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`