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 [2]. 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 [1].

Utilizing a pattern representation language (PRL) 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 PRL 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, [3] 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[1] 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[2] ODP. Additionally, the ModifiedHazardousSituation[3] Design may contain the triple

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

which states that ModifiedHazardousSituation is derived from the HazardousSituation[4] ODP. Figure 2 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, is implemented for Protégé 5. At the time of this writing, plugin registration through the Protégéwiki[5] is ongoing. We provide a portal online[6] 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 2 and [3]. 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.[7] 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,

[1] http://ontologydesignpatterns.org/wiki/Submissions:MicroblogEntry
[2] http://ontologydesignpatterns.org/wiki/Submissions:NewsReportingEvent
[3] http://ontologydesignpatterns.org/wiki/Submissions:ModifiedHazardousSituation
[4] http://ontologydesignpatterns.org/wiki/Submissions:HazardousSituation
[5] https://protegewiki.stanford.edu/wiki/Main_Page
[6] http://dase.cs.wright.edu/content/oplatab
[7] http://ontologydesignpatterns.org/opla/

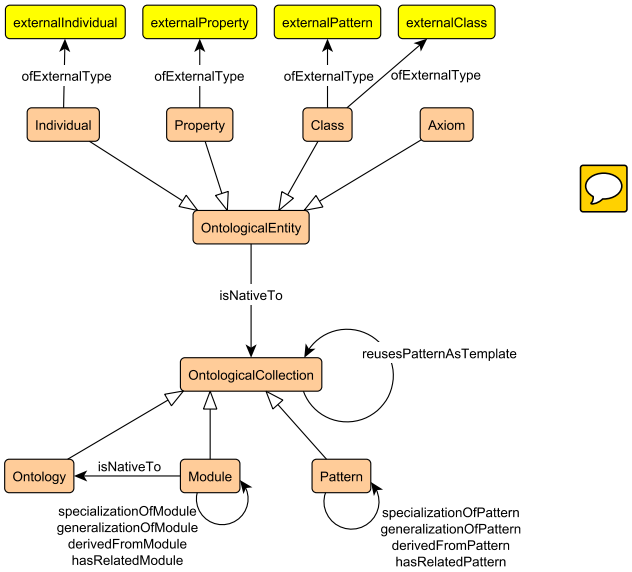

Fig. 1: The schema diagram representing the Ontology Design Pattter Representation Language (OPLa) [3].

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 options annotations so that it is easy to identify which entities have yet to be annotated.

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

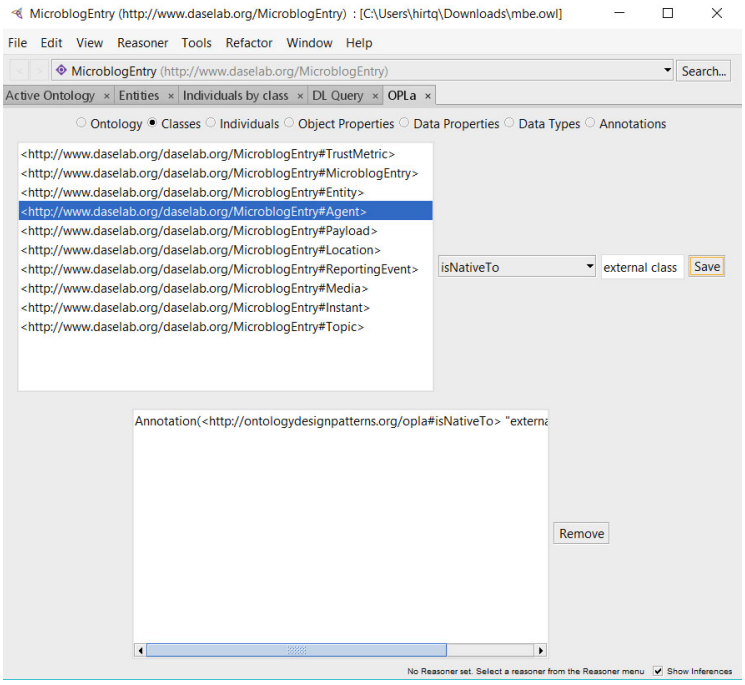

Fig. 2: A view of OPLaTab's interface. This view shows an example annotation be added to the loaded ontology.

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.[8]

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

References

1. 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.
2. 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.
3. 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.

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