type   cpp:Substance
         cpp:asText "70% nitric acid"            .
cpp:ap1  rdf:type   cpp:Apparatus
         cpp:hasApparatusType "auto-clave"       .
```

From the next sentence we extract the StirAction and HeatAction. In order to capture their simultaneity, we use the SimultaneousAction.

```
cpp:te1   rdf:type                   cpp:TemporalExtent      .
cpp:sa1   rdf:type                   cpp:StirAction          .
cpp:ha1   rdf:type                   cpp:HeatAction          .
cpp:sim1  rdf:type                   cpp:SimultaneousAction
          cpp:hasSimultaneousAction  cpp:sa1
          cpp:hasSimultaneousAction  cpp:ha1
          cpp:occursOver             cpp:te1                 .
```

From the next sentence, we extract the apparatus and resulting state of the action. The Condition is provided an asText property for illustrative purposes.

```
cpp:ap2   rdf:type              cpp:Apparatus
          cpp:hasApparatusType  "fume hood"          .
cpp:c1    rdf:type              cpp:Condition
          ewp:isAttributedTo    cpp:ap2              .
          cpp:asText            "high pressure"      .
cpp:s2    rdf:type              cpp:State            .
cpp:s1    rdf:type              cpp:State
          cpp:hasNextState      cpp:s2               .
```

```
cpp:ca1   rdf:type              cpp:ChemicalActivity
          cpp:startsFrom        cpp:s1
          cpp:endsAt            cpp:s2               .
cpp:sim1  cpp:actsOn            cpp:s1
          cpp:triggers          cpp:ca1              .
```

In the last step, we note that a hazardous state has been entered. However, the development of this part of the ontological ecosystem is still planned in future work. We note possible integration the Modified Hazardous Material Pattern [2] to help model this aspect. Finally, we may wrap it all together into the Chemical Process.

```
cpp:cp1  rdf:type                cpp:ChemicalProcess
         cpp:hasAction           cpp:sa1
         cpp:hasAction           cpp:ha1
         cpp:hasAction           cpp:sim1
         cpp:hasChemicalActivity cpp:ca1
         cpp:hasState            cpp:s1
         cpp:hasState            cpp:s2              .
```

## 4 Conclusions

In this paper, we have described a foundational pattern to building a ontology design pattern ecosystem for modelling chemical processes. The core pattern is based on the State Transition Pattern, which in turn, is adapted from the Semantic Trajectory Pattern. The intent of this pattern and the surrounding ecosystem is to provide chemists–and their students– with a resource for analyzing experiments and potentially finding unforeseen interactions that can result in hazardous states, events, or situations.

A sufficiently populated ontology of chemical processes can also be used as background knowledge for training a more sophisticated learning model or could be used to explain the decisions *made* by such a system (deep learning models and explainable AI, respectively).

In the future, we expect to integrate more closely with the large chemistry based datasets, such as PubChem and M/SDS. In addition, there are existing patterns that may be integrated to enhance the functionality of the core pattern and complete other pieces, such as QUDT[16] for measurements and units, the ModifiedHazardous Material Pattern [2] for modelling hazardous states, and the Material Transformation [8] for extending ChemicalActivity.

*Acknowledgement.* Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI).

[16] `https://qudt.org/`

## References


1. Nitric acid. *National Center for Biotechnology Information. PubChem Compound Database; CID=944, datasheet=lcss.* Accessed May 30th, 2018.
2. M. Cheatham, H. Ferguson, C. Vardeman, and C. Shimizu. A modification to the hazardous situation ODP to support risk assessment and mitigation. In K. Hammar et al., editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 97–104. IOS Press, 2017.
3. N. R. Council. *Prudent Practices in the Laboratory: Handling and Management of Chemical Hazards, Updated Version.* The National Academies Press, Washington, DC, 2011.
4. R. L. Frank. volume 30:, pages 33–48. 1952.
5. 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.
6. P. Hitzler and A. Krisnadhi. On the roles of logical axiomatizations for ontologies. 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 73–80. IOS Press, 2016.
7. Y. Hu, K. Janowicz, D. Carral, S. Scheider, W. Kuhn, G. Berg-Cross, P. Hitzler, M. Dean, and D. Kolas. A geo-ontology design pattern for semantic trajectories. In T. Tenbrink, J. G. Stell, A. Galton, and Z. Wood, editors, *Spatial Information Theory - 11th International Conference, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings*, volume 8116 of *Lecture Notes in Computer Science*, pages 438–456. Springer, 2013.
8. C. F. V. II, A. A. Krisnadhi, M. Cheatham, K. Janowicz, H. Ferguson, P. Hitzler, and A. P. C. Buccellato. An ontology design pattern and its use case for modeling material transformation. *Semantic Web*, 8(5):719–731, 2017.
9. M. Leah. *ci*, volume 39, chapter Chemical Health and Safety Data Management, page 31. 2018 2017. 3.
10. L. McEwen and R. Stuart. Meeting the google expectation for chemical safety information. *Chemistry International*, 37(5-6):12–16, 2015.
11. M. B. Mulcahy, C. Boylan, S. Sigmann, and R. Stuart. Using bowtie methodology to support laboratory hazard identification, risk management, and incident analysis. *Journal of Chemical Health and Safety*, 24(3):14 – 20, 2017.
12. C. Shimizu, Q. Hirt, and P. Hitzler. A protégé plugin for annotating OWL ontologies with opla. ESWC 2018, June 2018. To Appear.
13. R. B. Stuart and L. R. McEwen. The safety "use case": Co-developing chemical information management and laboratory safety skills. *Journal of Chemical Education*, 93(3):516–526, 2016.

# Ontology Design Patterns for Winston's Taxonomy Of Part-Whole Relations*

Cogan Shimizu, Pascal Hitzler, and Clare Paul

[1] Data Semantics (DaSe) Laboratory, Wright State University, OH, USA
[2] Air Force Research Laboratory, Dayton, Ohio, USA

**Abstract.** While the formal modeling of part-whole relationships has been of interest, and studied, in many fields including ontology modeling, as of yet there has been no dedicated ontology design pattern which goes beyond the modeling of an absolute minimum. We correct this by providing two patterns based on Winston's landmark paper, "A Taxonomy of Part-Whole Relations."

## 1 Introduction

Part-whole relations are of fundamental importance for how we organize concepts. Consequently, they have been studied in philosophy [1,20,19], linguistics [3,4] geographical information systems (GIS) [2,9,18], to name just a few. Corresponding *partonomies* or *meronomies*, i.e. hierarchies built from part-whole relations, are therefore a recurring theme in ontology modeling.

Despite this, however, we have been unable to find a readily available or documented ontology design pattern for part-whole relationships, other than some very minimalistic proposals in the ontologydesignpatterns.org portal. In this paper we want to rectify this by providing such a pattern, together with a contextualized version of it. Our approach to this is to keep things as simple as possible, yet to make sure that the resulting patterns are comprehensive yet general enough to be applied in many contexts.

Concretely, we will follow an approach laid out by Winston in his 1987 landmark paper on "A Taxonomy of Part-Whole Relations" [20].[3] While this paper was based on linguistic considerations, it also provided for logical characterizations and axiomatics, which will inform our pattern. As such we do not claim much novelty, other than that we cast previous observations by us and others into reuseable ontology design patterns. In fact, the technical content of Section 3 is adapted from [8] by carrying it over to the context of ontology design patterns.

* This work will be published as part of the book "Emerging Topics in Semantic Technologies. ISWC 2018 Satellite Events. E. Demidova, A.J. Zaveri, E. Simperl (Eds.), ISBN: 978-3-89838-736-1, 2018, AKA Verlag Berlin."

[3] A discussion of different such theories in the context of logical knowledge representation for ontology engineering can be found in [10].

| Relation Type | funct. | hom. | sep. | Example |
|---|---|---|---|---|
| component-integral object | yes | no | yes | handle and cup |
| feature-activity | yes | no | no | paying and shopping |
| portion-mass | no | yes | yes | slice and pie |
| place-area | no | yes | no | everglades and florida |
| member-collection | no | no | yes | tree and forest |
| stuff-object | no | no | no | gin and martini |

**Table 1.** Types of part-whole relations according to Winston. funct. stands for functional, hom. stands for homeomerous, sep. stands for separable.

The rest of the paper is organized as follows: In Section 2 we briefly review Winston's approach to lay the ground for the technical contributions. In Section 3 we provide the basic Winston-Part-Whole Pattern. In Section 4 we provide the Contextualized Winston-Part-Whole Pattern as an extention of the one presented in Section 3. In Section 5 we describe a usage scenario. In Section 6 we briefly discuss a provenance pattern as an example for contextualization, which is essentially adapted from the core of the PROV-O ontology. Section 7 contains additional release information for the patterns, and Section 8 concludes.

## 2 Winston's Approach

Winston in [20] distinguishes six different types of part-whole relationships. His categorization is based on the following three aspects, a different selection of which holds for each of the types.

**separable (versus inseparable):** Parts can in principle be physically disconnected from the whole.
**functional (versus non-funcational):** Parts are in specific spatial and temporal position relative to each other which supports their functional role as parts of the whole.
**homeomerous (versus non-homeomerous):** Parts are similar to each other and to the whole.

The six types distinguished by Winston are listed in Table 1. The table also lists which of the just mentioned three aspects holds for each type, and an example from each, taken from [20].

Winston furthermore provides a discussion of logical properties for each type of part-whole relation. E.g., he observes that each type of relation is transitive, however if you mix types, transitivity generally does not hold. E.g., if you have two relations which are both of the component-integral object type, then transitivity holds, as in toe being part of the foot, foot being part of the leg, therefore toe is part of the leg. If you mix types, though, e.g. by mixing a component-integral object relation such as "Derek's nose is part of Derek" and a member-collection relation such as "Derek is part of the Department faculty,"

then transitivity would result in the nonsensical "Derek's nose is part of the Department faculty."

Rather than going through Winston's observations in detail, let us refer here to the axiomatization which we have drawn from it, and which we give in the next section.

## 3 The Winston-Part-Whole Pattern

We are now going to cast Winston's part-whole types into a part-whole ontology design pattern, and that will include the capturing, in OWL, of the logical relationships identified by Winston.

We will use the OWL property names

- component-integral object: po-component
- member-collection: po-member
- potion-mass: po-portion
- stuff-object: po-stuff
- feature-activity: po-feature
- place-area: po-place

and we will refer to these as *the specific part-whole relations*. We also use some other, related, relations identified and discussed by Winston. These are, in paticular, spatially-located-in as the spatial (topological) located-in relation and part-of as the generic part-whole relation of which the specific ones listed above are specializations (i.e., subProperties).

From [20] we can now draw the axioms which together constitute the pattern. They are listed in Figure 1.

Axioms (1) through (12) declare transitivity and asymmetry for each of the specific part-whole relations. According to Winston, however, we would also need to declare irreflexivity for each of the specific part-whole relations, which would render each of them a strict partial order. However this is not allowed in OWL 2 DL: according to [15, Section 11] a property cannot be both transitive (and, therefore, non-simple) and irreflexive.[4]

We believe that dropping the irreflexivity axioms should usually not cause any problems in terms of logical reasoning over the pattern, however as usual it is difficult to formally assess this. A formal declaration of irreflexivity may sometimes be helpful for ontology debugging or data curation, and of course some (correct) inferences will be missed through OWL 2 DL reasoning if the axiom is omitted. Note, though, that due to the open world assumption all inferences drawn from the OWL 2 ontology are still correct with respect to the complete theory (i.e., the one including irreflexivity).

Winston lists a number of additional axioms, however as discussed in [8] they are in fact tautologies, and while they may be informative for a linguistic

[4] Alternatively, we could also have dropped the transitivity axoims, but that seems less appealing. As discussed in [8], a third option would be to employ nominal schemas [12,14] and provide weaker forms of some of the axioms.

$$\mathsf{po\text{-}component} \circ \mathsf{po\text{-}component} \sqsubseteq \mathsf{po\text{-}component} \tag{1}$$
$$\mathsf{po\text{-}member} \circ \mathsf{po\text{-}member} \sqsubseteq \mathsf{po\text{-}member} \tag{2}$$
$$\mathsf{po\text{-}portion} \circ \mathsf{po\text{-}portion} \sqsubseteq \mathsf{po\text{-}portion} \tag{3}$$
$$\mathsf{po\text{-}stuff} \circ \mathsf{po\text{-}stuff} \sqsubseteq \mathsf{po\text{-}stuff} \tag{4}$$
$$\mathsf{po\text{-}feature} \circ \mathsf{po\text{-}feature} \sqsubseteq \mathsf{po\text{-}feature} \tag{5}$$
$$\mathsf{po\text{-}place} \circ \mathsf{po\text{-}place} \sqsubseteq \mathsf{po\text{-}place} \tag{6}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}component}) \tag{7}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}member}) \tag{8}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}portion}) \tag{9}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}stuff}) \tag{10}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}feature}) \tag{11}$$
$$\mathrm{AsymmetricObjectProperty}(\mathsf{po\text{-}place}) \tag{12}$$
$$\mathsf{po\text{-}component} \sqsubseteq \mathsf{part\text{-}of} \tag{13}$$
$$\mathsf{po\text{-}member} \sqsubseteq \mathsf{part\text{-}of} \tag{14}$$
$$\mathsf{po\text{-}portion} \sqsubseteq \mathsf{part\text{-}of} \tag{15}$$
$$\mathsf{po\text{-}stuff} \sqsubseteq \mathsf{part\text{-}of} \tag{16}$$
$$\mathsf{po\text{-}feature} \sqsubseteq \mathsf{part\text{-}of} \tag{17}$$
$$\mathsf{po\text{-}place} \sqsubseteq \mathsf{part\text{-}of} \tag{18}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{19}$$
$$\mathrm{ReflexiveObjectProperty}(\mathsf{spatially\text{-}located\text{-}in}) \tag{20}$$
$$\mathsf{po\text{-}component} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{21}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}component} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{22}$$
$$\mathsf{po\text{-}member} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{23}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}member} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{24}$$
$$\mathsf{po\text{-}portion} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{25}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}portion} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{26}$$
$$\mathsf{po\text{-}stuff} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{27}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}stuff} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{28}$$
$$\mathsf{po\text{-}feature} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{29}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}feature} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{30}$$
$$\mathsf{po\text{-}place} \circ \mathsf{spatially\text{-}located\text{-}in} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{31}$$
$$\mathsf{spatially\text{-}located\text{-}in} \circ \mathsf{po\text{-}place} \sqsubseteq \mathsf{spatially\text{-}located\text{-}in} \tag{32}$$

**Fig. 1.** Pattern axioms for the first pattern variant from Section 3.

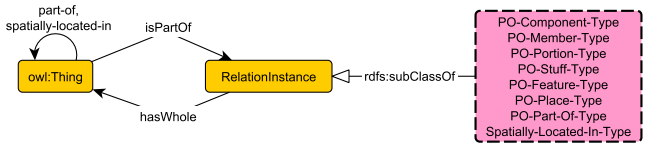



**Fig. 2.** Schema Diagram for the Contextualized Winston-Part-Whole Pattern. The dashed box on the right hand side lists seven different subclasses of PartWholeType. The subproperties of part-of from Section 3 are also used. Further explanations can be found in the text.

discussion, they do not really contribute to ontology modeling, and we do not want to include them in the pattern.

Please note that we do not provide a schema diagram for this pattern, as the pattern exists of related properties only.

## 4 A Pattern Extension Accounting for Provenance And Other Context Information

Some usages of the Winston-Part-Whole Pattern, such as the one from [8] on which this pattern is based, suggest that it would be helpful to store context information for the part-of relationship. We conceive that this would mostly be in the form of provenance information. For example, in the case of [8], part-of relationships of the various types defined by Winston were generated automatically using Hearst patterns over Web text corpora. In such a case, one may want to store confidence values, or even pointers to the exact algorithm used in each case.

In order to store this information, we now provide a contextualized version of the pattern described in Section 3; it is essentially obtained by "reifying" the properties. It is a known technique, and one could also refer to it as "lifting" or as "typecasting" of properties into classes following [13].

To explain, consider the schema diagram in Figure 2. A triple which according to the pattern in Section 3 would simply be stated as

```
:everglades   po:po-place    :florida .
```

would now be expressed using the following set of four triples—note that the original triple is still included. We use cpo as namespace, for "contextualized part-of."

```
:everglades   cpo:po-place     :florida ;
              cpo:isPartOf     :everglades-po-place-florida .
```

```
:everglades-po-place-florida   rdf:type       cpo:PO-Place-Type ;
                               cpo:hasWhole   :florida .
```

Additional context information, such as provenance information can then be attached to :everglades-po-place-florida, and we will further elaborate on this in Section 6.

We now show how to derive the axiomatization for the Contextualized Winston-Part-Whole Pattern. First of all, note that all axioms from Figure 1 are fully adopted (with adjusted namespace of course). In the following, let $R$ denote any one of po-component, po-member, po-portion, po-stuff, po-feature, po-place, and $C_R$ be the corresponding PO-Component-Type, ..., PO-Place-Type from Figure 2.

$$\mathsf{Po\text{-}Component\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{33}$$
$$\mathsf{Po\text{-}Member\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{34}$$
$$\mathsf{Po\text{-}Portion\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{35}$$
$$\mathsf{Po\text{-}Stuff\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{36}$$
$$\mathsf{Po\text{-}Feature\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{37}$$
$$\mathsf{Po\text{-}Place\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{38}$$
$$\mathsf{Po\text{-}Part\text{-}Of\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{39}$$
$$\mathsf{Spatially\text{-}Located\text{-}In\text{-}Type} \sqsubseteq \mathsf{RelationInstance} \tag{40}$$

Then we would like to have all of the following axioms, which are here expressed using rules.

$$\mathsf{isPartOf}(x, y) \wedge C_R(y) \wedge \mathsf{hasWhole}(y, z) \rightarrow R(x, z)$$

This rule actually constitutes a generalized role chain which can be cast into OWL using the *rolification*[5] technique described in [12]. The resulting OWL axioms are as follows (please note the lowercase $c_R$, which is the result of typecasting the class $C_R$ into a property).

$$C_R \equiv \exists c_R.\mathsf{Self} \tag{41}$$
$$\mathsf{isPartOf} \circ c_R \circ \mathsf{hasWhole} \sqsubseteq R \tag{42}$$

The same axioms would be added for spatially-located-inin place of $R$.

Note that instead of axioms (42), we would actually have preferred to use

$$\mathsf{isPartOf} \circ c_R \circ \mathsf{hasWhole} \equiv R,$$

however this is not expressible in OWL. According to [13] use of the latter axiom would be proper typecasting between properties and classes, however this requires right-hand-side property chains, which if added to OWL DL would cause

[5] The name *rolification* comes from the fact that properties are called *roles* in description logics [7].

undecidability and are therefore not included in the standard. Please see [13] for a further discussion of this matter. For similar reasons, we are not able to lift most axioms from Figure 1 fully to the contextualized pattern, as they would also result in right-hand-side property chains. In fact, in addtition to the 14 axioms above we have six axioms

$$R \sqsubseteq \mathsf{part\text{-}of},$$

which correspond to axioms (13) through (18). The asymmetry declarations from Figure 1 cannot be fully lifted to the contextualized version: to the best of our abilities, they cannot be expressed in OWL, and the same holds for the reflexivity axiom. For axioms (1) to (6), (21) through (32), and (19), partial liftings could be given. However, they would be redundant, i.e., inferrable through OWL DL reasoning from the axioms already given. We thus refrain from adding them.

$$\top \sqsubseteq \forall \mathsf{isPartOf}.\mathsf{RelationInstance} \tag{43}$$

$$\forall \mathsf{hasWhole}.\mathsf{RelationInstance} \sqsubseteq \top \tag{44}$$

Finally, we give the range and domain for isPartOf and hasWhole, (43) and (44), respectively. In total, we have 32 axioms inherited from the non-contextualized pattern, plus 30 new ones, for a total of 62 axioms for the Contextualised Winston-Part-Whole Pattern.

## 5 Usage Scenario

We give a usage scenario for the presented patterns, from the domain of Materials Science. Materials Science is an interdisciplinary field which focuses on the discovery and design of new or enhanced materials. Of central importance to the field is the determination of materials properties using experiment or modeling and simulation. Examples of such properties include ultimate tensile strength and crack growth rate. More data than ever is being generated as the materials science and engineering domain seeks to enhance throughput through the automation of sequential experiments and greater use of modeling and simulation [16]. At the same time, there is no widely accepted ontology we are aware of to facilitate the digital exchange and integration of data in this fast-growing and very active discipline. To start filling this gap, we have begun to investigate core ontology design patterns needed for such an ontology, and this in fact prompted our development of the Winston Part-Whole Patterns based on earlier mentioned work.

The important role of part-whole relations in this context comes from the fact that engineered products are usually created by combining previously created engineered products—and that includes engineered materials. For example, fiberglass and epoxy (glue) are part of a composite material.

Product designers seek materials which possess specific properties (e.g. color, strength) to enable a function (e.g. be atheistically pleasing, resist deformation due to mechanical loads). These properties are established by combining specific

materials in a particular way to achieve a certain microstructure. Once the processing is complete, the characteristic properties of the material are "locked-in." If the composition and structure of a material are described completely, a unique set of properties can be inferred. Additionally, since the processing can be associated with the composition and microstructure, it can also be associated with the unique set of properties. Thus, the recording of the parts or components of an engineered material is of importance.

Eventually, one would like to record the whole Part-Whole chain from a complex engineered product down to a very fine granularity. Examples for such relations could be the following.

- A radar system is part of a boat. – component-integral object
- An antennae radome is part of a radar system. – component-integral object
- Some composite material is part of an antennae radome. – stuff-object
- Epoxy is part of this composite material. – stuff-object
- Glass fiber is part of this composite material. – stuff-object
- Some composite material cure is part of some composite manufactoring. – feature-activity
- Some damaged area is part of some composite material surface. – place-area
- Some broken fiber is part of this damaged area. – component-integral object

It becomes apparent from these examples, that a naive approach, i.e., encoding all of these relationships using part-of only, is inferior to using a model based on Winston's work. E.g., in the former it would be incorrect, as duscussed, to declare part-of to be transitive, while our Winston Part-Whole Pattern allows for corresponding inferences where appropriate, e.g., from the above we could infer that An antennae radome is part of a boat (component-integral object) and that Glass fiber is part of an antennae radome (stuff-object).

## 6 A Provenance Pattern Derived From PROV-O

Provenance information is arguably among the most prominent types of context information for all kinds of data. We show in the following, how the Contextualized Winston-Part-Whole Pattern can be extended using a Provenance pattern which is derived from the core of PROV-O [5]. In a very similar way, other context information such as confidence values could be added.

The three core classes of PROV-O are Entity, Activity, and Agent. Briefly, an Entity is simply an item that has provenance. Entities are generated by Activities, which are the *execution* of some algorithm or method. The Activity or Entity may be performed by or attributed to some Agent which may be, for examples, a person or a script.

However, for use in the context of pattern-based modular ontology modeling [11], it is more convenient to have a dedicated pattern—rather than a full-blown ontology—at our disposal, although the pattern we provide is, essentially, the

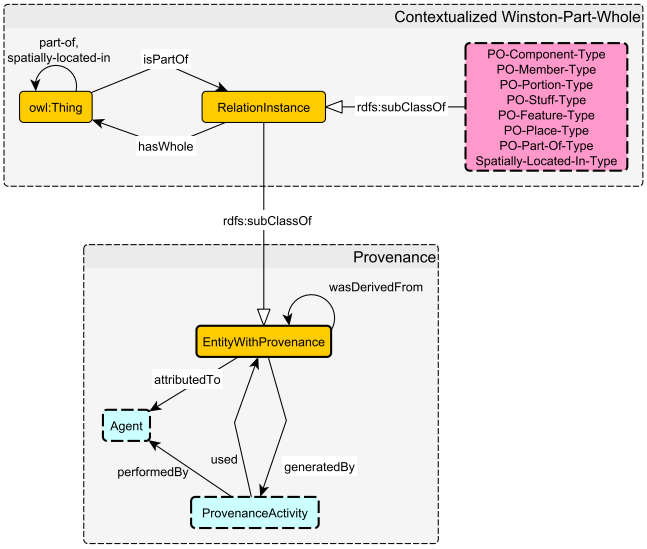



**Fig. 3.** Schema Diagram for the Contextualized Winston Part-Whole Pattern extended by a Provenance pattern following PROV-O.

core of PROV-O. We very simply align our extracted pattern to PROV-O via the following equivalences.

$$\mathsf{EntityWithProvenance} \equiv \mathsf{Entity}$$
$$\mathsf{ProvenanceActivity} \equiv \mathsf{Activity}$$

Figure 3 provides a graphical overview of this subpattern and how it may extend the Winston-Part-Whole Pattern. The following axioms specify the behavior of

this subpattern.

$$\begin{aligned}
\mathsf{EntityWithProvenance} &\sqsubseteq \forall \mathsf{wasDerivedFrom.EntityWithProvenance} \\
\forall \mathsf{attributedTo.Agent} &\sqsubseteq \mathsf{EntityWithProvenance} \\
\mathsf{EntityWithProvenance} &\sqsubseteq \forall \mathsf{attributedTo.Agent} \\
\forall \mathsf{generatedBy.ProvenanceActivity} &\sqsubseteq \mathsf{EntityWithProvenance} \\
\mathsf{EntityWithProvenance} &\sqsubseteq \forall \mathsf{generatedBy.ProvenanceActivity} \\
\forall \mathsf{used.EntityWithProvenance} &\sqsubseteq \mathsf{ProvenanceActivity} \\
\mathsf{ProvenanceActivity} &\sqsubseteq \forall \mathsf{used.EntityWithProvenance} \\
\forall \mathsf{performedBy.Agent} &\sqsubseteq \mathsf{ProvenanceActivity} \\
\mathsf{ProvenanceActivity} &\sqsubseteq \forall \mathsf{performedBy.Agent}
\end{aligned}$$

We add some explanations of these axioms, they follow the standard templates of scoped domain and range restrictions.

1. The scoped range of wasDerivedFrom, scoped by EntityWithProvenance, is EntityWithProvenance.
2. The scoped domain of attributedTo, scoped by Agent, is EntityWithProvenance.
3. The scoped range of attributedTo, scoped by EntityWithProvenance, is Agent.
4. The scoped domain of generatedBy, scoped by ProvenanceActivity, is EntitWithProvenance.
5. The scoped range of generatedBy, scoped by EntityWithProvenance, is ProvenanceActivity.
6. The scoped domain of used, scoped by EntityWithProvenance, is ProvenanceActivity
7. The scoped range of used, scoped by ProvenananceActivity, is EntityWithProvenance.
8. The scoped domain of performedBy, scoped by Agent, is ProvenanceActivity.
9. The scoped range of performedBy, scoped by ProvenanceActivity, is Agent.

Of course, pairs of different entities with provenance, or different agents, or different provenance activities, may in turn carry part-whole relationships, which could be expressed using Contextualized Winston-Part-Whole Pattern.

## 7 Pattern Release Information

We have released the Winston-Part-Whole Pattern,[6] the Contextualized Winston-Part-Whole Pattern, [7], and the Provenance Pattern[8] in OWL/XML syntax on the ontologydesignpatterns.org portal.

[6] `https://ontologydesignpatterns.org/wiki/Submissions:WinstonPartWhole`
[7] `https://ontologydesignpatterns.org/wiki/Submissions:ContextualizedWinstonPartWhole`
[8] `http://ontologydesignpatterns.org/wiki/Submissions:Provenance`

In addition, we have annotated the patterns with the appropriate annotations following the OPLa ontology which serves as ontology design pattern representation language [6]. The annotations were generated using the OPLa plugin for Protégé [17].

## 8 Conclusion

Part-whole relations are omnipresent and are fundamental to how we organize information and perceive the world. Thus, it is necessary to have a firm understanding of how to model these *partonomies* or *meronomies*. To do so, we have followed Winston's approach, as discussed in [20] and as a result, have developed two patterns: the Winston-Part-Whole Pattern and the Contextualized Winston-Part-Whole Pattern. Additionally, we provide a mechanism for augmenting the pattern with provenance.

*Acknowledgements.* Cogan Shimizu acknowledges funding from the Dayton Area Graduate Studies Institute.

## References

1. Artale, A., Franconi, E., Guarino, N., Pazzi, L.: Part-whole relations in object-centered systems: An overview. Data & Knowledge Engineering 20(3), 347–383 (1996)
2. Casati, R., Varzi, A.: Parts and places: The structures of spatial representation. The MIT Press (1999)
3. Girju, R., Badulescu, A., Moldovan, D.: Learning semantic constraints for the automatic discovery of part-whole relations. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. pp. 1–8. Association for Computational Linguistics (2003)
4. Girju, R., Badulescu, A., Moldovan, D.: Automatic discovery of part-whole relations. Computational Linguistics 32(1), 83–135 (2006)
5. Groth, P., Moreau, L. (eds.): PROV-Overview: An Overview of the PROV Family of Documents. W3C Working Group Note 30 April 2013 (2013)
6. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V.: Towards a simple but useful ontology design pattern representation language. In: Blomqvist, E., et al. (eds.) Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) Vienna, Austria, October 21, 2017. CEUR Workshop Proceedings, vol. 2043. CEUR-WS.org (2017), http://ceur-ws.org/Vol-2043/paper-09.pdf
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Jain, P., Hitzler, P., Verma, K., Yeh, P.Z., Sheth, A.P.: Moving beyond SameAs with PLATO: partonomy detection for linked data. In: Munson, E.V., Strohmaier, M. (eds.) 23rd ACM Conference on Hypertext and Social Media, HT '12, Milwaukee, WI, USA, June 25-28, 2012. pp. 33–42. ACM (2012)

9. Jain, P., Yeh, P.Z., Verma, K., Henson, C.A., Sheth, A.P.: SPARQL query rewriting using partonomy based transformation rules. In: Janowica, K., Raubal, M., Levashkin, S. (eds.) Proceedings of the 3rd International Conference on GeoSpatial Semantics. pp. 140–158. GeoS '09, Springer-Verlag, Berlin, Heidelberg (2009)
10. Keet, C.M., Kutz, O.: Orchestrating a network of mereo(topo)logical theories. In: Corcho, Ó., Janowicz, K., Rizzo, G., Tiddi, I., Garijo, D. (eds.) Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017. pp. 11:1–11:8. ACM (2017)
11. Krisnadhi, A., Hitzler, P.: Modeling with ontology design patterns: Chess games as a worked example. In: Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.) Ontology Engineering with Ontology Design Patterns – Foundations and Applications, Studies on the Semantic Web, vol. 25, pp. 3–21. IOS Press (2016)
12. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and Rules. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P.F. (eds.) Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer, Heidelberg (2011)
13. Krisnadhi, A.A., Hitzler, P., Janowicz, K.: On the capabilities and limitations of OWL regarding typecasting and ontology design pattern views. In: Tamma, V.A.M., Dragoni, M., Gonçalves, R.S., Lawrynowicz, A. (eds.) Ontology Engineering – 12th International Experiences and Directions Workshop on OWL, OWLED 2015, co-located with ISWC 2015, Bethlehem, PA, USA, October 9-10, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9557, pp. 105–116. Springer (2015)
14. Krötzsch, M., Maier, F., Krisnadhi, A.A., Hitzler, P.: A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In: Sadagopan, S., Ramamritham, K., Kumar, A., Ravindra, M., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International World Wide Web Conference, WWW2011, Hyderabad, India, March/April 2011. pp. 645–654. ACM, New York (2011)
15. Motik, B., Patel-Schneider, P., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), available at http://www.w3.org/TR/owl2-syntax/
16. Nikolaev, P., Hooper, D., Perea-Lopez, N., Terrones, M., Maruyama, B.: Discovery of wall-selective carbon nanotube growth conditions via automated experimentation. ACS Nano 8(10), 10214–10222 (2014)
17. Shimizu, C., Hirt, Q., Hitzler, P.: A Protégé plugin for annotating OWL ontologies with OPLa. The Semantic Web: ESWC 2018 Satellite Events (2018), to appear.
18. Tryfona, N., Egenhofer, M.J.: Consistency among parts and aggregates: A computational model. Transactions in GIS 1(3), 189–206 (1996)
19. Varzi, A.: Parts, wholes, and part-whole relations: The prospects of mereotopology. Data & Knowledge Engineering 20(3), 259–286 (1996)
20. Winston, M.E., Chaffin, R., Herrmann, D.: A taxonomy of part-whole relations. Cognitive Science 11(4), 417–444 (1987)

# Extensions to the Ontology Design Pattern Representation Language*

Quinn Hirt[1], Cogan Shimizu[1,3], and Pascal Hitzler[1,3]

[1] Data Semantics Laboratory, Wright State University, Dayton, OH, USA
[2] Data Semantics Laboratory, Kansas State University, Manhattan, KS, USA

**Abstract.** Recently, modular ontology modeling has become a more popular ontology engineering paradigm. With it, the need for additional metadata associated with ontology design patterns has grown. The Ontology Pattern Language (OPLa) was developed to facilitate annotating ontologies with useful metadata, as well as supporting tooling infrastructure. In this paper, we detail three extensions to OPLa into a reorganized namespace: OPLa-core, containing the original annotations; OPLa-SD, for use in detailing schema diagrams; and OPLa-CP, an adaptation of the content-pattern annotation schema.

## 1 Introduction

As the pattern-based, modular ontology engineering methodology has matured, it has become apparent that additional metadata describing that patterns and modules that compose the modular ontology is necessary. Indeed, we have seen that the documentation of an ODP has three facets which are espeically useful during development: structural and provenance, content and usage, and visualization.

Structural and provenance documentation refers to those annotations that describe how patterns and modules may interact with each other, as well as related patterns, how they connect, or their most general form. Recently, [2] presented the Ontology Design Pattern Representation Language (OPLa) to cover this use case.

Content and usage has been historically covered by the cp-annotation-schema; these metadata annotations are critical for the discovery, reuse, and interoperability with existing patterns. When a pattern is submitted to the ontology design patterns web portal[3], the form requires that the pattern be annotated with the cp-annotation-schema.[4] This schema includes a number of annotation properties that allow a developer to express certain usage characteristics of their *content* ontology design pattern. However, it is not always clear how to use these annotations and may pose as an obstacle for advanced querying.



[3] http://ontologydesignpatterns.org
[4] url2cp-annotation.owl

Table 1: The prefixes and namespaces for the reorganized OPLa.

| Prefix | Namespace |
|---|---|
| opla-core: | `http://ontologydesignpatterns.org/opla-core#` |
| opla-cp: | `http://ontologydesignpatterns.org/opla-cp#` |
| opla-sd: | `http://ontologydesignpatterns.org/opla-sd#` |

Furthermore, we have seen that schema diagrams are an important tool for understanding the purpose, content, and structure of an ontology design pattern or module [6, 3]. A schema diagram is a graph-based visual diagram that gives a human understandable view of the stucture that concepts, properties, and datatypes create within the onotlogy. Further, it does not aim to represent the full logical implications of each axiom in the ontology; instead, it focuses on representing simple relationships between concepts in an ontology. Graphical ontology modelling is not a new paradigm, but with increasing accessibility of multimodal input devices, as well as increased adoption in non-ontologists, it has recently become more prevalent [8, 1, 4]. Modular ontology modelling seems to be an excellent complement to graphical ontology modelling, given that its "plug-and-play" nature may easily correspond to "drag-and-drop." Metadata describing best-practices for visualizing a pattern, or set of patterns, will certainly be of benefit to developers of tooling infrastructure. While only three annotations have been specified in OPLa-sd, defined in a later section, additional annotations to support future tooling applications and infrastructure will be added.

To support these needs, this paper describes the extension and slight reorganization of OPLa into three distinct sub-namespaces. Table 1 shows these new namespaces. Thus, we enable a simplified documentation process, as well as provide multiple dimensions for pattern discovery. The contributions are as follows.

1. OPLa-Core: the new namespace of the original OPLa ontology and the annotations therein remain unchanged.
2. OPLa-CP: contains the annotation properties describing a number of non-technical facets on the usage of a particular ontology design pattern or module. These annotation properties are adapted from the cp-annotation-schema.
3. OPLa-SD: the foundational positional axioms used for expressing visual location of an entity on a canvas.

## 2 Extending OPLa

As mentioned in the previous section, there are three distinct representations of a pattern that we wish OPLa to cover. Furthermore, we wish to have each of the annotations to support discoverability, as such, some disambiguation of the uses of the annotations was necessary. In the next sections we briefly discuss the

usage characteristics of the new OPLa-cp annotations and two annotations from OPLa-core. The updated OWL file for these OPLa extensions (as well as the original specification) is available online.[5]

### 2.1 OPLa-core

In this section, we merely disambiguate some common questions pertaining to two original OPLa annotations and a new core annotation, as adapted from cp-annotation-schema. While some of these annotations were in the original specifications for OPLa, their use-cases were not explicit.

opla-core:isNativeTo should be used to express the provenance of some ontological entity. This annotation is not functional, e.g. some entity may be native to more than one pattern, e.g. the Agent may belong to both AgentRole and Provenance patterns.

opla-core:ofExternalType should be used to indicate that another pattern may hook into this entity. For example, a hook might indicate that the pattern developer acknowledges a certain concept is out of scope of the particular pattern.

opla-core:extractedFrom is adapted from cp-annotation-schema This annotation should be used to indicate that a pattern has been created from where one originally did not exist. Otherwise, opla-core:derivedFrom should be used.

### 2.2 OPLa-cp

The adaptation of cp-annotation-schema is the primary contribution in this paper. The purpose of the cp-annotation-schema is to describe many aspects pertaining to the usage of an ontology design pattern, thus we have adapted many of the annotations for use in OPLa. However, not all of the annotations in cp-annotation-schema were necessary to be included, as their purposes were covered by existing OPLa annotations. Additionally, few of the original annotations implied that their payload was plural, e.g. hasConsequences. However, in order to aid in discovery and minimze complicated text-processing, we felt that it was more natural to have singular natural language payloads, i.e. hasConsequence.

opla-cp:hasConsequence Describes a potential gain and drawback when using the annotated module or pattern. For example, it may be used to express the impact of an ontological commitment.

opla-cp:coversRequirement Should point to a blank node, wherein a competency question and SPARQL query should be paired. This pair should pretain to a question answerable by the original pattern itself without being instantiated for a specific usecase.

opla-cp:hasCompetencyQuestion Points to an example competency question that can be evaluated against the annotated ontology, as expressed in natural language.

[5] `https://github.com/cogan-shimizu-wsu/Extended-OPLa/`

```
:Agent rdf:type owl:Class ;
  opla-sd:entityPosition
    [ opla-sd:entityPositionX "19.608958656638492"^^xsd:double ;
      opla-sd:entityPositionY "89.84401282241834"^^xsd:double ;
      rdfs:comment "This is an entity positioning annotation
      generated by CoModIDE (https://comodide.com/). Removing
      this annotation will break rendering the CoModIDE
      schema diagram view."@en
    ] .
```

Fig. 1: An excerpt of the AgentRole pattern showcasing the opla-sd position annotations.

opla-cp:hasUnitTest Points to an example SPARQL query that can be evaluated against the content of the annotated ontology. It should be paired with a natural language description, e.g. Competency Question.

opla-cp:addressesScenario Describes a potential or existing usecase or instantiation of the ontology design pattern. It should describe the commitments necesarry to make the instantiation possible.

### 2.3 OPLa-sd

OPLa-SD includes a set of annotations specifically for tooling software to query and utilize. Currently, there are only three annotations belong to OPLa-SD, as detailed below. Figure 1 shows an excerpted opla-sd:EntityPosition from the annotated version of the AgentRole pattern in [7]. These annotations are not meant to be manipulated directly by a user and doing so can cause tools to break.

opla-sd:entityPosition This property has a blank node as its target. This blank node is intended to "encapsulate" all other position related annotations.

opla-sd:entityPositionX This property specifies the X coordinate (expressed as a double) of a node in a schema diagram.

opla-sd:entityPositionY This property specifies the Y coordinate (expressed as a double) of a node in a schema diagram.

## 3 Conclusions

This paper describes the reorganization of the OPLa namespace into three new sub-namespaces, OPLa-core, OPLa-CP, and OPLa-SD. This reorganization, and the population of the new sub-namespaces, supports the documentation of ODPs, in three critical facets, as well providing descriptions to annotations previously without documentation. Each namespace has a set of annotations for specialized roles and functionality.

These new OPLa annotations can be seen in two ongoing projects: MODL: a modular ontology design library [7], where it is used to index design patterns for use; and the Comprehensive Modular Ontology IDE (CoModIDE[6] [5]), which leverages both MODL and its index to enable pattern-based modular ontology engineering and uses the OPLa-SD annotation properties in order to keep a consistent rendering of the ontology across sessions.

*Acknowledgement.* Quinn Hirt acknowledges funding from the Air Force Office of Scientific Research under award number FA9550-18-1-0386. Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI).

## References

1. K. Hammar. Ontology design patterns in WebProtégé. In S. Villata, J. Z. Pan, and M. Dragoni, editors, *Proceedings of the ISWC 2015 Posters & Demonstrations Track*. CEUR-WS.org, 2015.
2. 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, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017)*. CEUR-WS.org, 2017.
3. N. Karima, K. Hammar, and P. Hitzler. How to document ontology design patterns. In K. Hammar, P. Hitzler, A. Lawrynowicz, A. Krisnadhi, A. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 15–28. IOS Press, Amsterdam, 2017.
4. M. K. Sarker, A. A. Krisnadhi, and P. Hitzler. OWLAx: A protégé plugin to support ontology axiomatization through diagramming. In T. Kawamura and H. Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track*. CEUR-WS.org, 2016.
5. C. Shimizu. Towards a comprehensive modular ontology IDE and tool suite. In S. Kirrane and L. Kagal, editors, *Proceedings of the Doctoral Consortium at ISWC 2018.*, volume 2181 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2018.
6. C. Shimizu, A. Eberhart, N. Karima, Q. Hirt, A. Krisnadi, and P. Hitzler. A method for automatically generating schema diagrams for modular ontologies. In *1st Iberoamerican Conference on Knowledge Graphs and the Semantic Web*, 2019. To Appear.
7. C. Shimizu, Q. Hirt, and P. Hitzler. Modl: A modular ontology design library. Technical report, Wright State University, Dayton, Ohio, April 2019.
8. V. Wiens, S. Lohmann, and S. Auer. WebVOWL Editor: Device-Independent Visual Ontology Modeling. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks*. CEUR-WS.org, 2018.

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

# A Protégé Plug-In for Annotating OWL Ontologies with OPLa

Cogan Shimizu, Quinn Hirt, and Pascal Hitzler

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

**Abstract.** The Ontology Engineering community has recognized needs for both a simple, exstensible representation language for patterns and tools that support such workflows. In this demonstration, we describe a Protégé plugin that guides a user in documenting the loaded OWL ontology and its entities with annotations from the Ontology Design Pattern Representation Language (OPLa).

## 1 Motivation

The use of ontology design patterns (ODP) has established itself as an ontology engineering paradigm [3]. There are, however, a number of open challenges to be considered by researchers concerning the future of ODPs and modular ontology engineering. In this demonstration, we are particularly interested in both recognizing the substantial need for a robust pattern representation language and increasing the availability of easy-to-use, supporting tools. For a more thorough examination of these challenges and others, please see [2].

Utilizing a pattern representation language is an important piece for improving the development process, as it begins to address a perennial challenge in the Semantic Web community: ontology sharing and reuse; and as it applies to the ontology engineer: *ontology design pattern* sharing and reuse. A commonly used pattern representation language is immediately impactful by allowing the ontology engineer to more explicitly express

- how to use a pattern (e.g. what are natural "hooks" into the ODP)
- which other ODPs have been adapted or reused to create the pattern
- from where an ontology module was derived

Together, these examples can enable a so-called "smart" repository. Such a repository will allow an ontology engineer to more easily navigate and explore patterns and modules, thus realizing a centralized mechanism for the sharing, reuse, or adaptation of ODPs. As such, it is in the best interest of the community to utilize the pattern language.

As a first step in addressing this challenge, [4] presented the Ontology Design Pattern Representation Language (OPLa). OPLa annotations are fully compatible with OWL and its semantics are formally described. For each of our motivating examples we provide some concrete examples that illustrate exactly how OPLa can be used to formally describing those relationships.

The MicroblogEntry pattern may contain the following triples:

```
mbe:Location  opla:ofExternalType           opla:externalClass .
mbe:Media     opla:reusesPatternAsTemplate  nre:Media .
```

The first triple indicates that the Location is a "hook" for other engineers to use. The second triple indicates that the MicroblogEntry pattern has adapted the Media class from another ODP, in this case the NewsReportingEvent ODP [7, 5]. Additionally, the ModifiedHazardousSituation ODP may contain the triple

```
mhs:    opla:derivedFromPattern    hs: .
```

which states that ModifiedHazardousSituation is derived from the HazardousSituation ODP [1, 6]. Figure 1 provides a graphical overview for the other OPLa annotations.

However, like many forms of documentation, it is a tedious task to exhaustively perform. The key to adding complexity to any engineering process is making the change trivial to the end-user. As such, sufficient, easy-to-use tooling is necessary for facilitating process adoption. To address this, we have developed a Protégé plug-in that has been optimized for walking an ontology engineer through annotating their ontology, module, or pattern with the correct OPLa annotations.

## 2 Implementation

Our plugin, the OPLaTab (Figure 2), is implemented for Protégé 5. At the time of this writing, plugin registration through the Protégéwiki[1] is ongoing. We provide a portal online[2] for a more detailed examination of the plugin's source code and .JAR file, and closer view of the interface and its use and installation.

The purpose of the OPLaTab plugin is to guide the user through the construction of a valid OPLa annotation. As such, it is optimized for annotating the ontology itself and its entities with only those annotations explicitly outlined in Figure 1 and [4]. Thus, the interface is purposefully minimalist and restricted: it condenses all annotation functionality to a single screen and reduces the number of choices a user needs to make in order to insert an annotation into the ontology.

The tab's only silent behavior is to add the OPLa namespace to the ontology.[3] All other changes to the ontology are done via the "Save" and "Remove" buttons.

The interface is separated into three parts: navigation, construction, and view/remove. The plugin currently supports the annotation of the Ontology, Classes, Individuals, Object Properties, Data Properties, Datatypes and Annotations. By selecting one of these options, the construction area will be populated with the appropriate entities. Further, the list of annotation properties will update to display only those properties that are valid for the selected entity. Finally, the view/remove area will display only OPLa annotations so that it is easy to identify which entities have yet to be annotated.

[1] https://protegewiki.stanford.edu/wiki/Main_Page
[2] http://dase.cs.wright.edu/content/oplatab
[3] http://ontologydesignpatterns.org/opla/

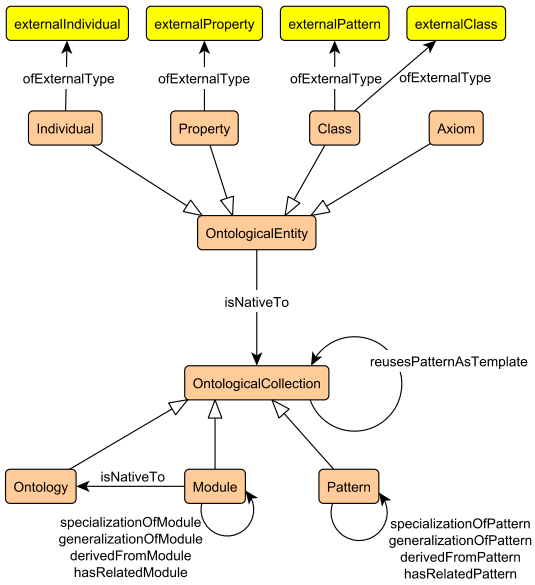



Fig. 1: A graphical overview of the Ontology Design Pattter Representation Language (OPLa) [4].

Currently, the annotation value for the annotation is user-dependent, in the absence of a standardized controlled vocabulary or repository. We briefly describe a sample workflow:

1. Select the ontology itself or an ontological entity.
2. Select the appropriate annotation property.
3. Enter the annotation value.
4. Save the annotation.

At this time, the bottom portion of the screen will update with the new annotation. The annotation may be removed by selecting the appropriate button.

## 3 Conclusions, Future Work, & Demonstration

OPLaTab is a useful tool for constructing OPLa annotations. There is currently an ongiong intention to develop a comprehensive tool suite for modular ontology

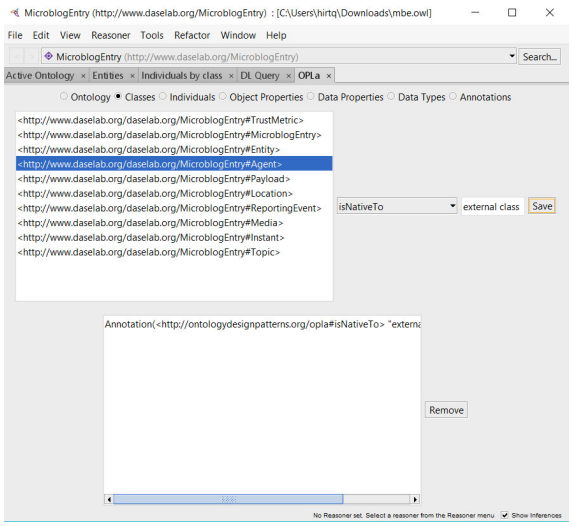



Fig. 2: A view of OPLaTab's interface. This view shows an example annotation be added to the loaded ontology.

engineering. We see OPLa becoming a central component of this tool suite. From visualization and interactive browsing to a smart repository, each will need some way of communicating to a user exactly how ontologies and ODPs relate to each other. As we provide more sophisticated tools in this realm, there are several foreseeable next steps.

While some of these next steps will require extensions to OPLa, OPLa was purposefully developed to be easily extendable. For example, it may be possible to embed visualization information into annotations, allowing software to determine which properties are visible at different levels of granularity. The same principle can be extended for interactive browsing. Perhaps most importantly, however, is the ability to connect to a machine-readable repository of ontology design patterns and modules. This "smart" repository can act as a dynamically updated controlled vocabulary of namespaces, thus allowing an ontology engineer to select the appropriate namespace when constructing their OPLa annotations.

Additionally, we will work to improve the user experience and look to more appropriately match the Protégé workspace.

**Demonstration**

The demonstration will consist of a live walkthrough of annotating a loaded ontology. In the interest of space, we have outlined in more detail a step-by-step walkthrough in our online portal.[4]

*Acknowledgement.* Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI).

## References

1. M. Cheatham, H. Ferguson, C. Vardeman, and C. Shimizu. A modification to the hazardous situation ODP to support risk assessment and mitigation. In K. Hammar, P. Hitzler, A. Krisnadhi, A. Lawrynowicz, A. G. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 97–104. IOS Press, 2017.
2. 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, 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 189–198. IOS Press, 2016.
3. 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*. 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, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
5. E. Kowalczuk and A. Lawrynowicz. The reporting event odp and it's extension to report news events. In K. Hammar, P. Hitzler, A. Krisnadhi, A. Lawrynowicz, A. G. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 105–117. IOS Press, 2017.
6. A. Lawrynowicz and I. Lawniczak. The hazardous situation ontology design pattern. In E. Blomqvist, P. Hitzler, A. Krisnadhi, T. Narock, and M. Solanki, editors, *Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pensylvania, USA, October 11, 2015*, volume 1461 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.
7. C. Shimizu and M. Cheatham. An ontology design pattern for microblog entries. In E. Blomqvist, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[4] `http://dase.cs.wright.edu/content/oplatab`

# MODL: A Modular Ontology Design Library*

Cogan Shimizu[1], Quinn Hirt[1], and Pascal Hitzler[1,2]

[1] Data Semantics Laboratory, Wright State University, Dayton, OH, USA
[2] Data Semantics Laboratory, Kansas State University, Manhattan, KS, USA

**Abstract.** Pattern-based, modular ontologies have several beneficial properties that lend themselves to FAIR data practices, especially as it pertains to Interoperability and Reusability. However, developing such ontologies has a high upfront cost, e.g. reusing a pattern is predicated upon being aware of its existence in the first place. Thus, to help overcome these barriers, we have developed MODL: a modular ontology design library. MODL is a curated collection of well-documented ontology design patterns, drawn from a wide variety of interdisciplinary use-cases. In this paper we present MODL as a useful resource for the development of high-quality, modular ontologies, discuss its use, and provide some examples of its contents.

## 1 Introduction

The Information Age is an apt description for these modern times; between the World Wide Web and the Internet of Things an unfathomable amount of information is accessible to humans and machines, but the sheer volume and heterogeneity of the data have their drawbacks. Humans have difficulty drawing *meaning* from large amounts of data. Machines can parse the data, but do not *understand* it. Thus, in order to bridge this gap, data would need to be organized in such a way that some critical part of the human conceptualization is preserved. Ontologies are a natural fit for this role, as they may act as a vehicle for the sharing of *understanding* [5].

Unfortunately, published ontologies have infrequently lived up to such a promise, hence the recent emphasis on FAIR (Findable, Accessible, Interoperable, and Reusable) data practices [24]. More specifically, many ontologies are not interoperable or reusable. This is usually due to incompatible ontological commitments: strong—or very weak—ontological committments lead to an ontology that is really only useful for a specific use-case, or to an ambiguous model that is almost meaningless by itself.

To combat this, we have developed a methodology for developing so-called modular ontologies [14]. In particular, we are especially interested in pattern-based modules [11]. A modularized ontology is an ontology that individual users can easily adapt to their own use-cases, while still preserving relations with other

versions of the ontology; that is, keeping it *interoperable* with other ontologies. Such ontologies may be so adapted due to their "plug-and-play" nature; that is, one module may be swapped out for another developed from the same pattern.

An ontology design pattern is, essentially, a small self-contained ontology that addresses a general problem that has been observed to be invariant over different domains or applications [9]. By tailoring a pattern to a more specific use-case, an ontology engineer has developed a *module*. This modelling paradigm moves much of the cost away from the formalization of a conceptualization (i.e. the logical axiomatization). Instead, pattern-based modular ontolody design (PBMOD) is predicated upon knowledge of available patterns, as well as being aware of the use-cases it addresses and its ontological commitments.

Thus, in order to address the findability and accessibility aspects of PBMOD, we have developed MODL: a modular ontology design library. MODL is a curated collection of well-documented ontology design patterns. The particular research contribution is both the curation and documentation. Some of the patterns are novel, but many more have been extracted from existing ontologies and streamlined for use in a general manner. MODL, as an artefact, is distributed online as a collection of annotated OWL files and a technical report containing schema diagrams and explanations of each OWL axiom.[3]

The rest of the paper is organized as follows. Section 2 discusses the relevance of this work. Section 3 presents our Modular Ontology Design Library in detail and in Section 4 we conclude and discuss future work.

## 2 Relevance

Pattern-based modular ontology development is not a conceptually new idea—instead, it is a continuation of an already established paradigm. Both modularization of ontologies [17] and pattern-based modelling [4] have been identified as improvements to the ontology engineering process. These concepts have culminated in mature paradigms (e.g. MOM [14, 11] and eXtreme Design [16, 1]), both having been used in large-scale projects (e.g. GeoLink [15] and VALCRI [3]). However, the ontology engineering community, especially those that utilize patterns, have indicated an increased need for better tooling support [7, 2], of which there are two complementary aspects: a dedicated development environment and a critical mass of Ontology Design Patterns (or ODPs for short).[4]

There is already a prototype that begins to address the need for a dedicated development environment for pattern-based ontologies [6]. It also provides a set of hard-coded patterns that were extracted (at the time of development) from the ODP Portal.[5] However, having the pattern library tightly coupled with the

[3] `https://dase.cs.wright.edu/content/modl-modular-ontology-design-library`

[4] Anecdotally, one of the more pervasive themes at both the 2018 and 2019 United States Semantic Technologies Symposia (`https://us2ts.org/`) was a call from ontology engineers in both academia and industry for better tooling support.

[5] `http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs`

tool is disadvantageous for future development. Indeed, decoupling the tool is desirable, for a number of reasons, as follows.

- Remove the onus of pattern development and upkeep off of the tool developer.
- Enable community driven improvements and tailoring of the library to the end-users use-cases.
- Enable plug-and-play pattern libraries for different domains, etc.

On the other hand, MODL also addresses the crucial need for a critical mass of ODPs. One may argue that this critical mass exists in the form of the ODP portal. Unfortunately, though, it has suffered under the weight of its own mission. Community enforced quality control has not succeeded in providing a ready-to-use suite of quality patterns for use across multiple domains.

Furthermore, while the quality of a set of patterns is largely subjective, MODL strives for consistency in documentation, uses best practices [13, 10], and limited ontological commitments. In some cases this required polishing extant documentation, writing it from scratch, and tweaking or detecting errors in the formalization. We also include all new schema diagrams [12] following a single paradigm and style.

MODL therefore addresses, in some fashion, both aspects of improving tooling support. In turn, we expect this to lower the barrier of entry to PBMOD, which in turn lowers the barrier of entry for wider adoption semantic web technologies in application areas.

## 3 A Modular Ontology Design Library

In this section, we present in detail MODL. Section 3.1 explains our methodology and the organization of MODL, Section 3.2 provides a brief overview on the anticipated usecases of MODL, Section 3.3 provides an example pattern that has been excerpted from the documentation (some of the language and structure, e.g. subsections, have been adapted to fit this paper format), and finally, Section 3.4 provides information pertaining to accessibility, sustainability, and more.

### 3.1 MODL's Methodology

MODL is a curated collection of well-documented ontology design patterns. MODL, itself, can be considered to be the combination of two artifacts, the collection of patterns, specified in OWL, and the accompanying documentation. The separation is a little fuzzy, as the OWL serialization is also heavily annotated for convenience. The mission of MODL is to make patterns both findable and accessible. Therefore, it is of utmost importance that every pattern therein is thoroughly documented. One drawback of the ODP Portal is that there are no guidelines provided for documenting the patterns and, during submission, a form is provided with many optional, ill-defined fields. That is not to say all of the patterns documented therein are poorly documented—some patterns did indeed

have thorough documentation. Indeed, we would like to emphasize that the main contribution of MODL is not the patterns in and of themselves. The ODP portal and many of the included patterns are well-known, well-used, and grounded in literature. Where possible, we preserved these efforts, from either the portal or associated publication, and corresponding credit is given in the MODL documentation. Links to ancestor patterns are included in both the annotations and documentation.

However, for many of the patterns included in MODL, we needed to fill some gaps. For this we have elected to follow the guidelines set forth in [13]. These guidelines are a result of a community wide survey that ranks the perceived importance of ten different components of ODP documentation. For our purposes, we have chosen to include the top seven. They are *Schema Diagram*, *Example of Pattern Instantiation*, *Compentency Questions*, *Axiomatization*, *OWL File*, *Pointers to Related Patterns*, and *Metadata*. The remaining three components (*Set of Example SPARQL Queries*, *Examples of Available Datasets for Population*, and *Constraints Using ShEx*[6]) are being considered for future versions of MODL.[7]

The schema diagrams for our documentation were manually created using the algorithm found in [12, 21]. We elected to use a simplified visual syntax that conveyed relations between concepts and also contains visual cues for identifying concepts that should be used as *hooks* into the ODP. In this case, a "hook" is some concept that is not fully fleshed out by the pattern, but recognizes that there is some relation at some level. This hook can be another pattern, a module, or a stub (described in more detail later).

The provided OWL files for each of the patterns are annotated with the Extended Ontology Design Pattern Representation Language (OPLa)[8] [10]. This allows us to embed provenance metadata (e.g. where did this pattern originate?) or provide pointers to related patterns (e.g. generalizations or specializations of the pattern) in annotations.

Finally, each pattern uses the namespace associated with the persistent URI for this resource[9]. However, the patterns contained inside of MODL are intended to be used as templates [8]. Further, the patterns in a MODL-like pattern library are meant to be local and the collection bespoke to the domain. MODL itself is meant as both example and seed. As such, the pattern URIs are not intended to be resolvable. By instantiating a pattern or making a module, the original pattern namespace becomes inconsequential. However, we acknowledge that this may be a perspective that runs counter to some established view points; thus, as MODL matures, we intend to include redirects to landing pages (e.g. using

[6] `http://shex.io/`

[7] Furthermore, there is some community indecision on embracing ShEx or SHACL, a newer W3C recommendation. More information can be found at `https://www.w3.org/TR/shacl/`.

[8] `https://github.com/cogan-shimizu-wsu/Extended-OPLa`

[9] `https://archive.org/services/purl/purl/modular_ontology_design_library`

| Category | Patterns |
|---|---|
| Metapatterns | Explicit Typing<br>Property Reification<br>Stubs |
| Organization of Data | Aggregation, Bag, Collection<br>Sequence, List<br>Tree |
| Space, Time, and Movement | Spatiotemporal Extent<br>Spatial Extent<br>Temporal Extent<br>Trajectory<br>Event |
| Agents and Roles | AgentRole<br>ParticipantRole<br>Name Stub |
| Description and Details | Quantities and Units<br>Partonymy/Meronymy<br>Provenance<br>Identifier |

Table 1: This table contains the patterns included in MODL. They have been partitioned into five categories (metapatterns; organization of data; space, time, and movement; agents and roles; and description and details) which are loosely defined by their general use-cases.

WiDoCo or similar) in the purl service via content negotiation, as it will certainly further improve usability.

Table 1 lists the patterns included in MODL. They have been loosely organized into five categories: metapatterns; organization of data; space, time, and movement; agents and roles; and description and details.

**Metapatterns** This category contains patterns that can be considered to be "patterns for patterns." In other literature, notably [4], they may be called *structural ontology design patterns*, as they are independent of any specific context, i.e. they are content-independent. This is particularly true for the metapattern for property reification, which, while a modelling strategy, is also a workaround for the lack of $n$-ary relationships in OWL. The other metapatterns address structural design choices frequently encountered when working with domain experts. They present a best practice to non-ontologists for addressing language specific limitations. In general, these patterns are not meant to be truly instantiated. One use of these patterns would be to utilize their axioms as a guide.

**Organization of Data** This category contains patterns that pertain to how data might be organized. These patterns are necessarily highly abstract, as they are ontological reflections of common data structures in computer science. The pattern for aggregation, bag, or collection is a simple model for connecting many concepts to a single concept. Analogously, for the list and tree pattterns, which

aim to capture ordinality and acyclicity, as well. More so than other patterns in this library, these patterns provide an axiomatization as a high-level framework that must be specialized (or modularized) to be truly useful.

**Space, Time, and Movement** This category contains patterns that model the movement of a thing through a space or spaces and a general event pattern. The semantic trajectory pattern is a more general pattern for modelling the discrete movements along some dimensions. The spatiotemporal extent pattern is a trajectory along the familiar dimensions of time and space. Both patterns are included for convenience.

**Agents and Roles** This category contains patterns that pertain to agents interacting with things. Here, we consider an agent to be anything that performs some action or role. This is important, as it decouples the role of an agent from the agent itself. For example, a Person may be Husband and Widower at some point, but should not be both simultaneously. These patterns enable the capture of this data. In fact, the agent role and participante role patterns are convenient specializations of property reification that have evolved into a modelling practice writ large. In this category, we also include the name stub, which is a convenient instantiation of the stub metapattern; it allows us to acknowledge that a name is a complicated thing, but sometimes we only really need the string representation.

**Description and Details** This category contains patterns that model the description of things. These patterns are relatively straightforward, models for capturing "how much?" and "what kind?" for a particular thing; patterns that are derived from Winston's part-whole taxonomy [23]; a pattern extracted from PROV-O [18], perhaps to be used to answer "where did this data come from?"; and a pattern for associating an identifier with something.

### 3.2 Using MODL

There are two different ways to use MODL—for use in ontology modelling and for use in tools. In both cases, MODL is distributed as a ZIP archive of the patterns' OWL files and accompanying documentation. In the case of the Ontology Engineer, it is simply used as a resource while building an ontology, perhaps by using Modular Ontology Modelling or eXtreme Design methodologies. For the tool developer, we also supply an ontology consisting of exactly the OPLa annotations from each pattern that pertain to OntologicalCollection. As OPLa is fully specified in OWL, these annotations make up an ontology of patterns and their relations. One particular use-case that we foresee is a tool developer querying the ontology for which patterns are related to the current pattern, or looking for a pattern based on keywords or similarity to competency questions.

### 3.3 Excerpt from Pattern Documentation

**Summary**

Figure 1 depicts the schema diagram for the Provenance pattern, as included in MODL. The EntityWithProvenance Pattern is extracted from the PROV-O

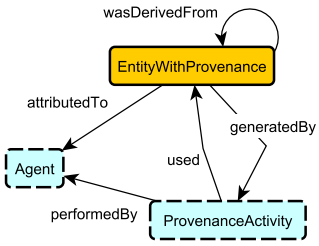



Fig. 1: This figure depicts the schema diagram for the EntityWithProvenance Pattern which is essentially the core of the Provenance Ontology (PROV-O). Yellow boxes are concepts. Light blue boxes with a dashed border are external to both the pattern and MODL that the developer may want to also make into a module.

ontology. At the pattern level, we do not want to make the ontological committment to a full-blown ontology. It suffices to align a sub-pattern to the core of PROV-O. [18]

The EntityWithProvenance class is any item of interest to which a developer would like to attach provenance information. That is they are interested in capturing, who or what created that item, what was used to derive it, and what method was used to do so. The "who or what" is captured by using the Agent class. The property, wasDerivedFrom is eponymous—it denotes that some set of resources was used during the ProvenanceActivity to generate the EntityWithProvenance.

**Axiomatization**[10]

$$\exists \mathsf{attributedTo.Agent} \sqsubseteq \mathsf{EntityWithProvenance} \quad (1)$$
$$\mathsf{EntityWithProvenance} \sqsubseteq \forall \mathsf{attributedTo.Agent} \quad (2)$$
$$\exists \mathsf{generatedBy.ProvenanceActivity} \sqsubseteq \mathsf{EntityWithProvenance} \quad (3)$$
$$\mathsf{EntityWithProvenance} \sqsubseteq \forall \mathsf{generatedBy.ProvenanceActivity} \quad (4)$$
$$\exists \mathsf{used.EntityWithProvenance} \sqsubseteq \mathsf{ProvenanceActivity} \quad (5)$$
$$\mathsf{ProvenanceActivity} \sqsubseteq \forall \mathsf{used.EntityWithProvenance} \quad (6)$$
$$\exists \mathsf{performedBy.Agent} \sqsubseteq \mathsf{ProvenanceActivity} \quad (7)$$
$$\mathsf{ProvenanceActivity} \sqsubseteq \forall \mathsf{performedBy.Agent} \quad (8)$$

**Axiom Explanations**

[10] Axiomatization is extensive while avoiding undesirably strong ontological commitments. Most axioms for the MODL patterns follow the template of the OWLAx Protégé plug-in [19].

1. Scoped Domain:The scoped domain of attributedTo, scoped by Agent, is EntityWithProvenance.
2. Scoped Range: The scoped range of attributedTo, scoped by EntityWithProvenance, is Agent.
3. Scoped Domain:The scoped domain of generatedBy, scoped by ProvenanceActivity, is EntityWithProvenance.
4. Scoped Range: The scoped range of generatedBy, scoped by EntityWithProvenance, is ProvenanceActivity.
5. Scoped Domain:The scoped domain of used, scoped by EntityWithProvenance, is ProvenanceActivity
6. Scoped Range: The scoped range of used, scoped by ProvenananceActivity, is EntityWithProvenance.
7. Scoped Domain:The scoped domain of performedBy, scoped by Agent, is ProvenanceActivity.
8. Scoped Range: The scoped range of performedBy, scoped by ProvenanceActivity, is Agent.

**Competency Questions**

CQ1. Who are the contributors to this Wikidata page?
CQ2. From which database is this entry taken?
CQ3. Which method was used to generate this chart and from which spreadsheet did the data originate?
CQ4. Who provided this research result?

### 3.4 Details

*Persistent URI* The persistent URI for this resource is `https://archive.org/services/purl/purl/modular_ontology_design_library`. The Version 1.0 snapshot and its documentation may be found there. Additionally, it provides helpful links to a technical report and the living data on GitHub, as discussed below. We emphasize that this should not be considered to be a migration of the ODP portal to GitHub, instead, simply where this resource lives, and as such is not meant to supercede or replace the ODP portal.

*Canonical Citation* The canonical citation for this resource may be found on arXiv [22]. The first version of the release has a DOI through Zenodo[11]

*Documentation* In addition to this document, we provide in-depth documentation on the library. This documenation contains a primer on ontology design patterns, as a concept, as well as common techniques used in their formalization. Most importantly, for each pattern it provides a schema diagram, its axiomatization, and explanations for each of those axioms. As mentioned in Section 3.1, each pattern is thoroughly annotated with OPLa which provides further documentation on its use and provenance.

[11] `10.5281/zenodo.3228128`

*Sustainability & Maintenance* MODL straddles the realms of dataset and software library; the resource is essentially a snapshot of data that lives. Due to this potential for change, we intend to maintain MODL analogously to a software project. Indeed, while the snapshots will be distributed as ZIP archives, the living data is (at the time of this writing) hosted on GitHub.[12] The Data Semantics Laboratory[13] will host MODL's snapshots and appropriate documentation indefinitely. The authors plan to drive further development of needed or requested patterns. Furthermore, by using Git[14] we inherit mechanisms for tracking issues and versions and incorporating such community contributions into future releases.

*License Information* This resource is released under the Creative Commons Attribution 4.0 International Public License the details of which can be found online.[15] A copy of license text is included in the repository.

## 4 Conclusions

MODL is a curated collection of well-documented ontology design patterns. We have created this resource to meet a community-recognized need for tooling infrastructure for ontology engineering. In particular, this resource makes ontology design patterns both findable and accessible, shows how they are interoperable, and promotes their reuse. Furthermore, we posit that future ontologies reusing these patterns will promote their interoperability and reuse.

### 4.1 Next Steps

The next steps are many, as MODL is a multifacted, foundational resource. We have identified several patterns that we deem necessary for covering additional frequently encountered modelling needs, e.g. a process pattern or patterns. In addition, there are many alternative patterns that could be considered for future releases. As mentioned in Section 3.1, we also want to further flesh out the documentation with respect to [13], as well as provide individual landing pages describing the ODPs. One future use case that we foresee for this resource is the mapping of competency questions to example SPARQL queries, which maybe could be used as a gold-standard training set for an automated translator. Also mentioned in Section 3.1, we intend to work closely with the digital humanities community for their knowledge representation needs. Finally, we have noted the extreme importance of working closely with tool developers; there is ongoing work to create a Protégé plug-in that utilizes MODL as a base for modular ontology modelling, as inspired by [6, 20]. Furthermore, we wish to explore automating the creation of a MODL-like resource. That is, provide a set of scripts

[12] `https://github.com/cogan-shimizu-wsu/modular-ontology-design-library`
[13] `http://daselab.org/`
[14] `https://git-scm.com/`
[15] `https://creativecommons.org/licenses/by/4.0/legalcode`

or instructions that allow developers to create their own local repository of their own frequently used patterns. Finally, we wish to layout a template for describing MODL-like resources using the Data Catalog Vocabulary[16] and Schema.org's Dataset.[17]

*Acknowledgement.* Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI). Quinn Hirt acknowledges funding from the Air Force Office of Scientific Research under award number FA9550-18-1-0386.

## 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. P. A. Bonatti, S. Decker, A. Polleres, and V. Presutti. Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371). *Dagstuhl Reports*, 8(9):29–111, 2019.
3. Z. Dragisic, P. Lambrix, and E. Blomqvist. Integrating ontology debugging and matching into the extreme design methodology. In E. Blomqvist, P. Hitzler, A. Krisnadhi, T. Narock, and M. Solanki, editors, *Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pensylvania, USA, October 11, 2015.*, volume 1461 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.
4. A. Gangemi and V. Presutti. Ontology design patterns. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 221–243. Springer, 2009.
5. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.
6. K. Hammar. Ontology design patterns in WebProtégé. 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.
7. 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, 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 189–198. IOS Press, 2016.
8. K. Hammar and V. Presutti. Template-based content ODP instantiation. In K. Hammar, P. Hitzler, A. Krisnadhi, A. Lawrynowicz, A. G. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns [revised and extended versions of the papers presented at the 7th edition of the Workshop on*

[16] `https://www.w3.org/TR/vocab-dcat/`
[17] `https://schema.org/Dataset`

*Ontology and Semantic Web Patterns, WOP@ISWC 2016, Kobe, Japan, 18th October 2016]*, volume 32 of *Studies on the Semantic Web*, pages 1–13. IOS Press, 2016.
9. 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*. IOS Press, 2016.
10. 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, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
11. P. Hitzler and A. Krisnadhi. A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. *CoRR*, abs/1808.08433, 2018.
12. N. Karima. *Automated Rendering of Schema Diagram for Ontologies*. PhD thesis, Wright State University, 2017.
13. N. Karima, K. Hammar, and P. Hitzler. How to document ontology design patterns. In K. Hammar, P. Hitzler, A. Lawrynowicz, A. Krisnadhi, A. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 15–28. IOS Press, Amsterdam, 2017.
14. A. Krisnadhi and P. Hitzler. Modeling with ontology design patterns: Chess games as a worked example. 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 3–21. IOS Press, 2016.
15. 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, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 301–309. Springer, 2015.
16. V. Presutti, E. Daga, A. Gangemi, and E. Blomqvist. eXtreme Design with content ontology design patterns. In E. Blomqvist, K. Sandkuhl, F. Scharffe, and V. Svátek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009) , collocated with the 8th International Semantic Web Conference ( ISWC-2009 ), Washington D.C., USA, 25 October, 2009.*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
17. A. L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In J. H. Gennari, B. W. Porter, and Y. Gil, editors, *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP 2003), October 23-25, 2003, Sanibel Island, FL, USA*, pages 121–128. ACM, 2003.
18. S. Sahoo, D. McGuinness, and T. Lebo. PROV-o: The PROV ontology. W3C recommendation, W3C, Apr. 2013. http://www.w3.org/TR/2013/REC-prov-o-20130430/.
19. M. K. Sarker, A. A. Krisnadhi, and P. Hitzler. OWLAx: A protégé 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.

20. C. Shimizu. Towards a comprehensive modular ontology IDE and tool suite. In S. Kirrane and L. Kagal, editors, *Proceedings of the Doctoral Consortium at ISWC 2018 co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018.*, volume 2181 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2018.
21. C. Shimizu, A. Eberhart, N. Karima, Q. Hirt, A. Krisnadi, and P. Hitzler. A method for automatically generating schema diagrams for modular ontologies. In *1st Iberoamerican Conference on Knowledge Graphs and the Semantic Web*, 2019. To Appear.
22. C. Shimizu, Q. Hirt, and P. Hitzler. Modl: A modular ontology design library. Technical report, Wright State University, Dayton, Ohio, April 2019.
23. C. Shimizu, P. Hitzler, and C. Paul. Ontology design patterns for winston's taxonomy of part-whole relations. In E. Demidova, A. Zaveri, and E. Simperl, editors, *Emerging Topics in Semantic Technologies – ISWC 2018 Satellite Events [best papers from 13 of the workshops co-located with the ISWC 2018 conference]*, volume 36 of *Studies on the Semantic Web*, pages 119–129. IOS Press, 2018.
24. M. D. Wilkinson, M. Dumontier, et al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018 EP –, Mar 2016. Comment.

# Modular Ontologies As A Bridge Between Human Conceptualization and Data

Pascal Hitzler and Cogan Shimizu

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

**Abstract.** Ontologies can be viewed as the middle layer between pure human conceptualization and machine readability. However, they have not lived up to their promises so far. Most ontologies are too tailored to specific data and use-cases. By making sometimes strong, or sometimes too weak, ontological commitments, many existing ontologies do not adequatly reflect human conceptualizations. As a result, sharing and reuse of ontologies is greatly inhibited. In order to more effectively preserve this notion of human conceptualization, an ontology should be designed with modularity and extensibility in mind. A modular ontology thus may act as a bridge between human conceptualization and data.

## 1 The Case for Modular Ontologies

The Internet is the single largest repository of knowledge to have ever existed and continues to grow every second. The amount of data continuously generated by both humans and machines defies comprehension: from second-by-second meteorological data gathered by sensors to academic articles written by scientists to communications on social media networks to collaborative articles on Wikipedia. How can we represent and link these disparate forms of data together in order to generate an understandable gestalt? We would require a way to organize acquired data such that some critical part of the human conceptualization of each piece is preserved.

Ontologies, as "explicit specifications of conceptualizations," seem like a natural fit for the role [2]. With the explosive growth of the Semantic Web in the last decade, it would seem that they have seen no small success for that purpose. Ontologies offer a human accessible organization of immense amounts of data and act as a vehicle for the sharing and reuse of knowledge.

Unfortunately, published ontologies have often not lived up to these promises. Large, monolithic ontologies, designed with very strong – or very weak – ontological commitments are very difficult to reuse across the same domain, let alone different domains. Strong ontological commitments lead to overspecification, to ontologies essentially being only fit for the singular purpose for which they were originally designed. Weak ones lead to ambiguity of the model, sometimes to the extent that is hard to grasp what is actually being modeled.

We posit that one effective way to obtain ontologies which are easier to reuse, is to build them in a modular fashion. A sufficiently modularized ontology [9]

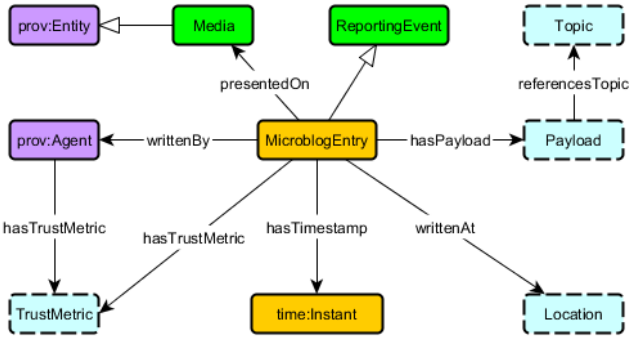



Fig. 1: This is a graphical view of the MicroblogEntry ODP [12]. Yellow boxes indicate datatypes, light blue boxes with dashed borders indicate external patterns. Purple is used for external classes belonging to PROV-O [1]. Green is used for external classes belonging to [8]. White arrowheads represent the owl:SubclassOf relation.

is designed such that individual users can easily adapt an ontology to their use cases, while maintaining integration and relationships with other versions of the ontology. A modular ontology is constructed by piecing together so-called ontology modules. Ontology modules are created by adapting Ontology Design Patterns to the domain and use-case [4, 5].

## 2 Ontology Design Patterns

In general, patterns are invariances that may be observed over different media (e.g. data or processes). Ontology Design Patterns (ODP) are the recognition of *conceptual* patterns that occur across different domains. Modeling with ODPs has established itself as an ontology engineering paradigm [5].

The Semantic Trajectory ODP [7] is a classic example of a recurring pattern. However, patterns are designed to be sufficiently general as to apply to many different cases as possible. Thus, it is necessary to create a module from them by adapting the pattern to the specific domain and use-case in mind. The Semantic Trajectory ODP has been successfully modularized a number of times; two prominent examples are the CruiseTrajectory ODP [11] and the SpatiotemporalExtent ODP [10]. For a thorough tutorial on creating modules out of patterns, see [9].

As another example, Figure 1 shows a graphical representation of the MicroblogEntry ODP [12]. This ODP clearly demonstrates pattern reuse and how

adequate ontological commitment eases of modularization. This ODP was engineered to leverage as much existing work as possible. For example, the Media and ReportingEvent concepts are defined in [8]. The concepts Entity and Agent come from the popular PROV Ontology that express provenence data [1]. Further, this ODP avoids overly strong ontological commitments, allowing it to be easily modularized to represent specific microblogs (e.g. Twitter vs. Facebook vs. Instagram).

## 3 The Future of Modular Ontology Engineering

The promise of modular ontologies is still being realized. There are yet open questions concerning ontology design patterns, their usage, and the surrounding supporting tools and infrastructure. For a more thorough examination of these questions and challenges, see [3]. That is not to say that there are no efforts underway; here, we briefly identify some of these ongoing efforts.

Perhaps the most fundamental purpose of the Semantic Web is to enable the sharing and reuse of knowledge. Certainly, a pattern is knowledge in and of itself. Thus, it is only reasonable that there needs to be a way to enable the sharing and reuse of patterns, as well. To do so, we are working towards the development of a "smart," central repository. Such a repository would be initially populated with a critical mass of fundamental ODPs. That is, a collection of ODPs with sufficient breadth and generalization such that their combination covers any complex conceptualization.

In addition, these patterns will be annotated in a systematic and rigorous way. Answering questions such as

- How do patterns interact with each other?
- Do they import other patterns?
- Which pattern did this module reuse as a template?

Recently, [6] introduced the Ontology Design Pattern Representation Language (OPLa) as a way to address those questions, and others. The smart repository would use these OPLa annotations in order to inform an ontology engineer on available patterns, especially those related to their domain and use-cases.

Between the central repository and OPLa, the next step will be to create a graphical interface for the assembly and modularization of ontologies. This will be a combination of different visualization strategies and a plug-and-play system for ODPs.

And finally, as we learn how to most help humans create ontologies, can we attempt to also automate these processes? That is, automatically create an ontology from a dataset and present it as a "first draft" to the ontology engineer for editing?

*Acknowledgement.* Cogan Shimizu acknowledges support by the Dayton Area Graduate Studies Institute (DAGSI).

## References


1. P. Groth and L. Moreau, editors. *PROV-Overview: An Overview of the PROV Family of Documents*. W3C Working Group Note 30 April 2013, 2013.
2. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.
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. K. Hammar, P. Hitzler, A. Krisnadhi, A. G. Nuzzolese, and M. Solanki, editors. *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*. IOS Press, 2017.
5. 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*. IOS Press, 2016.
6. 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.
7. Y. Hu, K. Janowicz, D. Carral, S. Scheider, W. Kuhn, G. Berg-Cross, P. Hitzler, M. Dean, and D. Kolas. A geo-ontology design pattern for semantic trajectories. In T. Tenbrink et al., editors, *Spatial Information Theory, COSIT 2013, Scarborough, UK, September 2-6, 2013. Proceedings*, volume 8116 of *Lecture Notes in Computer Science*, pages 438–456. Springer, 2013.
8. E. Kowalczuk and A. Lawrynowicz. The Reporting Event ODP and its extension to report news events. In K. Hammar et al., editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 105–117. IOS Press, 2017.
9. 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.
10. A. Krisnadhi, P. Hitzler, and K. Janowicz. A spatiotemporal extent pattern based on semantic trajectories. In K. Hammar et al., editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 47–54. IOS Press/AKA Verlag, 2017.
11. 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.
12. C. Shimizu and M. Cheatham. An ontology design pattern for microblog entries. 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.

# The Enslaved Ontology: Peoples of the Historic Slave Trade

Cogan Shimizu[a], Pascal Hitzler[a], Quinn Hirt[b], Dean Rehberger[c], Seila Gonzalez Estrecha[c], Catherine Foley[c], Alicia M. Sheill[c], Walter Hawthorne[c], Jeff Mixter[d], Ethan Watrall[c], Ryan Carty[c], Duncan Tarr[c]

[a]*Department of Computer Science, Kansas State University, USA*
[b]*Department of Computer Science and Engineering, Wright State University, USA*
[c]*Matrix, Michigan State University, USA*
[d]*OCLC Research, USA*

**Abstract**

We present the Enslaved Ontology (V1.0) which was developed for integrating data about the historic slave trade from diverse sources in a use case driven by historians. Ontology development followed modular ontology design principles as derived from ontology design pattern application best practices and the eXtreme Design Methodology. Ontology content focuses on data about historic persons and the event records from which this data can be taken. It also incorporates provenance modeling and some temporal and spatial aspects. The ontology is available as serialized in the Web Ontology Language OWL, and carries modularization annotations using the Ontology Pattern Language (OPLa). It is available under the Creative Commons CC BY 4.0 license.

*Keywords:* digital humanities, modular ontology, data integration, ontology design patterns, history of the slave trade

## 1. Introduction

The scourge of African enslavement was fundamental to the making of Europe, Africa, the Americas, and Middle East and parts of the Asian subcontinent. The enduring legacies of black bondage shape the moral questions of humanity in our times. We have seen in the past decade a growth in interest in the subject in film, on television, and in historical fiction. Historians have spilled much ink writing monographs aimed primarily at other scholars. At the same time, however, it is a worthy goal to expand the production of scholarly output and to bring what historians do to the general public. This aims to shed light on questions such as: How can we more effectively answer important moral questions? How can we make those questions part of a broader public discourse? What sources are available? How can we give broad access to them? And how in the decades to come will scholars answer questions about black bondage and its legacies when much valuable source material is deteriorating due to inattention, siloed scholarly activities, and underfunded archives?

During the past two decades there has been a significant shift in perceptions about what we can know about enslaved Africans, their descendants, and those who asserted ownership over them throughout the world. Those on the cutting edge of the digital humanities and social sciences have set about identifying, digitizing, analyzing, and making these resources available on innovative public history and cultural heritage websites. As a result, a growing number of collections of scanned original manuscript documents, digitized material culture, and databases, that organize and make sense of records of enslavement, are free and readily accessible for scholarly and public consumption.

Online databases about African slavery and the slave trade have a history that stretches back to the late 1990s and early 2000s, first provided on CD-ROMs, then on the World Wide Web as time progressed – and we provide key references in the related work section. Over time, a plethora of projects and teams generated a wide variety of databases with different foci and specializations. So, although this data is available through these data silos, this proliferation of different projects and databases presents scholars, students, and the interested public with a number of challenges:

- Most of these databases focus on the individuals of the slave trade, but data is often limited to the focus of the project. Further, the task of disambiguating (or merging) individuals across multiple datasets is nearly impossible given the current, siloed nature of all databases about slavery and the enslaved;
- There is no central, universally recognized clearinghouse for slave data. As such, it is difficult to find projects and databases;
- Individual projects and databases are isolated, preventing federated and cross project searching, browsing, and quan-

*Email addresses:* cogan.shimizu@coganshimizu.com (Cogan Shimizu), pascal@pascal-hitzler.de (Pascal Hitzler), hirtquinn@gmail.com (Quinn Hirt), rehberge@msu.edu (Dean Rehberger), seilage@msu.edu (Seila Gonzalez Estrecha), foleyc@msu.edu (Catherine Foley), asheill@msu.edu (Alicia M. Sheill), walterh@msu.edu (Walter Hawthorne), mixterj@oclc.org (Jeff Mixter), watrall@msu.edu (Ethan Watrall), cartyrya@msu.edu (Ryan Carty), tarrdunc@msu.edu (Duncan Tarr)

titative analysis;

- There are no best practices for digital data creation collectively agreed upon by the scholarly community;
- Important data is often lost or remain locked away in scholars' files, completely inaccessible to other scholars, students, descent communities, and the general public;
- Project participants rarely get scholarly credit for the work that goes into creating and releasing digital data;
- Humanists have little incentive to deposit datasets.

To address these challenges, the Enslaved project, funded by The Andrew W. Mellon Foundation and led by Michigan State University, is currently underway, and it is set to pioneer a new model for humanities scholarship. Enslaved brings together programmers, project managers, archivists, librarians, and historians in a collective endeavor and, over the years, with an expanding consortium of contributors. This collaborative approach, which is made possible by the World Wide Web, is set to challenge humanists to broaden their thinking about the production of knowledge; the sharing, as opposed to guarding, of research materials; and the benefits of collaboration. In sum, the model of Enslaved promises to disrupt conventions of humanities scholarship in much the way it would disrupt – for the better – historical perspectives on slavery and the individual lives of those enslaved.

The technical goal of Enslaved is thus to establish what we call the Enslaved Hub,[1] a website that provides one-stop querying and inspection capabilities for integrated historic data on the slave trade, originating from a diverse set of data sources and contributors, thereby allowing students, researchers and the general public to search over numerous databases to understand and reconstruct the lives of individuals who were part of the historical slave trade. To address the underlying data integration issues, Enslaved opted to follow the state of the art by establishing a knowledge graph, expressed in RDF, with an underlying schema in form of an OWL ontology. We call this the Enslaved Ontology; our modeling approach and its core concepts comprise the core of this paper.

The Enslaved Ontology expresses metadata record types and core fields that the Enslaved research team identified as frequently occuring in historic slave trade data projects. After months of evaluating datasets from the domain of slavery studies, Enslaved standardized metadata fields and developed controlled vocabularies for four types of records regularly found in these data collections: EVENT, PERSON, PLACE, and SOURCE. There are 43 Enslaved fields: 9 EVENT fields; 19 PERSON fields; 9 PLACE fields; and 6 SOURCE fields. Enslaved defined controlled vocabulary terms that standardize data in nine Enslaved fields. Documentation for Enslaved Metadata and Controlled Vocabularies is available at https://docs.enslaved.org/

[1] Eventually to be located at `http://enslaved.org/`

Like the metadata from which it originates, the Enslaved Ontology is the shared language that allows the Enslaved Hub to search over numerous disperate datasets to understand and reconstruct the lives of individuals who were part of the historical slave trade. Without a data model based on the records, fields, and terms included in the metadata and controlled vocabularies, it would be impossible for machine processing to make sense of all of the data available to the system.

The rest of this paper is structured as follows. In Section 2 we briefly describe our modeling approach. In Section 3 we provide a description of the ontology's key modules and their relationships. In Section 4 we briefly discuss the intended usage of the ontology. In Section 5 we discuss related work and in Section 6 we conclude.

The ontology is available at `https://docs.enslaved.org/`. Its detailed technical documentation is available from the same site and from [27].

## 2. Ontology Modeling Approach

We follow a modular ontology modeling approach based on ontology design patterns [3, 5, 6], as it will produce an ontology with desirable traits. Such a methodology is designed to ensure high quality and reuseability of the ontology, as well as cater to future expansions, both in terms of scope and in terms of granularity. These will allow the Enslaved Ontology to evolve as needs evolve and the number of collaborators increase. The modular ontology modeling approach and its rationale has been described in [16], and it is closely related to the eXtreme Design approach [2].

Following this methodology as laid out in [16] and further detailed in [9], we took the following subsequent steps.

*Step 1: Define use case or scope of use cases.*

The use case, including its anticipated future trajectory, was laid out in the Introduction, Section 1.

*Step 2: Collect competency questions while looking at possible data sources and scoping the problem, i.e., decide on what should be modeled now, and what should be left for a possible later extension.*

Competency questions were assembled from various sources. In January 2018, the Enslaved project team solicited search questions from the eight partner projects, seeking expert input from historians actively engaged in slavery studies and databasing. Over a four-week period, partners shared thoughts about potential audiences for the Enslaved project and how different audiences would have different search expectations and needs. Specific search suggestions focused on categories such as names, events, relationships, and place and time constraints. The Matrix team summarized this input.[2] Then the Enslaved

[2] The summary is available from `https://docs.enslaved.org/competencyquestions/v1/enslavedcompetencyquestions-v1.pdf`.

- Public 5: List the enslaved people in Reed County, NC, in the second half of the eighteenth century.
- Public 12: Who were the godparents of my great-great grandmother, Beatriz of the Ambaca nation, baptized at São José church in Rio de Janeiro on April 12, 1840.
- K12 1: Who did Thomas Jefferson enslave at Monticello?
- K12 9: How many enslaved children lived in Boston when Phillis Wheatley lived there?
- Pro 4: What were the gender ratios of enslaved people identified as being of XXXX ethnicity?
- Pro 6: In what records does the enslaved person named XXXX appear? What were XXXX's professions? What places did he live? Who were his/her children and childrens children? Who did he marry?
- Pro 9: I am researching an enslaved person named Mohammed who was a new arrival from West Africa in Charleston in 1776. Is there data about what slave ship he might have been on?
- Pro 20: What ever happened to Bernarda Angola, a Free African who ran away from her mistress Maria dos Santos Pereira, in June 1845?

Figure 1: Sample competency questions.

team invited four project partners to draft competency questions, queries about the data but formulated in natural language. These scholars created 17 questions from a Public user perspective, 14 questions from students and teachers in primary through secondary school (K12), and 24 questions from a professional scholarly user.[3] Example competency questions are listed in Figure 1.

The collected competency questions span the scope of an expanded Enslaved Hub, which goes significantly beyond the first stage of modeling and demonstration which we – and the ontology in the current version – needed to focus on. The historians in the team thus designed a suitable scope for a first stage of the effort, based on the competency questions and the availability of data sources. From the plethora of potential data sources, some of which are presented in the related work section, eight were selected, namely African Origins,[4] Voyages: The Trans-Atlantic Slave Trade Database,[5] Slave Societies Digital Archive,[6] Dictionary of Caribbean and Afro-Latin American Biography, Dictionary of African Biography and African American National Biography,[7] Freedom Narratives,[8] Legacies of British Slave-ownership,[9] The Liberated Africans Project[10] and Slave Biographies.[11] These sources were selected as they provided a range of different kinds of data and seemed representative as a starting point.

Enslaved used the eight original datasets as the foundation for the Ontology. Many of the underlying data points are the same across the data collections, for example Names of individuals, Gender, Race/Color, and freedom status (enslaved, owner, freed person). However, historians tend to be idiosyncratic in their data collection practices. Some copy verbatim from sources while others may have complex coding practices. Field names and values can vary in spelling, language and use antiquated and contemporary terms. Categorization also tends to be non-standardized. Therefore there are some challenges mapping disparate datasets to the data model. The original datasets were selected for the project because they included a wide range of challenges that needed to be addressed. For example, a dataset focused on biographies combines agent and event data in the single row while another dataset captured all agent data in one table and connected it to event data in another table using an Enslaved property. Despite these divergences, Enslaved has found that robust metadata documentation for legacy datasets and Enslaved has allowed coherent mapping the data model. To date Enslaved has mapped all eight original datasets to the Enslaved Ontology. From the outset, Enslaved intended to include only a subset of the data points in these datasets, those that commonly recur across historical slave trade data collections.

*Step 3: Identify key notions from the data and the use case and identify which pattern should be used for each. Construct a set of modules from these.*

The list of key notions was quickly finalized during the first in-person modeling meeting in summer 2018, where about a dozen researchers, including historians, data experts, and ontology engineers, met to draft the modules and the overall ontology. As an ontology for historic data, time, place, and provenance play a necessary role. The content focus of the ontology is on persons and key biographical or person data, more precisely name, age, sex, occupation, status (e.g. enslaved or freed), race, ethnolinguistic and/or geographical origin, participation in events, and relationships to other persons or organizations such as family relations or ownership relations. Events play a particular role in the data, as relevant historic records usually originate from specific events such as estate inventories. As further modules it seemed necessary to have a generic way of providing descriptions and external references, as well as a way to refer to specific research projects contributing data.

As for identifying ontology design patterns as a basis for corresponding modules, some were obvious choices, such as using the core of PROV-O [21] for provenance; using agent instead of person to include organizations or groups when needed; an agent roles pattern [15] for participations in events and for inter-agent relationships. Some person data, such as sex, occupation, enslaved/freed status, race, and origin seemed to be best captured by using controlled vocabularies, as were age categories, though numeric age also seemed desirable. It also seemed opportune to defer a complex modeling of names from different ethnic and linguistic origins and instead to go for a simple name

[3] The full list of competency questions is available from https://docs.enslaved.org/competencyquestions/v2/enslavedcompetencyquestions-v2.pdf.
[4] http://www.african-origins.org/
[5] http://www.slavevoyages.org/
[6] http://www.vanderbilt.edu/esss/
[7] https://hutchinscenter.fas.harvard.edu/AANB
[8] http://freedomnarratives.org/
[9] http://www.ucl.ac.uk/lbs/
[10] http://liberatedafricans.org
[11] http://slavebiographies.org/

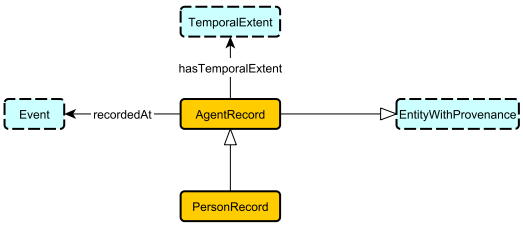



Figure 2: Schema Diagram for the AgentRecord module. Orange boxes indicate classes. Dashed blue boxes indicate classes which are part of another module. White-headed arrows indicate subclass relationships.

stub [18]. The usage of time was also rather restricted, so for the first stage a decision was made to develop a relatively simple placeholder module which could be refined later. A comprehensive treatment of historic places was clearly out of scope for the project, and since others pursue historic gazetteers in a principled way[12] we took a limited approach compatible with these efforts.

Since the focus is on historic data, and on historic persons, as reported by various (and possibly conflicting) sources, it was also obvious that the modeling, and the resulting knowledge graph, would contain possibly conflicting data as reported, rather than anything resembling a base truth. This is of course obvious from a historian's perspective, but is sometimes less prominently modeled in current approaches to knowledge graph schema design. In the Enslaved Ontology, we make this explicit by primarily speaking about the *records* of agents (so-called AgentRecords) as they pertain to different Agents.

*Step 4: Put the modules together and add axioms which involve several modules.*

We defer this discussion to the description of the modules and the ontology, in Section 3.

*Step 5: Create OWL files.*

OWL files were created using Protégé [23], and the OPLa Annotator, presented in [26], for adding additional metadata using the Ontology Design Pattern Representation Language (OPLa) [7].

## 3. Description of the Enslaved Ontology

As discussed previously, the focus of the Enslaved Ontology is on the records of historic agents. In this paper, we do not intend to replicate the comprehensive documentation, which can be found at [27]. Thus, we provide a broad overview, discuss key modeling choices, and give a few more detailed examples.

The key notion of our model is that of an agent record. Figure 2 shows the corresponding schema diagram. Note the use of temporal and provenance information, which are described in separate modules, and of the Event module: In this historic context, agent records were usually recorded at some historic events. We will discuss provenance in more detail below.

The schema diagram, of course, is just a simplified visualization of the module. In terms of formal model, it consists of the OWL axioms listed in Figure 3.[13] Here and elsewhere in the ontology, the primary purpose of the formal axiomatization is to disambiguate the model, i.e., we were striving for as complete an axiomatization as possible, while avoiding ontological overcommitments. Each axiom was discussed in detail between the ontology engineers and the historians on the team. The axiomatization is expresssed using the OWL 2 DL profile. Note that while it is not currently our primary goal to do formal reasoning over the ontology [8], we do not want to rule out such goals in the future (e.g. the use of reasoning for consistency checking). Furthermore, in order to fully encode the knowledge as determined to be important by stakeholders and domain experts, we make use of a number of features only expressible in OWL 2 DL, e.g. right-hand disjunctions appearing in Axiom 7 in the formalization of AgentRecord or Axiom 2 in the formalization of ExternalReference. These axioms, and others, may be found in more detail in the full documentation [27].

$$\text{AgentRecord} \sqsubseteq \text{EntityWithProvenance} \quad (1)$$
$$\text{PersonRecord} \sqsubseteq \text{AgentRecord} \quad (2)$$
$$\text{AgentRecord} \sqsubseteq =1\text{hasAgentRecord}^{-}.\text{Agent} \quad (3)$$
$$\text{PersonRecord} \sqsubseteq =1\text{hasPersonRecord}^{-}.\text{Person} \quad (4)$$
$$\text{AgentRecord} \sqsubseteq =1\text{hasTemporalExtent.TemporalExtent} \quad (5)$$
$$\text{AgentRecord} \sqsubseteq \leq 1\text{recordedAt.Event} \quad (6)$$
$$\text{AgentRecord} \sqsubseteq \leq 1\text{isDirectlyBasedOn.EntityWithProvenance} \quad (7)$$
$$\text{hasPersonRecord} \sqsubseteq \text{hasAgentRecord} \quad (8)$$
$$\text{Person} \sqsubseteq \geq 0\text{hasPersonRecord.PersonRecord} \quad (9)$$

Figure 3: OWL axioms for the AgentRecord module.

Systematically speaking, most of the axiomatization follows the template laid out for the OWLAx Protégé plug-in [25, 9], i.e., axioms were selected from those available in OWLAx, which – consistent with our previous modeling experiences – suffices for most modeling requirements. Indeed, all of the axioms in Figure 3, except for the last two, follow the OWLAx template. (1) and (2) are subclass relationships, (3) and (4) are combined existentials and inverse functionalities, (5) is a combined existential and functionality, (6) and (7) are functionalities, (8) is a subproperty relationship. (9) is an axiom which is actually a tautology, but is included for the benefit of humans trying to understand the ontology: it indicates that a person may have person records.

[12] `http://whgazetteer.org/`

[13] A primer on description logic and the notation can be found in [1].

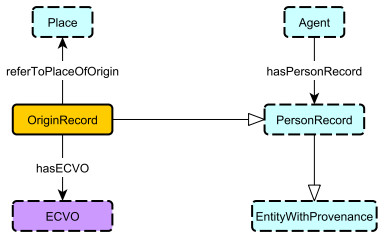



Figure 4: Schema Diagram for the OriginRecord module, color and shape usage is the same as in the previous diagram. Purple dashed boxes indicate a controlled vocabulary, i.e., all URIs of type ECVO (Ethnolinguistic Controlled Vocabulary of Origin) should be considered part of a controlled vocabulary.

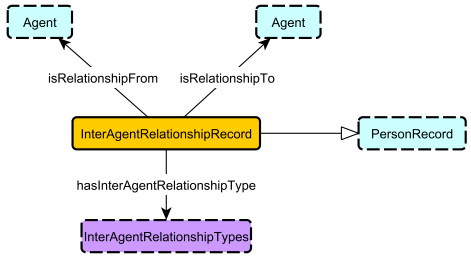



Figure 5: Schema Diagram for the InterAgentRelationshipRecord module, color and shape usage is the same as in the previous diagrams.

Several agent and person record modules are part of the Enslaved ontology; they are used for place of origin and ethnolinguistic origin of a person, for race, age, sex, occupation, name, and freedom status. They are also used for recording relationships between agents, and participations in events. We will now discuss a few of them.

The OriginRecord module is rather similar in structure to most of the other AgentRecord submodules, and so we use it to serve as a typical example. Its schema diagram can be found in Figure 4. We see that OriginRecord is a subclass of PersonRecord. A record of origin can be recorded in three ways, either by indicating a place (refering to the Place module), or by reference to a controlled vocabulary, the ECVO, which is being designed by Enslaved historians, or finally by referring to one ECVO and a place of origin. The OWL axiomatization which constitutes the module disambiguates the usage of this module: It prescribes that each OriginRecord can have at most one ECVO, and that each OriginRecord refers to at least one Place and/or at least one ECVO. Note that several places are allowed (as two places together again constitute a place, conceptually this may be considered the *union* of those places), and that there is no required relationship between place and ECVO, in case both are listed. The reasons for the latter are that, on the one hand, such relationships are rather controversial among historians, and that, on the other hand, such spatial information could be made part of the controlled vocabulary if desired. We anticipate that a refinement of this module (and of most others) may have to be done as usage of the ontology expands.

Records for inter-agent relationships, such as family or ownership relations, follow a standard relationship reification (or $n$-ary relation) pattern [11], with the addition of a controlled vocabulary for relationship types. The corresponding schema diagram can be found in Figure 5. The axiomatization, which we do not replicate here, follows the OWLAx templates. Note, though, that due to the reification it is not possible, in OWL DL, to specify that no agent can be in an InterAgentRelationship with itself, as non-simple OWL DL properties must not be irreflexive [22, 10].

Let us briefly pause to discuss the use of controlled vocabularies in our context. They are used in the Enslaved Ontology whenever a crisp categorization made sense from the domain experts' point of view, e.g., to enable efficient end-user querying of the data. The competency questions showed that queries for specific family relations or ethnolinguistic origins should occur frequently. In some cases, such as that for inter agent relationships, the controlled vocabulary entities act as types (in this case, for individuals belonging to InterAgentRelationshipRecord), and it could be argued that they should be modeled as subclasses of the InterAgentRelationshipRecord class, rather than as individuals. There are several reasons why we did not do that. (1) The controlled vocabulary is likely to change more rapidly than the rest of the ontology, and adding (or even removing) individuals appears to be less invasive rearding the ontology. That is, adding an individual of a class adds relatively little additional complexity as opposed to adding a new class, which could necessitate additional axiomatization. Indeed, we can now consider the controlled vocabularies to be seperate entities, with separate versioning, which can be updated without releasing a new version of the ontology. (2) In some cases, such as for the OriginRecord, it seems conceptually questionable to use the controlled vocabulary for classes. Thus, with all controlled vocabularies being individuals, the ontology has more coherence. (3) If needed, e.g., to establish formal relationships, expressed by OWL axioms, between controlled vocabulary items, then it is possible, in OWL DL, to map between these individual and corresponding class identifiers – this is a form of typecasting, laid out in detail in [20]. We acknowledge, though, that it could have been done differently; since typecasting is possible, though, the choice does not seem to make a huge difference.

Records for event participation, such as a baptism or a slave rebellion, follow a standard agent role pattern [15], which is a reification pattern. We again make use of controlled vocabularies for role types. The corresponding schema diagram can be found in Figure 6. Axiomatization is relatively straightforward following the OWLAx templates.

Event is a key module in the Enslaved ontology, as the creation of historical records usually happens at certain events, such as baptisms. While there are existing ontologies to capture event, such as the Simple Event Model [31], they did not fit our pur-

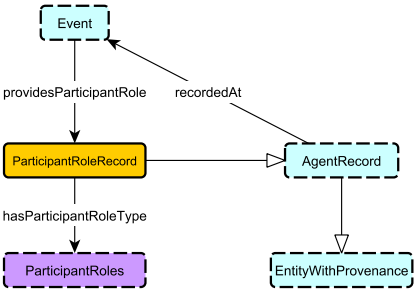



Figure 6: Schema Diagram for the ParticipantRoleRecord module, color and shape usage is the same as in the previous diagrams.

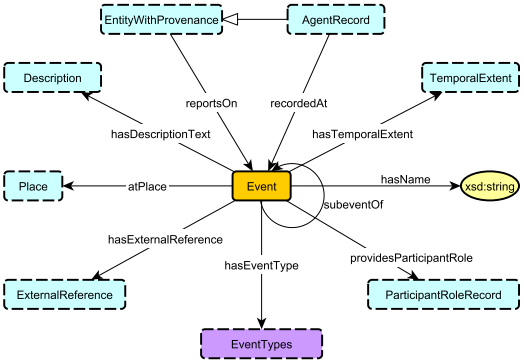



Figure 7: Schema Diagram for the Event module, color and shape usage is the same as in the previous diagrams. Yellow oval nodes indicate data types.

poses, mostly because they were much more detailed than required at this time. We thus derived our event module from a very simple event pattern [17] with appropriate modifications. E.g., at this stage there was no need for full-fledged spatiotemporal extents, and we added a controlled vocabulary to record event types, always keeping in mind that the module may have to be replaced by a module with finer granularity in the future. A schema diagram can be found in Figure 7.

For the axiomatization, which mainly follows the OWLAx templates, we required to express that, if an agent record is recorded at an event, then that agent record also reports on this event. This can be expressed as a (first order predicate logic) rule as

$$\text{AgentRecord}(x) \wedge \text{recordedAt}(x, y) \wedge \text{Event}(y) \rightarrow \text{reportsOn}(x,y).$$

Using a technique known as *rolification* [19, 24], this rule can

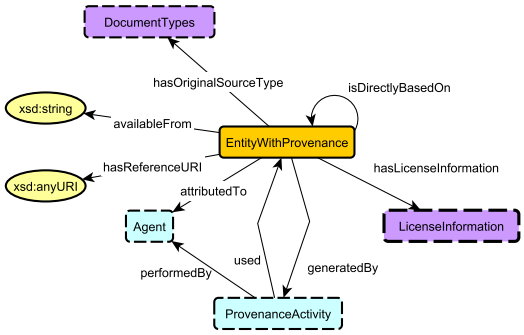



Figure 8: Schema Diagram for the Provenance module, color and shape usage is the same as in the previous diagrams.

be converted to OWL DL as the following three axioms.

$$\begin{aligned} \text{AgentRecord} &\sqsubseteq \exists \text{agentRecordSelfProperty.Self} \\ \text{Event} &\sqsubseteq \exists \text{eventSelfProperty.Self} \\ \text{agentRecordSelfProperty} \circ \text{recordedAt} &\circ \\ \text{eventSelfProperty} &\sqsubseteq \text{reportsOn} \end{aligned}$$

For provenance information, we borrowed the core pattern from PROV-O [21, 28] as a template, and added controlled vocabularies for license information and document types of the original source, which is important for Enslaved use case scenarios. We also added the option to record from where an entity is available (e.g., which database) together with a reference URI pointing to the exact database entry. This module's schema diagram can be found in Figure 8. The axiomatization follows mainly OWLAx templates. We record, in addition, that the original source type of an entity with provenance is the same as that of the entity with provenance it is directly based on, and vice-versa, which can be expressed as property chain axioms, as follows.

$$\begin{aligned} \text{isDirectlyBasedOn} \circ \text{hasOriginalSourceType} &\\ \sqsubseteq \text{hasOriginalSourceType} &\\ \text{isDirectlyBasedOn}^{-} \circ \text{hasOriginalSourceType} &\\ \sqsubseteq \text{hasOriginalSourceType} & \end{aligned}$$

The ontology has additional modules, in particular preliminary ones for place and time, which we do not discuss in more detail. However, we provide an overview schema diagram for the whole ontology in Figure 9. Further details can of course be found in the referenced technical report.

Modular ontologies are one way of addressing the maintenance, interoperability, extensibility, and reusability of ontologies. To do so, each module is annotated using the Ontology Design Pattern Representation Language (OPLa) [7] using the OPLa Annotator plugin [26] for Protégé [23]. These annotations allow us to fully describe the structure of a modular ontology

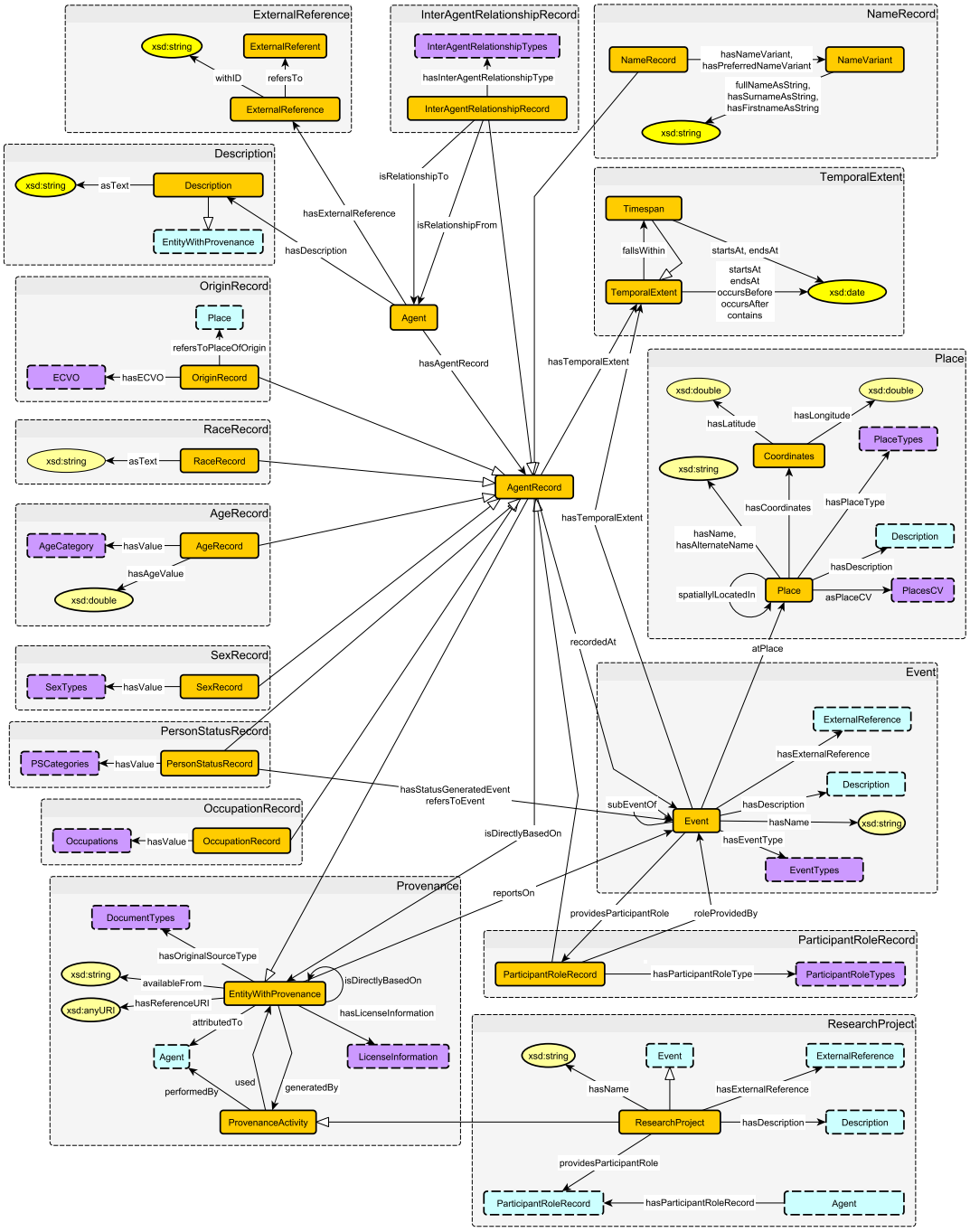



Figure 9: Overview schema Diagram for the Enslaved ontology, color and shape usage is the same as in the previous diagrams. The gray frames indicate (some of the) modules.

in a machine-readable way. Briefly, OPLa shows how the modules that comprise a modular ontology are interrelated and from which patterns those modules were derived. This, in turn, makes it easier to replace or modify modules as the ontology evolves.

The Enslaved Ontology is available from the Enslaved Hub[14] and is available under the Creative Commons Attribution License CC BY 4.0,[15] which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

## 4. Intended Usage

The Enslaved project aims to connect the different communities pertaining to the historic slave trade. The nature of this endeavor calls for a bootstrapping approach, where a demonstrator hub would first be limited in scope, diversity, and number of datasets integrated, however with the expectation that it would grow significantly once under way, as additional collaborators and contributors start sharing their data for the integration. The Enslaved Ontology, due to its modular nature, has the capacity to expand and evolve alongside the Enslaved Hub.

Additionally, we wish to enable community driven population of the knowledge graph. Currently, the Enslaved Ontology serves as the underlying schema for the Enslaved knowledge graph. The axiomatizations provided in earlier sections and in [27] are a way for disambiguating the model—at this time, there is no intent to do formal, deductive reasoning over the ontology. Instead, the ontology is meant for human consumption, and for informing the knowledge graph structure.

The Enslaved Ontology is also used to inform the population interface; it has been completely mapped into a Wikibase installation, the same technology that which underlies Wikidata [32]. This enables compatibility with the greater data and software ecosystem, for example the Wikimedia Foundation's[16] Wikipedia,[17] while maintaining a different path for the curation of the data. Blazegraph[18] is used as the underlying triplestore. Classes, properties, and individuals (instances) may be queried through a SPARQL query endpoint.

## 5. Related Work

### *5.1. Databases About the Slave Trade*

Online databases about African slavery and the slave trade have a history that stretches back to the late 1990s and early 2000s. Then three projects were trailblazing. First, in 1993 Edward L. Ayres and William Thomas launched Valley of the Shadows at the Virginia Center for Digital History.[19] The site is a digital archive of primary sources centered on the lives of people, both black and white, in Virginia and Pennsylvania during the era of the American Civil War. It allows readers to take multiple pathways through documentation from the period and, in so doing, to build a variety of narratives that tell history in a way that a book cannot. Second, in March 2000, Gwendolyn Midlo Hall published a CD-ROM with the Louisiana State University Press. The CD had information about over 100,000 slaves who labored in colonial Louisiana. In 2001, Hall launched much of the same information on a website sponsored by University of North Carolina and I-Biblio, Afro-Louisiana History and Genealogy.[20] An update and revision of the dataset would later be incorporated into Slave Biographies.[21] Third, in 1999 David Eltis, Stephen D. Brehrendt, David Richardson, and Herbert S. Klein published The Trans-Atlantic Slave Trade: A Database on CD-ROM with Cambridge University Press. The CD had information about 27,233 Atlantic slave ship voyages from 1595 to 1866. In 2006, the team launched Voyages: The Trans-Atlantic Slave Trade Database.[22] The Voyages site has grown, currently holding information about almost 36,000 slave voyages. For both Afro-Louisiana and Voyages, the advantages of the World Wide Web over a CD-ROM were obvious: audiences grew, information was available at no cost to anyone with Internet access, and data could be updated.

Other projects followed and were housed on the Internet. Slave Biographies: The Atlantic Database Network, which is home to updated data from Hall's I-Biblio site and to other datasets, is a digital project hosted by Michigan State University and Matrix: The Center for Digital Humanities and Social Sciences. It provides open-source demographic information, including names, ethnicities, skills, occupations, and illnesses, about individual slaves in colonial Louisiana and Maranhão, Brazil. In addition, there is Cornell University's Freedom on the Move,[23] Emory University and Harvard University's W.E.B. Du Bois Institute's African Origins.[24] and Vanderbilt University's Slave Societies Digital Archive.[25] Still other projects are supported at universities and heritage institutions both within and outside the United States. Among them are the Liberated Africans Project,[26] Digital Archaeological Archive of Comparative Slavery,[27] Legacies of British Slave-ownership,[28] Marronnage in Saint-Domingue, Haiti,[29] Baptismal Records Database for Slave Societies,[30] and Studies in the History of the African Diaspora.[31]

[14] https://docs.enslaved.org/
[15] http://creativecommons.org/licenses/by/4.0/
[16] https://en.wikipedia.org/wiki/Wikimedia_Foundation
[17] https://en.wikipedia.org/wiki/Wikipedia:About
[18] https://blazegraph.com/
[19] http://valley.lib.virginia.edu/
[20] http://www.ibiblio.org/laslave/
[21] http://www.slavebiographies.org/
[22] http://www.slavevoyages.org/
[23] http://freedomonthemove.org/
[24] http://www.african-origins.org/
[25] http://www.vanderbilt.edu/esss/
[26] http://www.liberatedafricans.org/
[27] http://www.daacs.org/
[28] http://www.ucl.ac.uk/lbs/
[29] http://www.marronnage.info/en/index.html
[30] http://bardss.matrix.msu.edu/
[31] http://tubman.info.yorku.ca/publications/shadd/

### 5.2. Related Digital Humanities Ontologies

To the best of our knowledge, there is no other ontology that aims to model the lives, times, and movements of peoples in the historic slave trade. However, there is an active community of historians and digital humanitarians utilizing Semantic Web technologies. We discuss a few such efforts in the digital humanities to provide context for our work. These chosen works are intended to be representative of a fairly extensive body of literature. We selected them based on relevance, shared vision, and recency, as well as attempting to choose representative types of topics.

**dataLegend** [12] is a platform for linking historical data, first published in 2015. Essentially, it allows for a researcher to upload their data, link it to other's data, and view the results of their mapping. This project is similar in vision to the Enslaved Hub, but with broader scope. In particular, the Enslaved project focuses on those people involved in the historic slave trade, where as dataLegend allows for the mapping of arbitrary historical data. While this is not undesirable, it is outside the scope of our work to be similarly comprehensive. However, dataLegend does remain a source of inspiration as Enslaved grows.

**Europeana**[32] is a massive in scope, digital archive with over 58 million metadata records of various media, first launched in 2008. Europeana acts as a discovery portal; it redirects users to different institutions' datasets. This work is similar in vision to the Enslaved Hub, but has a much more relaxed semantics for interrelating datasets, using Schema.org annotations.

**Semantic National Biography of Finland**[33] [14, 13] (BiographySampo) is a collection of short biographies of historic peoples that have been extracted from large biographical collections published in 2017. BiographySampo uses a combination of different vocabularies and models to represent historic peoples and the events and roles that connected them. This has similar goals as the Enslaved Ontology (e.g. the tracking of historical peoples); we see multiple points for alignment and inspiration as the ontology evolves.

**Historical Ecology and Recipes from Newspapers** [29, 30] are two projects, first published in 2018, that attempt to extract ecological and recipe information from historical newspapers. This approach is somehow similar to ours, in the sense that it is about extracting records of items of interest from historical documents and tracking how they change over time.

Additionally, we identify two upper ontologies that have been used in the digital humanities.

The **Simple Event Model** [34] (SEM) is an upper ontology for describing events, in particular those relating to history, cultural heritage, geography and multimedia. It captures some spatial and temporal data and different ways of describing how people, places, and objects interact. The SEM is a rather finely granular, complex, and robust model, but exceeds our current modeling needs. As the Enslaved Ontology is a modular ontology, it would be possible to use SEM instead of our Event module. In the future, as the Enslaved Ontology evolves, we may consider replacing the current Event module with an adaptation of the SEM.

The **International Committee for Documentation's Conceptual Reference Model** (CIDOC-CRM) is an upper ontology for "describing the implicit and explicit concepts and relationships used in cultural heritage documentation."[35] In the same manner that SEM provides a more heavyweight and more finely granular model, so does CIDOC-CRM. As such, we developed the Enslaved Ontology independently, but due to its modular nature, as the ontology evolves and needs change, an adaptation of the relevant portions of CIDOC-CRM will be considered.

### 5.3. Considerations on External Ontologies and Modules

During the development of the Enslaved Ontology, we also considered a number of external ontologies for use. We discuss a representative selection of them here. Furthermore, we may include alignments from our developed modules to these external ontologies in subsequent versions of the Enslaved Ontology.

**OWL-Time** is an ontology of temporal concepts encoded in OWL 2 DL [4]. The Time ontology is as robust as it is complex. Our modeling needs were met by simply using some basic temporal relationships and the XML schema datatypes, so we felt it unnecessary to import OWL-Time.

**Geonames Ontology**[36] is a geographical database containing the names and locations of over ten million places. However, it is not sufficient for our purposes as we are dealing, primarily, with historical places.

## 6. Conclusions

In this paper, we have presented the Enslaved Ontology (V1.0) which has been developed for use in integrating a wide variety of heterogeneous data sources found in the historian research communities. The ontology was developed in a modular fashion, thus facilitating future maintenance and extensibility—critical and beneficial design aspects, as the expected growth of the Enslaved Hub must be matched by the Enslaved Ontology. The ontology incorporates modules for capturing the spatial, temporal, and provenance aspects of historic events and agents (i.e., organizations or persons). The Enslaved Ontology is serialized in OWL and is equipped with annotations describing the modular structure, further improving its reusability in the future. It may be found online at `https://docs.enslaved.org/`.

[32] `https://www.europeana.eu/portal/en`
[33] `https://seco.cs.aalto.fi/projects/biografiasampo/en/`
[34] `https://semanticweb.cs.vu.nl/2009/11/sem/`
[35] `http://www.cidoc-crm.org/`
[36] `https://www.geonames.org/`

*Acknowledgement.* This work was supported by The Andrew W. Mellon Foundation through the Enslaved project. Cogan Shimizu acknowledges partial support from the Dayton Area Graduate Studies Institute.

## References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] 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.

[3] E. Blomqvist and K. Sandkuhl. Patterns in ontology engineering: Classification of ontology patterns. In C. Chen, J. Filipe, I. Seruca, and J. Cordeiro, editors, *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*, pages 413–416, 2005.

[4] S. Cox and C. Little, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation, 11 December 2012. Available at http://www.w3.org/TR/owl2-syntax/.

[5] A. Gangemi. Ontology design patterns for semantic web content. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, volume 3729 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2005.

[6] 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*. IOS Press, 2016.

[7] 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, Ó. Corcho, M. Horridge, D. Carral, and R. Hoekstra, editors, *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.*, volume 2043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[8] P. Hitzler and A. Krisnadhi. On the roles of logical axiomatizations for ontologies. 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 73–80. IOS Press, 2016.

[9] P. Hitzler and A. Krisnadhi. A tutorial on modular ontology modeling with ontology design patterns: The Cooking Recipes Ontology. Technical report, Data Semantics (DaSe) Lab, Wright State University, Dayton, OH, USA, 2018. Available from `http://pascal-hitzler.de/topics/publications.html`.

[10] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2010.

[11] R. Hoekstra. *Ontology Representation – Design Patterns and Ontologies that Make Sense*, volume 197 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[12] R. Hoekstra, A. Meroño-Peñuela, A. Rijpma, R. Zijdeman, A. Ashkpour, K. Dentler, I. Zandhuis, and L. Rietveld. The dataLegend ecosystem for historical statistics. *J. Web Semant.*, 50:49–61, 2018.

[13] E. Hyvönen, P. Leskinen, M. Tamper, H. Rantala, E. Ikkala, J. Tuominen, and K. Keravuori. BiographySampo—publishing and enriching biographies on the semantic web for digital humanities research. *The Semantic Web: ESWC*, 2019. To appear.

[14] E. Hyvönen, P. Leskinen, M. Tamper, J. Tuominen, and K. Keravuori. Semantic National Biography of Finland. In E. Mäkelä, M. Tolonen, and J. Tuominen, editors, *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference, DHN 2018, Helsinki, Finland, March 7-9, 2018.*, volume 2084 of *CEUR Workshop Proceedings*, pages 372–385. CEUR-WS.org, 2018.

[15] A. Krisnadhi. The role patterns. 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 313–319. IOS Press, 2016.

[16] A. Krisnadhi and P. Hitzler. Modeling with ontology design patterns: Chess games as a worked example. 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 3–21. IOS Press, 2016.

[17] A. Krisnadhi and P. Hitzler. A core pattern for events. In K. Hammar, P. Hitlzer, A. Lawrynowicz, A. Krisnadhi, A. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 29–38. IOS Press / AKA Verlag, 2017.

[18] A. Krisnadhi and P. Hitzler. The stub metapattern. In K. Hammar, P. Hitlzer, A. Lawrynowicz, A. Krisnadhi, A. Nuzzolese, and M. Solanki, editors, *Advances in Ontology Design and Patterns*, volume 32 of *Studies on the Semantic Web*, pages 39–64. IOS Press / AKA Verlag, 2017.

[19] A. Krisnadhi, F. Maier, and P. Hitzler. OWL and Rules. In A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, and P. F. Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures*, volume 6848 of *Lecture Notes in Computer Science*, pages 382–415. Springer, Heidelberg, 2011.

[20] A. A. Krisnadhi, P. Hitzler, and K. Janowicz. On the capabilities and limitations of OWL regarding typecasting and ontology design pattern views. In V. A. M. Tamma, M. Dragoni, R. S. Gonçalves, and A. Lawrynowicz, editors, *Ontology Engineering – 12th International Experiences and Directions Workshop on OWL, OWLED 2015, co-located with ISWC 2015, Bethlehem, PA, USA, October 9-10, 2015, Revised Selected Papers*, volume 9557 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 2015.

[21] T. Lebo, S. Sahoo, and D. McGuinness, editors. *PROV-O: The PROV Ontology*. W3C Recommendation, 30 April 2013. Available at http://www.w3.org/TR/prov-o/.

[22] B. Motik, P. Patel-Schneider, and B. Parsia, editors. *Time Ontology in OWL*. W3C Recommendation, 19 October 2017. Available at https://www.w3.org/TR/owl-time/.

[23] M. A. Musen. The Protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.

[24] 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.

[25] M. K. Sarker, A. A. Krisnadhi, and P. Hitzler. OWLAx: A Protégé 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.

[26] C. Shimizu, Q. Hirt, and P. Hitzler. A Protégé plug-in for annotating OWL ontologies with OPLa. In A. Gangemi, A. L. Gentile, A. G. Nuzzolese, S. Rudolph, M. Maleshkova, H. Paulheim, J. Z. Pan, and M. Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events – ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, volume 11155 of *Lecture Notes in Computer Science*, pages 23–27. Springer, 2018.

[27] C. Shimizu, P. Hitzler, Q. Hirt, A. Sheill, S. Gonzalez, C. Foley, D. Rehberger, E. Watrall, W. Hawthorne, D. Tarr, R. Carty, and J. Mixter. The Enslaved Ontology 1.0: Peoples of the historic slave trade. Technical report, Enslaved: Peoples of the Historic Slave Trade, March 2019. Available from `http://docs.enslaved.org`.

[28] C. Shimizu, P. Hitzler, and C. Paul. Ontology design patterns for Winston's taxonomy of part-whole relations. In M. G. Skjæveland, Y. Hu, K. Hammar, V. Svátek, and A. Lawrynowicz, editors, *Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 9th, 2018.*, volume 2195 of *CEUR Workshop Proceedings*, pages 2–13. CEUR-WS.org, 2018.

[29] M. van Erp, J. de Does, K. Depuydt, R. Lenders, and T. van Goethem.

Slicing and dicing a newspaper corpus for historical ecology research. In C. Faron-Zucker, C. Ghidini, A. Napoli, and Y. Toussaint, editors, *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings*, volume 11313 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 2018.

[30] M. van Erp, M. Wevers, and H. C. Huurdeman. Constructing a recipe web from historical newspapers. In D. Vrandecic, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, and E. Simperl, editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2018.

[31] W. R. van Hage, V. Malaisé, R. Segers, L. Hollink, and G. Schreiber. Design and use of the Simple Event Model (SEM). *J. Web Semantics*, 9(2):128–136, 2011.

[32] D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.

# A Domain Ontology for Task Instructions

Aaron Eberhart✉[1][0000−0003−3007−5460], Cogan Shimizu[1][0000−0003−4283−8701], Christopher Stevens[2], Pascal Hitzler[1][0000−0001−6192−3472], Christopher W. Myers[2][0000−0003−0556−5935], and Benji Maruyama[3]

[1] DaSe Lab, Kansas State University, Manhattan, KS, USA
[2] Air Force Research Laboratory, Wright-Patterson AFB, OH, USA
[3] Air Force Research Laboratory, Materials & Manufacturing Directorate, Wright-Patterson AFB, OH, USA
{aaroneberhart, coganmshimizu, hitzler}@ksu.edu, {christopher.myers.29, christopher.stevens.28, benji.maruyama}@us.af.mil

**Abstract.** Knowledge graphs and ontologies represent information in a variety of different applications. One use case, the Intelligence, Surveillance, & Reconnaissance: Mutli-Attribute Task Battery (ISR-MATB), comes from Cognitive Science, where researchers use interdisciplinary methods to understand the mind and cognition. The ISR-MATB is a set of tasks that a cognitive or human agent perform which test visual, auditory, and memory capabilities. An ontology can represent a cognitive agent's background knowledge of the task it was instructed to perform and act as an interchange format between different Cognitive Agent tasks similar to ISR-MATB. We present several modular patterns for representing ISR-MATB task instructions, as well as a unified diagram that links them together.

## 1 Introduction

Knowledge graphs facilitate data integration across highly heterogeneous sources in a semantically useful way. Knowledge graphs may be equipped with a schema, frequently an ontology, that combines the associative power of the knowledge graph with the semantics of the ontology. Due to this, they are uniquely suited to support research in cognitive science, where it is often necessary to incorporate information from fields like computer science, psychology, neuroscience, philosophy, and more.

*Cognitive agents* are a sub-field of cognitive science and an application of the more broad study of cognitive architectures. Cognitive architectures, like ACT-R[**?**] for example, are an approach to understanding intelligent behavior and cognition that grew out of the idea of Unified Theories of Cognition[**?**]. These systems have their roots in AI production systems and some types use rules-based cognition. Many in Computer Science are familiar with inductive themes from a different type, called Connectionism, due to its historic ties with artificial neural networks. Symbolic cognitive architectures, by contrast, are less widely known outside of cognitive science, and are abstracted and explicit like logic programming.

Both ontologies and cognitive architectures deal with symbolic knowledge. Symbolic cognitive architectures typically focus on the plausibility of knowledge and the way in which that knowledge is translated into human behavior within a specific task. Ontologies offer a set of robust mechanisms for reasoning over complex knowledge bases and could help cognitive architectures adapt to tasks in novel environments. One way the two may be integrated is by leveraging the ontology to reduce the specificity of a cognitive agent.

In general, cognitive agents are often specialized, or *differentiated*, to perform a specific task or set of tasks. An *undifferentiated* agent is one that has no specialization. The purpose of such an agent is to be adaptable to new tasks as needed. As part of initial work to develop such an undifferentiated cognitive agent, we have developed a modular ontology that captures instructions for a specific cognitive agent task called ISR-MATB. We discuss this platform in more detail in the next section.

Currently, the ontology supports the memory of a cognitive agent by adding structure to its knowledge and providing new varieties of query-like recall. And due to design methodology used during the modeling process, the ontology is general enough that it could model other cognitive agent experiments, which could then be evaluated against each other in a structured way. This allows the ontology to act as an invaluable interchange format between researchers developing cognitive agents.

The rest of this paper is organized as follows. Section 2 provides a brief overview of the use-case: ISR-MATB. Section 3 provides an in-depth examination of the ontology. Finally, in Section 4, we briefly conclude and discuss next steps.

## 2 ISR-MATB

ISR-MATB is a series of cognitive tasks that could be completed by a Cognitive Agent or a human[**?**]. A trial starts with one very simple task, the evaluation then branches into two sub-tasks that relate back to the first task. After the two sub-tasks are complete the agent completes one final task requiring integration of remembered information from all previous tasks. The final task is made more difficult by the possibility of incorrect feedback as the agent learns. ISR-MATB is intended to be repeated for a fixed time so that researchers can observe changes in the agent's response time and develop better computational cognitive agents.

### 2.1 Psychomotor Vigilance Test

The Psychomotor Vigilance Test is one of the more basic cognitive tasks[**?**]. In this task, there is an area of the screen where a letter could appear. When the letter does appear an agent must press a button that acknowledges they have seen it. If the agent pushes the button too soon a false start is recorded and the task continues normally. If too much time passes before the agent pushes the button then the task will continue with the letter unacknowledged. The next two tasks reference this letter, so the agent is instructed to remember it.

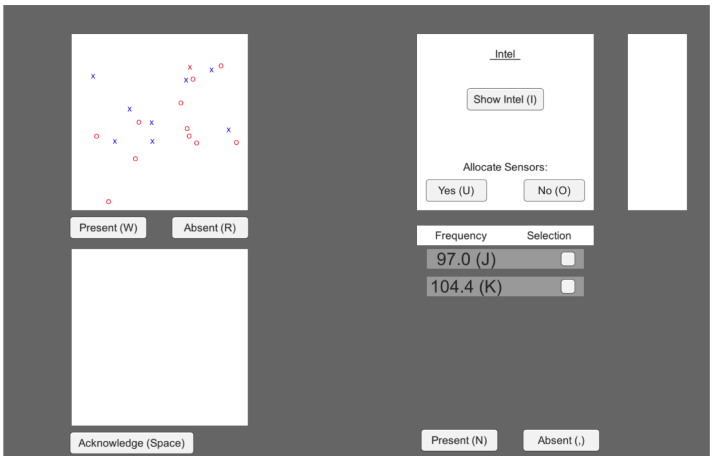



Fig. 1: An example depicting the four ISR-MATB tasks in a single interface.

### 2.2 Visual Search

The Visual Search task requires that the agent determine if the original letter is among a group of many letters that appear on the screen[**?**]. The other letters are distractors, and may be the same letter as the target with a different color, or the same color as the target with a different letter, or both color and letter different. The target may or may not appear among the distractors, and only ever appears once if at all. The agent pushes a button to indicate whether the the letter is present or absent.

### 2.3 Auditory Search

The Auditory Search task is very similar to the Visual Search, except of course that the agent must listen instead of look. In this task there are between one and four audio messages that each include a spoken color and letter. If one of the messages is the same as as the first task the agent pushes a button to indicate that it is present, otherwise they indicate that it is absent.

### 2.4 Decision Making

The final task, Decision Making, requires agents to infer a relationship between the outcomes of the Visual and Auditory Search tasks together with a new binary piece of information called “Intelligence” that appears after choosing whether to hypothetically allocate sensors or not. The rule the agent must guess is not too

hard, but it is complex enough that it must be learned by trial-and-error over multiple attempts. Learning the rule is made more difficult by the unlikely but not impossible event that the program responds incorrectly even when a correct answer is given. Responding 'yes' or 'no' to this sub-task ends one ISR-MATB trial.

## 3 Ontology Description

In this section we present the Instruction Ontology, a domain ontology built for use with the ISR-MATB experiment platform. This ontology was produced by following the Modular Ontology Modeling (MOM) methodology, outlined in [**?**,**?**] MOM is designed to ensure the high quality and reusability of the resulting ontology, both in terms of scope and in terms of granularity, which is a desired outcome.

The ontology consists of six modules: ISR-MATB Experiment, Instruction, SituationDescription, ItemRole, Action, and Affordance. For each module, we describe its purpose, provide a schema diagram,[4] and state its axiomatization in both description logic syntax and natural language. The OWL file for this ontology can be found online.[5] Figure 5 shows the schema diagram for the entire ontology.

### 3.1 ISR-MATB Experiment.

The ISR-MATB Experiment module is the core module for the ontology. The two main classes are ISR-MATB Experiment and ISR-MATB Task. As noted in Section 2, an experiment consists of up to four tasks that may require that information be carried between them, where each Task resides in a specific quadrant of the interface. Each Task provides roles to different Items, as well as a set of Instructions for the agent to carry out. We discuss these classes in more detail in their respective Module sections. The schema diagram for this module is shown in Figure 2c.

**Axiomatization:**

$$\top \sqsubseteq \forall \mathsf{affords.Affordance} \tag{1}$$

$$\mathsf{ISR\text{-}MATBTask} \sqsubseteq \geq 1\ \mathsf{hasInstruction.Instruction} \tag{2}$$

$$\mathsf{ISR\text{-}MATBExperiment} \sqsubseteq \leq 4\ \mathsf{hasTask.ISR\text{-}MATBTask} \tag{3}$$

$$\top \sqsubseteq \forall \mathsf{hasLocation.Location} \tag{4}$$

$$\top \sqsubseteq \forall \mathsf{hasName.xsd:string} \tag{5}$$

[4] A schema diagram is an informal, but intuitive way for conveying information about the structure and contents of an ontology. We use a consistent visual syntax for convenience, detailed in Figure 2.

[5] See `https://raw.githubusercontent.com/undiffagents/uagent/develop/ontology/uagent.owl`.

$$\text{ISR-MATBTask} \sqsubseteq =1\ \text{hasName.xsd:string} \quad (6)$$
$$\text{ISR-MATBTask} \sqsubseteq \forall \text{providesRole.ItemRole} \quad (7)$$
$$\text{ISR-MATBTask} \sqsubseteq \forall \text{informs.ISR-MATBTask} \quad (8)$$

**Explanation of axioms above:**

1. Range. The range of affords is Affordance.
2. Minimum Cardinality. An ISR-MATBTask has at least one Instruction.
3. Maximum Cardinality. An ISR-MATBExperiment consists of at most four ISR-MATBTasks.
4. Range. The range of hasLocation is Location.
5. Range. The range of hasName is xsd:string.
6. Scoped Range. The range of providesRole is ItemRole when the domain is ISR-MATBTask.
7. Scoped Range. The range of informs is ISR-MATBTask when the domain is ISR-MATBTask.

### 3.2 Action

The Action module is an instantiation of the Explicit Typing meta-pattern described in [**?**].[6]

In this case, we use a class, ActionType, to represent a controlled vocabulary. We believe that using a controlled vocabulary to represent this type information is less invasive to the ontology. This way, adding or removing types of actions from the controlled vocabulary does not actually change the ontology. Some instances of the controlled vocabulary are listed in Figure 5.

An Action, in this context, is the physical, actual action that takes place to transition between different states of the experiment, e.g. 'the action of clicking a button.' The schema diagram for this module is shown in Figure 2a.

**Axiomatization:**

$$\text{Action} \sqsubseteq =1\text{ofType.ActionType} \quad (1)$$

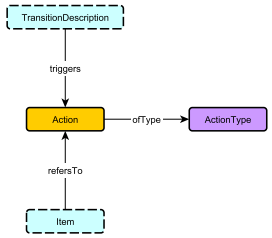
**Explanation of axioms above:**

1. Exact Cardinality. An Action has exactly one ActionType.

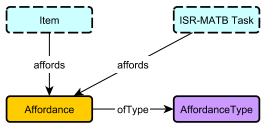
### 3.3 Affordance

The Affordance module is also instantiated from the Explicit Typing meta-pattern, explained in more detail in Section 3.2 and [**?**]. An Affordance is essentially some quality of an Item that indicates that "something" may be done with it. Familiar examples might include *clickable* buttons or text highlighted in blue (perhaps indicating that it's a hyperlink). Instances of the AffordanceType can be found in Figure 5. The schema diagram for this module is shown in Figure 2b.

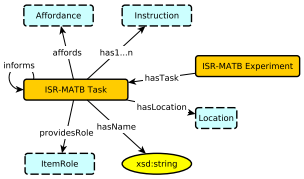
[6] [**?**] is a modular ontology design library; it contains a set of frequently used patterns and respective documentation.



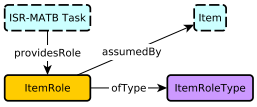

(a) The schema diagram for the Action module.




(b) The schema diagram for the Affordance module.




(c) The schema diagram for the ISR-MATB Experiment module.




(d) The schema diagram for the ItemRole module.

Fig. 2: Orange boxes are classes and indicate that they are central to the diagram. Blue dashed boxes indicate a reference to another diagram, pattern, or module. Gray frames with a dashed outline contain modules. Arrows depict relations and open arrows represent subclass relations. Yellow ovals indicate data types (and necessarily, arrows pointing to a datatype are data properties). Finally, purple boxes represent controlled vocabularies. That is, they represent a controlled set of IRIs that are of that type.

**Axiomatization:**

$$\mathsf{Affordance} \sqsubseteq =1\mathsf{hasAffordanceType}.\mathsf{AffordanceType} \tag{1}$$

**Explanation of axioms above:**

1. Exact cardinality. An Affordance has exactly one AffordanceType.

### 3.4 ItemRole

The ItemRole module is an instantiation of the AgentRole pattern, which may also be found in [**?**]. We also equip it with an explicit type, in the same manner as Action and Affordance.

Each ISR-MATB Task may provide roles to Items. That is, certain items may be a target or distractor, but not always. This allows us to assign certain roles to items that may, if they were qualities, be ontologically disjoint. The schema diagram for this module is shown in Figure 2d.

**Axiomatization:**

$$\text{ISR-MATBTask} \sqsubseteq \forall \text{providesRole.ItemRole} \quad (1)$$

$$\top \sqsubseteq \forall \text{hasItemRoleType.ItemRoleType} \quad (2)$$

$$\text{ItemRole} \sqsubseteq \forall \text{assumedBy.Item} \quad (3)$$

$$\text{ItemRole} \sqsubseteq \exists \text{assumedBy.Item} \quad (4)$$

**Explanation of axioms above:**

1. Scoped Range. The range of providesRole is ItemRole when the domain is ISR-MATBTask.
2. Range. The range of hasItemRoleType is ItemRoleType.
3. Scoped Range. ItemRoles are assumedBy Items.
4. Existential. Every ItemRole is assumedBy an Item.

### 3.5 SituationDescription

For this module, we opted to use the Situation and Description approach. We chose to use this conceptualization due to the non-linear nature of the instructions.[7] That is, an ISR-MATB Task is not a sequence of instructions, but a collection of directions or descriptions.

An Instruction, is a description of a way to transition between two states. In order to follow out an instruction the state described in the the pre-SituationDescription would need to be met. Following through would result in a new state, the Post-Situtation Description.

Furthermore, the SituationDescription will indicate the presence, or absence, of an item, as well as its description. Descriptions, in this case, are relegated to controlled vocabularies in the same manner as Affordance or Action. We call this an ItemDescription because it is inherent to the Instruction and not the Item, itself.

The schema diagram for this module is shown in Figure 3.

**Axiomatization:**

$$\text{SituationDescription} \sqsubseteq \forall \text{hasCurrentCondition.}(\text{RelativeCondition} \sqcup \text{ItemDescription}) \quad (1)$$

$$\text{SituationDescription} \sqsubseteq \forall \text{hasEarlierCondition.ItemDescription} \quad (2)$$

[7] For a deeper discussion on Descriptions, Situations, and Plans, see [**?**].

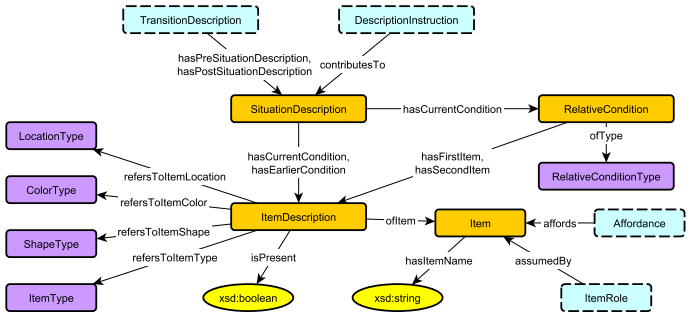



Fig. 3: The schema diagram for the SchemaDiagram module. Color and shape usage is the same as in previous diagrams.

$$\top \sqsubseteq \forall \mathsf{hasRelativeConditionType}.\mathsf{RelativeConditionType} \tag{3}$$
$$\mathsf{RelativeCondition} \sqsubseteq \forall \mathsf{hasFirstItem}.\mathsf{ItemDescription} \tag{4}$$
$$\mathsf{RelativeCondition} \sqsubseteq \forall \mathsf{hasSecondItem}.\mathsf{ItemDescription} \tag{5}$$
$$\mathsf{ItemDescription} \sqsubseteq \forall \mathsf{ofItem}.\mathsf{Item} \tag{6}$$
$$\mathsf{ItemDescription} \sqsubseteq =1\ \mathsf{isPresent}.\mathsf{xsd{:}boolean} \tag{7}$$
$$\top \sqsubseteq \forall \mathsf{refersToItemLocation}.\mathsf{LocationType} \tag{8}$$
$$\top \sqsubseteq \forall \mathsf{refersToItemColor}.\mathsf{ColorType} \tag{9}$$
$$\top \sqsubseteq \forall \mathsf{refersToShapeType}.\mathsf{ShapeType} \tag{10}$$
$$\top \sqsubseteq \forall \mathsf{refersToItemType}.\mathsf{ItemType} \tag{11}$$
$$\mathsf{ItemDescription} \sqsubseteq \geq 0\ \mathsf{refersToItemLocation}.\mathsf{LocationType} \tag{12}$$
$$\mathsf{ItemDescription} \sqsubseteq \geq 0\ \mathsf{refersToItemColor}.\mathsf{ColorType} \tag{13}$$
$$\mathsf{ItemDescription} \sqsubseteq \geq 0\ \mathsf{refersToItemShape}.\mathsf{ShapeType} \tag{14}$$
$$\mathsf{ItemDescription} \sqsubseteq \geq 0\ \mathsf{refersToItemType}.\mathsf{ItemType} \tag{15}$$
$$\top \sqsubseteq \forall \mathsf{hasItemName}.\mathsf{xsd{:}string} \tag{16}$$
$$\exists \mathsf{hasItemName}.\top \sqsubseteq \mathsf{Item} \tag{17}$$

**Explanation of axioms above:**

1. Scoped Range. The range of hasCurrentCondition is a RelativeCondition or ItemDescription when the domain is SituationDescription.
2. Scoped Range. The range of hasEarlierCondition is ItemDescription when the domain is SituationDescription.
3. Range. The range of hasRelativeConditionType is RelativeConditionType.

4. Scoped Range. The range of hasFirstItem is ItemDescription when the domain is RelativeCondition.
5. Scoped Range. The range of hasSecondItem is ItemDescription when the domain is RelativeCondition.
6. Scoped Range. The range of ofItem is Item when the domain is ItemDescription.
7. Scoped Range. An ItemDescription has exactly one Boolean flag indicating whether or not it is present.
8. Range. The range of refersToItemLocation is LocationType.
9. Range. The range of refersToItemColor is ColorType.
10. Range. The range of refersToItemShape is ShapeType.
11. Range. The range of refersToItemType is ItemType.
12. Structural Tautology. An ItemDescription may refer to a LocationType.
13. Structural Tautology. An ItemDescription may refer to a ColorType.
14. Structural Tautology. An ItemDescription may refer to a ShapeType.
15. Structural Tautology. An ItemDescription may refer to an ItemType.
16. Range. The range of hasItemName is xsd:string.
17. Domain Restriction. The domain of hasItemName is restricted to Items.

### 3.6 Instruction

Instructions are the atomic units of a task. They come in two varieties: descriptions and actions. The former are instructions that are prescriptive or descriptive. They are statements that indicate information about the environment or the task. They may, in natural language, take such form as "There is a button named 'Present'." The latter type of instruction instructs when or where to do something. For example, "Press the button if a high-pitched tone is heard." An Action-Instruction prescribes some transition between descriptions of situations, whereas Description-Instructions directly contribute to said SituationDescription. The module also uses a data property to capture the natural language formulation of the Instruction. The schema diagram for this module is shown in Figure 4.

**Axiomatization:**

$$\mathsf{ActionInstruction} \sqsubseteq \mathsf{Instruction} \tag{1}$$

$$\mathsf{ActionInstruction} \sqsubseteq \forall \mathsf{prescribes.TransitionDescription} \tag{2}$$

$$\top \sqsubseteq \forall \mathsf{asString.xsd{:}string} \tag{3}$$

$$\mathsf{Instruction} \sqsubseteq \geq 0\ \mathsf{asString.xsd{:}string} \tag{4}$$

$$\mathsf{DescriptionInstruction} \sqsubseteq \mathsf{Instruction} \tag{5}$$

$$\mathsf{DescriptionInstruction} \sqsubseteq \forall \mathsf{contributesTo.SituationDescription} \tag{6}$$

$$\top \sqsubseteq \forall \mathsf{hasPreSituationDescription.SituationDescription} \tag{7}$$

$$\top \sqsubseteq \forall \mathsf{hasPostSituationDescription.SituationDescription} \tag{8}$$

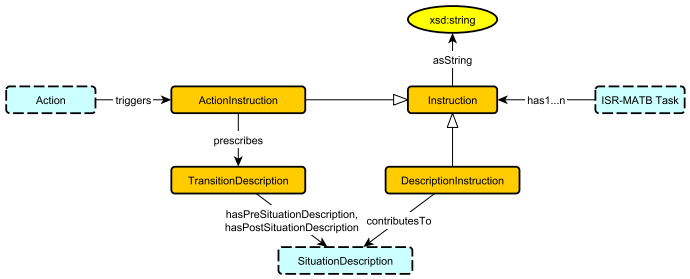



Fig. 4: The schema diagram for the Instruction module. Color and shape usage is the same as in previous diagrams.

**Explanation of axioms above:**

1. Subclass. Every ActionInstruction is an Instruction.
2. Scoped Range. The range of prescribes is TransitionDescription when the domain is ActionInstruction.
3. Range. The range of asString is xsd:string.
4. Structural Tautology. An Instruction may have a string representation.
5. Subclass. Every DescriptionInstruction is an Instruction.
6. Scoped Range. The range of contributesTo is SituationDescription when the domain is DescriptionInstruction.
7. Range. The range of hasPreSituationDescription is SituationDescription.
8. Range. The range of hasPostSituationDescription is SituationDescription.

## 4 Conclusion

In this paper we have presented an ontology for modeling the ISR-MATB cognitive agent task instructions. This ontology can be used, as we have, to directly support the memory of a cognitive agent performing tasks. It also could support experiment design, irrespective of any agent, by providing a structured basis for evaluating similar tasks. The modular structure facilitates adapting the ontology to other use cases and scenarios by replacing or adapting the existing modules. It is also possible to create new modules from the referenced patterns via template-based instantiation[**?**].

### 4.1 Future Work

In the future we plan to extend this ontology so that it can support a fully undifferentiated agent. This will include tasks like ISR-MATB, but also many

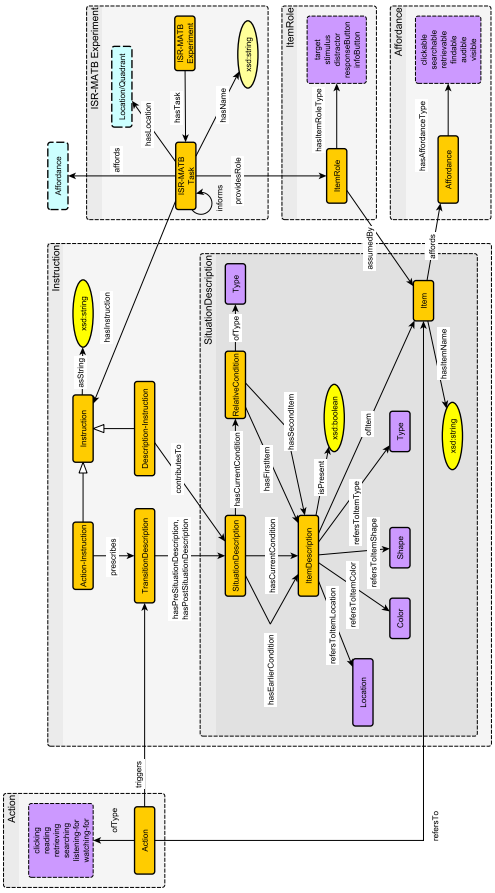



Fig. 5: The schema diagram for the entire ontology. Note that the SituationDescription module is nested in the Instruction Module. Color and shape usage is the same as in previous diagrams.

others that could be very different. One such task is supporting materials science research that uses the Autonomous Research System (ARES) framework. An undifferentiated cognitive agent could operate a robotic system that performs research, using software like ARES, saving materials researchers hours of potentially hazardous lab work.

*Acknowledgement* This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0386.

# 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.

# 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.

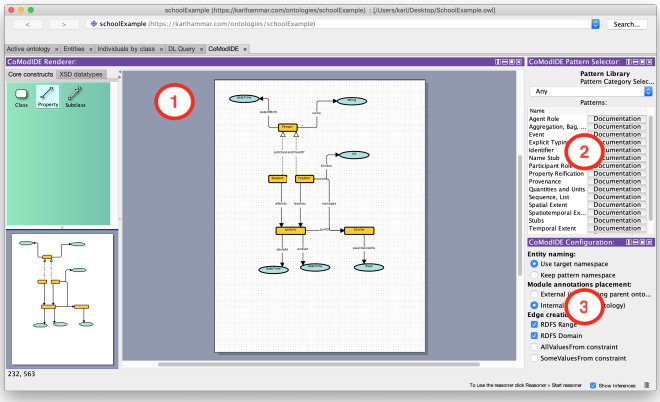



**Fig. 1.** CoModIDE User Interface featuring 1) the schema editor, 2) the pattern library, and 3) the configuration view.

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)

# Modular Graphical Ontology Engineering Evaluated

Cogan Shimizu[1][0000−0003−4283−8701] ✉, Karl Hammar[2][0000−0001−8767−4136], and Pascal Hitzler[1][0000−0001−6192−3472]

[1] Data Semantics Lab, Kansas State University, USA
{coganmshimizu, phitzler}@ksu.edu
[2] Jönköping AI Lab, Jönköping University, Sweden
karl.hammar@jail.ai

**Abstract.** Ontology engineering is traditionally a complex and time-consuming process, requiring an intimate knowledge of description logic and predicting non-local effects of different ontological commitments. Pattern-based modular ontology engineering, coupled with a graphical modeling paradigm, can help make ontology engineering accessible to modellers with limited ontology expertise. We have developed CoModIDE, the Comprehensive Modular Ontology IDE, to develop and explore such a modeling approach. In this paper we present an evaluation of the CoModIDE tool, with a set of 21 subjects carrying out some typical modeling tasks. Our findings indicate that using CoModIDE improves task completion rate and reduces task completion time, compared to using standard Protégé. Further, our subjects report higher System Usability Scale (SUS) evaluation scores for CoModIDE, than for Protégé. The subjects also report certain room for improvements in the CoModIDE tool – notably, these comments all concern comparatively shallow UI bugs or issues, rather than limitations inherent in the proposed modeling method itself. We deduce that our modeling approach is viable, and propose some consequences for ontology engineering tool development.

## 1 Introduction

Building a knowledge graph, as with any complex system, is an expensive endeavor, requiring extensive time and expertise. For many, the magnitude of resources required for building and maintaining a knowledge graph is untenable. Yet, knowledge graphs are still poised to be a significant disruptor in both the private and public sectors [17]. As such, lowering the barriers of entry is very important. More specifically, it will be necessary to increase the approachability of knowledge graph development best practices, thus reducing the need for dedicated expertise. Of course, we do not mean imply that *no* expertise is desirable, simply that a dedicated knowledge engineer may be out of reach for small firms or research groups. For this paper, we focus on the best practices according to the eXtreme design (XD) [4] and modular ontology modeling (MOM) [12] paradigms. To this point, we are interested in how tooling infrastructure can

improve approachability. In the context of our chosen paradigms and focus on tooling infrastructure, approachability may be proxied by *the amount of effort to produce correct and reasonable output*, where effort is a function of tool-user experience (UX) and time taken. Furthermore, by using tooling infrastructure to encapsulate best practices, it improves the maintainability and evolvability accordingly.

In particular, this paper investigates the use of a graphical modeling tool that encapsulates the pattern-driven philosophies of XD and MOM. To do so, we have developed CoModIDE (the *Comprehensive Modular Ontology IDE* – pronounced "commodity"), a plugin for the popular ontology editing platform, Protégé [16]. In order to show that CoModIDE improves approachability of knowledge graph development, we have formulated for the following hypotheses.

H1. When using CoModIDE, a user takes less time to produce correct and reasonable output, than when using Protege.
H2. A user will find CoModIDE to have a higher SUS score than when using Protege alone.

The remainder of this paper is organized as follows. Section 2 presents CoModIDE. Section 3 discusses related work on graphical modeling and ontology design pattern use and development. We present our experimental design in Section 4, our results in Section 5, and a discussion of those results and their implications in Section 6. Finally, Section 7 concludes the paper, and suggests possibilities for future research.

## 2 CoModIDE: A Comprehensive Modular Ontology IDE

### 2.1 Motivator: A Graphical and Modular Ontology Design Process

CoModIDE is intended 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. Such users (and, indeed also more advanced users) often prefer to do initial modeling in pair or group settings, and to do it graphically – whether that be on whiteboards, in vector drawing software, or even on paper. This further limits the modeling constructs to those that can be expressed somewhat intuitively using graphical notations (such that all involved participants, regardless of their ontology engineering skill level, can understand and contribute).

This initial design process typically iterates rapidly and fluidly, with the modeling task being broken down into individual problems of manageable complexity[3]; candidate solutions to these problem pieces being drawn up, analysed

[3] We find that the size of such partial solutions typically fit on a medium-sized whiteboard; but whether this is a naturally manageable size for humans to operate with, or whether it is the result of constraints of or conditioning to the available tooling, i.e., the size of the whiteboards often mounted in conference rooms, we cannot say.

and discussed; a suitable solution selected and documented; and the next step of the problem then tackled. Many times, the formalization of the developed solution into an OWL ontology is carried out after-the-fact, by a designated ontologist with extensive knowledge of both the language and applicable tooling. However, this comes at a cost, both in terms of hours expended, and in terms of the risk of incorrect interpretations of the previously drawn graphical representations (the OWL standard does not define a graphical notation syntax, so such representations are sometimes ambiguous).

The design process discussed above mirrors the principles of *eXtreme Design* (XD) [4]: working in pairs, breaking apart the modeling task into discrete problems, and iterating and refactoring as needed. XD also emphasizes the use of *Ontology Design Patterns* (ODPs) as solutions to frequently recurring modeling problems. Combining ODP usage with the graphical modeling process discussed above (specifically with the need to in an agile manner refactor and modify partial solutions) requires that the partial solutions (or *modules*) derived from ODPs are annotated, such that they can at a later time be isolated for study, modified, or replaced.

In summary it would be useful for our target user group if there were tooling available that supported 1) intuitive and agile graphical modeling, directly outputting OWL ontologies (avoiding the need for the aforementioned post-processing), and 2) reuse of ODPs to create and maintain ODP-based modules. Hence, CoModIDE.

### 2.2 Design and Features

The design criteria for CoModIDE, derived from the requirements discussed above, are as follows:

- 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 property 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[4]. If the user wishes, they should be able to load a separately downloaded ODP repository, to replace or complement the built-in one.

[4] Our experience indicates that while our target 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.

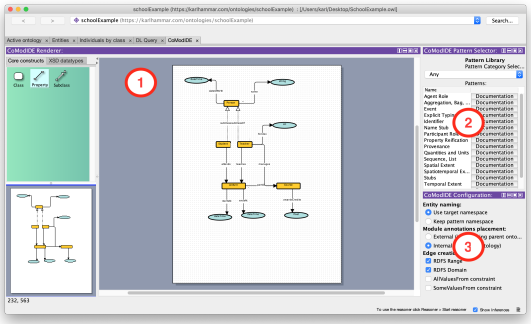



Fig. 1: CoModIDE User Interface featuring 1) the schema editor, 2) the pattern library, and 3) the configuration view.

– 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. The resulting ontology modules should maintain their coherence and be treated like modules in a consistent manner across restarts, machines, etc. 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 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.

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

[5] `http://ontologydesignpatterns.org/`
[6] `https://comodide.com`

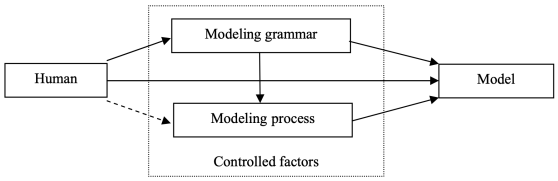



Fig. 2: Factors affecting conceptual modeling, from [9].

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 7) be manipulated (folded, unfolded, removed, annotated) as needed.

## 3 Related Work

**Graphical Conceptual Modeling** [9] proposes three factors (see Figure 2) that influence the construction of a conceptual model, such as an ontology; namely, the *person* doing the modeling (both their experience and know-how, and their interpretation of the world, of the modeling task, and of model quality in general), the *modeling grammar* (primarily its expressive power/completeness and its clarity), and the *modeling process* (including both initial conceptualisation and subsequent formal model-making). Crucially, only the latter two factors can feasibly be controlled in academic studies. The related work discussed below tends to focus on one or the other of these factors, i.e., studying the characteristics of a modeling language *or* a modeling process. Our work on CoModIDE straddles this divide: employing graphical modeling techniques reduces the grammar available from standard OWL to those fragments of OWL that can be represented intuitively in graphical format; employing design patterns affects the modeling process.

Graphical modeling approaches to conceptual modeling have been extensively explored and evaluated in fields such as database modeling, software engineering, business process modeling, etc. Studying model grammar, [22] compares EER notation with an early UML-like notation from a comprehensibility point-of-view. This work observes that restrictions are easier to understand in a notation where they are displayed coupled to the types they apply to, rather than the relations they range over. [7] proposes a quality model for EER diagrams that can also extend to UML. Some of the quality criteria in this model, that are relevant in graphical modeling of OWL ontologies, include *minimality* (i.e., avoiding duplication of elements), *expressiveness* (i.e., displaying all of the required elements), and simplicity (displaying no more than the required elements).

[1] study the usability of UML, and report that users perceive UML class diagrams (closest in intended use to ontology visualizations) to be less easy-to-

use than other types of UML diagrams; in particular, relationship multiplicities (i.e., cardinalities) are considered frustrating by several of their subjects. UML displays such multiplicities by numeric notation on the end of connecting lines between classes. [13] analyses UML and argues that while it is a useful tool in a design phase, it is overly complex and as a consequence, suffers from redundancies, overlaps, and breaks in uniformity. [13] also cautions against using difficult-to-read and -interpret adornments on graphical models, as UML allows.

Various approaches have been developed for presenting ontologies visually and enabling their development through a graphical modeling interface, the most prominent of which is probably *VOWL*, the *Visual Notation for OWL Ontologies* [15], and its implementation viewer/editor WebVOWL [14,23]. VOWL employs a force-directed graph layout (reducing the number of crossing lines, increasing legibility) and explicitly focuses on usability for users less familiar with ontologies. As a consequence of this, VOWL renders certain structures in a way that, while not formally consistent with the underlying semantics, supports comprehensibility; for instance, datatype nodes and `owl:Thing` nodes are duplicated across the canvas, so that the model does not implode into a tight cluster around such often used nodes. It has been evaluated over several user studies with users ranging from laymen to more experienced ontologists, with results indicating good comprehensibility. CoModIDE has taken influence from VOWL, e.g., in how we render datatype nodes. However, in a collaborative editing environment in which the graphical layout of nodes and edges needs to remain consistent for all users, and relatively stable over time, we find the force-directed graph structure (which changes continuously as entities are added/removed) to be unsuitable.

For such collaborative modeling use cases, the commercial offering *Grafo*[7] offers a very attractive feature set, combining the usability of a VOWL-like notation with stable positioning, and collaborative editing features. Crucially, however, Grafo does not support pattern-based modular modeling, and as a web-hosted service, does not allow for customizations or plugins that would support such a modeling paradigm.

CoModIDE is partially based on the Protégé plugin *OWLAx*, as presented in [19]. This plugin supports one-way translation from graphical schema diagrams drawn by the user, into OWL ontology classes and properties; however, it does not render such constructs back into a graphical form. There is thus no way of continually maintaining and developing an ontology using only OWLAx. There is also no support for design pattern reuse in this tool.

**Ontology Design Patterns** Ontology Design Patterns (ODPs) were introduced by Gangemi [8] and Blomqvist & Sandkuhl [2] in 2005, as a means of simplifying ontology development. ODPs are intended to guide non-expert users, by packaging best practices into reusable blocks of functionality, to be adapted and specialised by those users in individual ontology development projects. Presutti et al.[18] defines a typology of ODPs, including patterns for reasoning, naming, transformation, etc. The eXtreme Design methodology [4] describes how

[7] `https://gra.fo`

ontology engineering projects can be broken down into discrete sub-tasks, to be solved by using ODPs. Prior studies indicate that the use of ODPs can lower the number of modeling errors and inconsistencies in ontologies, and that they are by the users perceived as useful and helpful [3,5].

Applying the XD method and ODPs requires the availability of both high-quality ODPs, and of tools and infrastructure that support ODP use. Recent work in this area, by the authors and others, includes *XDP*, a fork of the WebProtégé ontology editor [10]; the *OPLa* annotations vocabulary that models how ontology concepts can be grouped into modules, and the provenance of and interrelations between such modules, including to ODPs [11]; and the MODL library, a curated and specially documented collection of high-quality patterns for use in many domains [21]. CoModIDE draws influence from all of these works, and includes the MODL library as its default pattern library, using an OPLa-based representation of those patterns.

## 4 Research Method

Our experiment is comprised of four steps: a survey to collect subject background data (familiarity with ontology languages and tools), two modeling tasks, and a follow-up survey to collect information on the usability of both Protégé and CoModIDE. The tasks were designed to emulate a common ontology engineering process, where a conceptual design is developed and agreed upon by whiteboard prototyping, and a developer is then assigned to formalizing the resulting whiteboard schema diagram into an OWL ontology.

During each of the modeling tasks, participants are asked to generate a *reasonable* and *correct* OWL file for the provided schema diagram. In order to prevent a learning effect, the two tasks utilize two different schema diagrams. To prevent bias arising from differences in task complexity, counterbalancing was employed (such that half the users performed the first task with standard Protégé and the second task with CoModIDE, and half did the opposite). The correctness of the developed OWL files, and the time taken to complete each tasks, were recorded (the latter was however, for practical reasons, limited to 20 minutes per task).

The following sections provide a brief overview of each the steps. The source material for the entire experiment is available online[8].

**Introductory Tutorial** As previously mentioned, our intent is to improve the approachability of ontology modeling by making it more accessible to those without expertise in knowledge engineering. As such, when recruiting our participants for this evaluation, we did not place any requirements on ontology modeling familiarity. However, to establish a shared baseline knowledge of foundational modeling concepts (such as one would assume participants would have in the situation we try to emulate, see above), we provided a 10 minute tutorial

[8] `http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-47887`

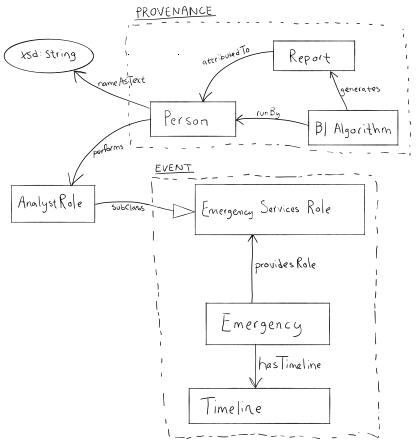



Fig. 3: Task A Schema Diagram

on ontologies, classes, properties, domains, and ranges. The slides used for this tutorial may be found online with the rest of the experiment's source materials.

***a priori* Survey** The purpose of the *a priori* survey was to collect information relating to the participants base level familiarity with topics related to knowledge modeling, to be used as control variables in later analysis. We used a 5-point Likert scale for rating the accuracy of the following statements.

CV1. I have done ontology modeling before.
CV2. I am familiar with Ontology Design Patterns.
CV3. I am familiar with Manchester Syntax.
CV4. I am familiar with Description Logics.
CV5. I am familiar with Protégé.

Finally, we asked the participants to describe their relationship to the test leader, (e.g. student, colleague, same research lab, not familiar).

**Modeling Task A** In Task A, participants were to develop an ontology to model how an analyst might generate reports about an ongoing emergency. The scenario identified two design patterns to use:

- **Provenance**: to track who made a report and how;
- **Event**: to capture the notion of an emergency.

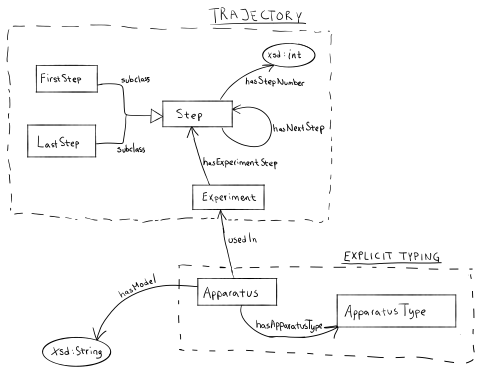



Fig. 4: Task B Schema Diagram

Figure 3 shows how these patterns are instantiated and connected together. Overall the schema diagram contains seven concepts, one datatype, one subclass relation, one data property, and six object properties.

**Modeling Task B** In Task B, participants were to develop an ontology to capture the steps of an experiment. The scenario identified two design patterns to use:

- **Trajectory**: to track the order of the steps;
- **Explicit Typing**: to easily model different types of apparatus.

Figure 4 shows how these patterns are instantiated and connected together. Overall, the schema diagram contains six concepts, two datatypes, two subclass relations, two data properties, and four object properties (one of which is a self-loop).

***a posteriori* Survey** The *a posteriori* survey included the SUS evaluations for both Protégé and CoModIDE. The SUS is a very common "quick and dirty," yet reliable tool for measuring the usability of a system. It consists of ten questions, the answers to which are used to compute a total usability score of 0–100. Additional information on the SUS and its included questions can be found online.[9]

Additionally, we inquire about CoModIDE-specific features. These statements are also rated using a Likert scale. However, we do not use this data in our evaluation, except to inform our future work, as described in Section 7. Finally, we requested any free-text comments on CoModIDE's features.

[9] `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`

Table 1: Mean, standard deviation, relative standard deviation, and median responses to *a priori* statements

| | mean | $\sigma$ | relative $\sigma$ | median |
|---|---|---|---|---|
| CV1: I have done ontology modeling before | 3.05 | 1.75 | 57 % | 3 |
| CV2: I am familiar with Ontology Design Patterns | 3.05 | 1.32 | 43 % | 3 |
| CV3: I am familiar with Manchester Syntax | 2.33 | 1.56 | 67 % | 1 |
| CV4: I am familiar with Description Logics | 2.81 | 1.33 | 47 % | 3 |
| CV5: I am familiar with Protégé | 2.95 | 1.63 | 55 % | 3 |

## 5 Results

### 5.1 Participant Pool Composition

Of the 21 subjects, 12 reported some degree of familiarity with the authors, while 9 reported no such connection. In terms of self-reported ontology engineering familiarity, the responses are as detailed in Table 1. It should be observed that responses vary widely, with a relative standard deviation ($\sigma$/mean) of 43–67 %.

### 5.2 Metric Evaluation

We define our two metrics as follows:

- **Time Taken:** number of minutes, rounded to the nearest whole minute and capped at 20 minutes due to practical limitations, taken to complete a task;
- **Correctness** is a discrete measure that corresponds to the structural accuracy of the output. That is, 2 points were awarded to those structurally accurate OWL files, when accounting for URIs; 1 point for a borderline case (e.g one or two incorrect linkages, or missing a domain statement but including the range); and 0 points for any other output.

For these metrics, we generate simple statistics that describe the data, per modeling task. Tables 2a and 2b show the mean, standard deviation, and median for the Time Taken and Correctness of Output, respectively.

In addition, we examine the impact of our control variables (CV). This analysis is important, as it provides context for representation or bias in our data set. These are reported in Table 2c. CV1-CV5 correspond exactly to those questions asked during the *a priori* Survey, as described in Section 4. For each CV, we calculated the bivariate correlation between the sample data and the self-reported data in the survey. We believe that this is a reasonable measure of impact on effect, as our limited sample size is not amenable to partitioning. That is, the partitions (as based on responses in the *a priori* survey) could have been tested pair-wise for statistical significance. Unfortunately, the partitions would have been too small to conduct proper statistical testing. However, we do caution that correlation effects are strongly impacted by sample size.

We analyze the SUS scores in the same manner. Table 4 presents the mean, standard deviation, and median of the data set. The maximum score while using

Table 2: Summary of statistics comparing Protege and CoModIDE.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 17.44 | 3.67 | 20.0 |
| CoModIDE | 13.94 | 4.22 | 13.5 |

(a) Mean, standard deviation, and median *time taken* to complete each modeling task.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 0.50 | 0.71 | 0.0 |
| CoModIDE | 1.33 | 0.77 | 1.5 |

(b) Mean, standard deviation, and median *correctness of output* for each modeling task.

| | CV1 | CV2 | CV3 | CV4 | CV5 |
|---|---|---|---|---|---|
| TT (P) | -0.61 | -0.18 | -0.38 | -0.58 | -0.62 |
| Cor. (P) | 0.50 | 0.20 | 0.35 | 0.51 | 0.35 |
| TT (C) | 0.02 | -0.34 | -0.28 | -0.06 | 0.01 |
| Cor. (C) | -0.30 | 0.00 | -0.12 | -0.33 | -0.30 |

(c) Correlations control variables (CV) on the Time Taken (TT) and Correctness of Output (Cor.) for both tools Protégé (P) and CoModIDE (C).

| | CV1 | CV2 | CV3 | CV4 | CV5 |
|---|---|---|---|---|---|
| SUS (P) | 0.70 | 0.52 | 0.64 | 0.73 | 0.64 |
| SUS (C) | -0.34 | -0.05 | -0.08 | -0.29 | -0.39 |

(d) Correlations with control variables (CV) on the SUS scores for both tools Protégé (P) and CoModIDE (C).

the scale is a 100. Table 2d presents our observed correlations with our control variables.

Finally, we compare the each metric for one tool against the other. That is, we want to know if our results are statistically significant—that as the statistics suggest in Table 2, CoModIDE does indeed perform better for both metrics and the SUS evaluation. To do so, we calculate the probability $p$ that the samples from each dataset come from different underlying distributions. A common tool, and the tool we employ here, is the Paired (two-tailed) T-Test—noting that it is reasonable to assume that the underlying data are normally distributed, as well as powerful tool for analyzing datasets of limited size. The threshold for indicating confidence that the difference is significant is generally taken to be $p < 0.05$. Table 3 summarizes these results.

### 5.3 Free-text Responses

18 of the 21 subjects opted to leave free-text comments. We applied fragment-based qualitative coding and analysis on these comments. I.e., we split the comments apart per the line breaks entered by the subjects, we read through the

Table 3: Significance of results.

| Time Taken | Correctness | SUS Evaluation |
|---|---|---|
| $p \approx 0.025 < 0.05$ | $p \approx 0.009 < 0.01$ | $p \approx 0.0003 < 0.001$ |

Table 4: Mean, standard deviation, and median SUS score for each tool. The maximum score is 100.

| | mean | $\sigma$ | median |
|---|---|---|---|
| Protégé | 36.67 | 22.11 | 35.00 |
| CoModIDE | 73.33 | 16.80 | 76.25 |

fragments and generated a simple category scheme, and we then re-read the fragments and applied these categories to the fragments (allowing at most one category per fragment) [6,20]. The subjects left between 1–6 fragments each for a total of 49 fragments for analysis, of which 37 were coded, as detailed in Table 5.

Of the 18 participants who left comments, 3 left comments containing no codable fragments; these either commented upon the subjects own performance in the experiment, which is covered in the aforementioned completion metrics, or were simple statements of fact (e.g., *"In order to connect two classes I drew a connecting line"*).

## 6 Discussion

**Participant Pool Composition** The data indicates no correlation (bivariate correlation $< \pm 0.1$) between the subjects' reported author familiarity, and their reported SUS scores, such as would have been the case if the subjects who knew the authors were biased. The high relative standard deviation for a priori knowledge level responses indicates that our subjects are rather diverse in their skill levels – i.e., they do not consist exclusively of the limited-experience class of users that we hope CoModIDE will ultimately support. As discussed below, this variation is in fact fortunate as it allows us to compare the performance of more or less experienced users.

**Metric Evaluation** Before we can determine if our results confirm H1 and H2 (replicated in Figure 5 from Section 1), we must first examine the correlations between our results and the control variables gathered in the *a priori* survey. In this context, we find it reasonable to use these thresholds for a correlation $|r|$:

Table 5: Free text comment fragments per category

| Code | Fragment # |
|---|---|
| Graph layout | 4 |
| Dragging & dropping | 6 |
| Feature requests | 5 |
| Bugs | 8 |
| Modeling problems | 5 |
| Value/preference statements | 9 |

H1. When using CoModIDE, a user takes less time to produce correct and reasonable output, than when using Protege.
H2. A user will find CoModIDE to have a higher SUS score than when using Protege alone.

Fig. 5: Our examined hypotheses, restated from Section 1.

0-0.19 very weak, 0.20-0.39 weak, 0.40-0.59 moderate, 0.60-0.79 strong, 0.80-1.00 very strong.

As shown in Table 2c, the metric *time taken* when using Protégé is negatively correlated with each CV. The *correctness* metric is positively correlated with each CV. This is unsurprising and reasonable; it indicates that familiarity with the ontology modeling, related concepts, and Protégé improves (shortens) time taken to complete a modeling task and improves the correctness of the output. However, for the metrics pertaining to CoModIDE, there are only very weak and three weak correlations with the CVs. We may construe this to mean that performance when using CoModIDE, with respect to our metrics, is largely agnostic to our control variables.

To confirm H1, we look at the metrics separately. *Time taken* is reported better for CoModIDE in both mean and median. When comparing the underlying data, we achieve $p \approx 0.025 < 0.05$. Next, in comparing the *correctness* metric from Table 2b, CoModIDE again outperforms Protégé in both mean and median. When comparing the underlying data, we achieve a statistical significance of $p \approx 0.009 < 0.01$. With these together, we reject the null hypothesis and confirm H1.

This is particularly interesting; given the above analysis of CV correlations where we see no (or very weak) correlations between prior ontology modeling familiarity and CoModIDE modeling results, and the confirmation of H1, that CoModIDE users perform better than Protégé users, we have a strong indicator that we have in fact achieved increased approachability.

When comparing the SUS score evaluations, we see that the usability of Protégé is strongly influenced by familiarity with ontology modeling and familiarity with Protégé itself. The magnitude of the correlation suggests that newcomers to Protege do not find it very usable. CoModIDE, on the other hand is weakly, negatively correlated along the CV. This suggests that switching to a graphical modeling paradigm may take some adjusting.

However, we still see that the SUS scores for CoModIDE have a greater mean, tighter $\sigma$, and greater median, achieving a very strong statistical significance $p \approx 0.0003 < 0.001$. Thus, we may reject the null hypothesis and confirm H2.

As such, by confirming H1 and H2, we may say that CoModIDE, via graphical ontology modeling, does indeed improve the approachability of knowledge graph development, especially for those not familiar with ontology modeling—with respect to our participant pool. However, we suspect that our results are

generalizable, due to the strength of the statistical significance (Table 3) and participant pool composition (Section 5.1).

**Free-text Responses** The fragments summarized in Table 5 paints a quite coherent picture of the subjects' perceived advantages and shortcomings of CoModIDE, as follows:

- *Graph layout:* The layout of the included MODL patterns, when dropped on the canvas, is too cramped and several classes or properties overlap, which reduces tooling usability.
- *Dragging and dropping:* Dragging classes was hit-and-miss; this often caused users to create new properties between classes, not move them.
- *Feature requests:* Pressing the "enter" key should accept and close the entity renaming window. Zooming is requested, and an auto-layout button.
- *Bugs:* Entity renaming is buggy when entities with similar names exist.
- *Modeling problems:* Self-links/loops cannot easily be modeled.
- *Value/preference statements:* Users really appreciate the graphical modeling paradigm offered, e.g., *"Mich easier to use the GUI to develop ontologies"*, *"Moreover, I find this system to be way more intuitive than Protégé"*, , *"comodide was intuitive to learn and use, despite never working with it before."*

We note that the there is a near-unanimous consensus among the subjects that graphical modeling is intuitive and helpful. When users are critical of the CoModIDE software, these criticisms are typically aimed at specific and quite shallow bugs or UI features that are lacking. The only consistent criticism of the modeling method itself relates to the difficulty in constructing self-links (i.e., properties that have the same class as domain and range).

## 7 Conclusion

To conclude, we have shown how the CoModIDE tool allows ontology engineers, irrespective of previous knowledge level, to develop ontologies more correctly and more quickly, than by using standard Protégé; that CoModIDE has a higher usability (SUS score) than standard Protégé; and that the CoModIDE issues that concern users primarily derive from shallow bugs as opposed to methodological or modeling issues. Taken together, this implies that the modular graphical ontology engineering paradigm is a viable way to improving the approachability of ontology engineering.

**Future Work** CoModIDE is under active development and is not yet feature-complete. Specifically, during the spring of 2020 we will implement the following features:

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

- 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.

In developing CoModIDE we have come across several trade-offs between usability and expressiveness, as discussed in Section 2. We intend to follow these threads, using CoModIDE as test bed, to study more precisely how the need for graphical representability affects the use of modeling constructs and/or ontology engineering methods. For instance, we initially assumed that a graphical modeling paradigm would help users verify the correctness of their designs; but the answers to our *a posteriori* survey questions on this matter proved inconclusive.

*Acknowledgement* Cogan Shimizu and Pascal Hitzler acknowledge partial support from the following financial assistance award 70NANB19H094 from U.S. Department of Commerce, National Institute of Standards and Technology and partial support from the National Science Foundation under Grant No. 1936677.

## References

1. Agarwal, R., Sinha, A.P.: Object-oriented modeling with uml: a study of developers' perceptions. Communications of the ACM **46**(9), 248–256 (2003)
2. Blomqvist, E., Sandkuhl, K.: Patterns in Ontology Engineering: Classification of Ontology Patterns. In: Proceedings of the 7th International Conference on Enterprise Information Systems. pp. 413–416 (2005)
3. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on Pattern-based Ontology Design. In: Gil, Y., Noy, N. (eds.) K-CAP '09: Proceedings of the Fifth International Conference on Knowledge Capture. pp. 41–48. ACM (2009)
4. 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, Studies on the Semantic Web, vol. 25, chap. 2, pp. 23–50. IOS Press (2016)
5. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with extreme design. In: Knowledge Engineering and Management by the Masses, pp. 120–134. Springer (2010)
6. Burnard, P.: A method of analysing interview transcripts in qualitative research. Nurse education today **11**(6), 461–466 (1991)
7. Cherfi, S.S.S., Akoka, J., Comyn-Wattiau, I.: Conceptual modeling quality-from eer to uml schemas evaluation. In: International Conference on Conceptual Modeling. pp. 414–428. Springer (2002)
8. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: The Semantic Web–ISWC 2005, pp. 262–276. Springer (2005)
9. Hadar, I., Soffer, P.: Variations in conceptual modeling: classification and ontological analysis. Journal of the Association for Information Systems **7**(8), 20 (2006)
10. Hammar, K.: Ontology Design Patterns in WebProtégé. In: Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Betlehem, USA, October 11, 2015. No. 1486 in CEUR-WS (2015)

11. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V.: Towards a simple but useful ontology design pattern representation language. In: Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., Hoekstra, R. (eds.) Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017. No. 2043 in CEUR Workshop Proceedings (2017)
12. Hitzler, P., Krisnadhi, A.: A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. CoRR **abs/1808.08433** (2018), http://arxiv.org/abs/1808.08433
13. Krogstie, J.: Evaluating uml using a generic quality framework. In: UML and the Unified Process, pp. 1–22. IGI Global (2003)
14. Lohmann, S., Link, V., Marbach, E., Negru, S.: Webvowl: Web-based visualization of ontologies. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 154–158. Springer (2014)
15. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing ontologies with vowl. Semantic Web **7**(4), 399–419 (2016)
16. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015)
17. Noy, N.F., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. Commun. ACM **62**(8), 36–43 (2019). https://doi.org/10.1145/3331166, https://doi.org/10.1145/3331166
18. Presutti, V., Gangemi, A., David, S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: D2.5.1: A Library of Ontology Design Patterns: Reusable Solutions for Collaborative Design of Networked Ontologies. Tech. rep., NeOn Project (2007)
19. Sarker, M.K., Krisnadhi, A.A., Hitzler, P.: OWLAx: A protégé 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 (2016)
20. Seaman, C.B.: Qualitative methods. In: Guide to advanced empirical software engineering, pp. 35–62. Springer (2008)
21. Shimizu, C., Hirt, Q., Hitzler, P.: MODL: A Modular Ontology Design Library. In: Proceedings of the 10th Workshop on Ontology Design and Patterns (WOP 2019) co-located with 18th International Semantic Web Conference (ISWC 2019). CEUR Workshop Proceedings, vol. 2459, pp. 47–58 (2019)
22. Shoval, P., Frumermann, I.: Oo and eer conceptual schemas: a comparison of user comprehension. Journal of Database Management (JDM) **5**(4), 28–38 (1994)
23. 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 (2018)