Reasonable Ontology Templates: APIs for OWL

Efficient and Practical Ontology Design and Maintenance

Martin G. Skjæveland[1], Henrik Forssell[1], Johan W. Klüwer[2], Daniel Lupp[1], Evgenij Thorstensen[1], and Arild Waaler[1]

[1] Department of Informatics, University of Oslo
[2] DNV GL, Norway

Abstract. Reasonable Ontology Templates, OTTRs for short, are OWL ontology macros capable of representing ontology design patterns (ODPs) and closely integrating their use into ontology engineering. An OTTR is itself an OWL ontology or RDF graph, annotated with a special purpose OWL vocabulary. This allows OTTRs to be edited, debugged, published, identified, instantiated, combined, used as queries and bulk transformations, and maintained—all leveraging existing W3C standards, best practices and tools. We show how such templates can drive a technical framework and tools for a practical, efficient and transparent use of ontology design patterns in ontology design and instantiation.

1 Introduction

Ontology-based methods have matured to where they offer knowledge workers practical solutions for data management. In particular, tools that support W3C recommendations, such as reasoning tools for OWL ontologies, are sufficiently stable and efficient to allow wide-scale industrial use. However, from the perspective of product vendors and consultancy companies in the IT industry, ontologies are still viewed as a fringe technology. Hence ontology-based solutions are rarely proposed to enterprise customers, and support from the software industry is limited. One factor that impedes uptake is the high cost of establishing and maintaining a high-quality ontology. In part this is due to the scarcity of ontology experts, with availability in most cases below critical mass, and the lack of abstraction mechanisms and tool supported methods for applying ontology design patterns ODPs [2] in ontology engineering. Efficient tool support is imperative to industrial scale deployment of ontology-based methods.

The work reported on in this paper has the potential to remedy the situation; *Reasonable Ontology Templates* (OTTRs) [6, An extended version of the current paper] are simple, but powerful, templates or macros for ontologies, cf. [7], represented in OWL using a dedicated OWL vocabulary. An OTTR can be viewed as a *parameterised ontology* which can be nested, i.e., defined using other OTTRs, and *instantiated* by providing arguments to fit the parameters of the template. By recursively *expanding* an OTTR by replacing any contained OTTR with the pattern it represents, we obtain a (regular) OWL ontology. Using this feature, we can reason both on the OTTR specification and its expansion, and additionally leverage existing W3C languages and tools for different ontology engineering tasks—all driven by OTTRs. Specifically, the implicit mapping between an OTTR's parameters and its pattern may be exploited to *generate* various

format descriptions and transformation specifications, e.g., queries for extracting pattern instances and transformations between tabular input formats and OTTR pattern instances. The only additional tool support that is needed to make use of OTTRs are tools that can perform the relatively simple operation of template expansion and instantiation.

We believe OTTRs can provide a framework whereby a few ontology experts can serve a large number of domain experts and put these in a position to actively contribute to the development and maintenance of ontologies by clearly separating the design of an ontology and the bulk content of the ontology: The ontology expert designs and combines patterns represented as OTTRs to provide *user-facing* patterns on a level of abstraction suitable for the domain matter experts. From the user-facing OTTRs a simple input format is generated together with a transformation specification of the input format to ontology format. The task of the domain matter experts is then "only" to provide instance arguments to the input format.

2 Reasonable Ontology Templates

A *template* $\mathcal{T}$ is a knowledge base $\mathcal{O}_\mathcal{T}$ together with a list of parameters $(\mathsf{p}_1, \ldots, \mathsf{p}_n)$ of distinguished concept, role, or individual names from $\mathcal{O}_\mathcal{T}$. We write a template as $\mathcal{T}(\mathsf{p}_1, \ldots, \mathsf{p}_n) :: \mathcal{O}_\mathcal{T}$ and refer to the left side as the *head* and the right side as the *body*. For a list $(q_1, \ldots, q_n)$ of constants, concepts or role expressions called *arguments*, we call $\mathcal{T}(q_1, \ldots, q_n)$ a *template instance*. Intuitively, a template instance is a shorthand for representing a specific occurrence of a pattern. In addition to regular ontology axioms, a template body may contain template instances. The *expansion* of a template instance $\mathcal{T}(q_1, \ldots, q_n)$ is the ontology obtained by replacing each parameter occurrence of p_i in $\mathcal{O}_\mathcal{T}$ with the argument q_i, for $1 \leq i \leq n$, and applying the expansion recursively to any template instances in the body. Cyclic template definitions are not allowed.

Example 1.

$$\mathsf{SubSome}(\mathsf{class}, \mathsf{prop}, \mathsf{range}) :: \{\mathsf{class} \sqsubseteq \exists \mathsf{prop}.\mathsf{range}\} \quad (1)$$

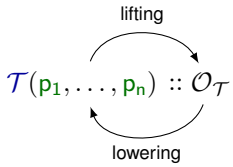
$$\mathsf{Disjoint}(\mathsf{class1}, \mathsf{class2}) :: \{\mathsf{class1} \sqcap \mathsf{class2} \sqsubseteq \bot\} \quad (2)$$

$$\mathsf{PartOf}(\mathsf{part}, \mathsf{whole}) :: \{\mathsf{SubSome}(\mathsf{part}, \mathsf{hasPart}, \mathsf{whole}), \mathsf{Disjoint}(\mathsf{part}, \mathsf{whole})\} \quad (3)$$

$$\mathsf{PartOf}(\mathsf{2CV}, \mathsf{SoftTop}) \implies \{\mathsf{2CV} \sqsubseteq \exists \mathsf{hasPart}.\mathsf{SoftTop}, \mathsf{SoftTop} \sqcap \mathsf{2CV} \sqsubseteq \bot\} \quad (4)$$

SubSome (1) is a template with three parameters: class, prop and range; and a single axiom body: $\{\mathsf{class} \sqsubseteq \exists \mathsf{prop}.\mathsf{range}\}$. The PartOf template (3) contains the SubSome and Disjoint (2) templates in its body, fixing the second argument of SubSome to the role name hasPart. An instance of this template, with its expansion, is seen in (4).

A core feature of OTTR templates is the ability to relate a simple tabular input format, represented by the parameter list of the template head, to a rich ontological structure in the template body, possibly specified via compositions of other templates. The fact that a template specifies both a tabular input format, an output ontology, and a mapping between the two formats may be exploited by leveraging the capabilities of existing W3C standards and implementations. In addition to specifying an ontology representing a prototypical ontology of the template



(the expanded template), a single template can specify different tabular and graph input format specifications using, e.g., XSD and ShEx, and transformations to and from (liftings and lowerings) the ontology output format using, e.g., XSL and SPARQL. This means that data can be captured in bulk using XML- or XSD-aware client tools, and efficiently processed using XSL and/or SPARQL processors, all of which are driven by specifications generated from a template.

Using this framework the ontology engineering task can be split in two more or less distinct responsibilities: one managed by the ontology engineer and the other by the domain matter expert. The main task of the ontology engineer is to design and maintain a library of interconnected templates capable of capturing the knowledge of the domain matter expert at the correct abstraction level and using a vocabulary and format recognisable by the user. The specificity needed for the ontology engineering task at hand is achieved by iteratively composing and combining basic and more complex templates, resulting in user-facing templates. From such templates, tabular input format specification and transformations may be generated from the template specification, presenting a simple tabular format for the user to fill in and the accompanying transformation specification for generating ontology data.

This process ensures uniformity and completeness of the captured domain knowledge: completeness, as the template specifies all the attributes that are necessary and variable; and uniformity, as the template instances are guaranteed to expand to the desired pattern. The correctness of templates may be secured by checking the prototype ontology resulting from expanding the template, as well as for each of the comprising templates in isolation. Additional syntactic constraints on the input data may be specified for the input formats and be used to check completeness of the input data.

Templates are adapted to the semantic web by serialising them using RDF with the OTTR vocabulary defined specifically for this task, and an implementation of the template mechanism that can perform the necessary expansion and substitution is available online; see http://www.ottr.xyz. This web service can read any OTTR template by providing its address to the web application in an IRI query parameter. An example template similar to Ex. 1 is published at its IRI http://draft.ottr.xyz/i18/partof and can be displayed in the online application at the following IRI: http://osl.ottr.xyz/info/?tmp=http://draft.ottr.xyz/i18/partof. This page shows how the vocabulary is used to specify parameters and arguments, and how these are passed on to containing template instances. Also available from this page are all the formats that can be generated from the template by the application. A library of OTTR templates is published at http://library.ottr.xyz.

We believe that a set of templates can act as an API for OWL ontology construction with its simple, but powerful abstraction mechanism based on the well-known concept of nested non-cyclic macros and syntactic substitutions. This allows complex ontology expressions to be compactly represented by a naturally compositional structure which supports a more efficient construction and maintenance of ontologies following "don't repeat yourself" (DRY) principles. Additionally, with OTTR templates ontology design patterns can be explicitly identified as such and clearly encapsulated. This improves provenance and supports interoperability between ontologies using the same or related templates without the requirement that the pattern must be discovered first. Since templates are formally defined as parameterised ontologies the semantics of the pattern can be verified using regular ontology reasoners. Furthermore, it makes the organisation of templates and the study of relations between them essentially an extension of the

same well-studied issues regarding ontologies, and familiar terminology and theoretical machinery can be reused. Finally, OTTRs can be compactly represented in RDF as OWL ontologies using the OTTR vocabulary. This allows us to leverage the complete stack of existing W3C languages and tools, such as ontology editors and reasoners.

3 Related and Future Work

There are many practical tools and languages for using template-like mechanisms, of which we highlight the following. The Ontology Pre-Processor Language (OPPL) [3], originally an ontology manipulation language for adding and removing ontology axioms, allows for expressing patterns as scripts which may be included directly in OWL ontologies. The M^2 mapping language [5] extends the OWL Manchester syntax with ontology pattern descriptions for translating spreadsheet data into ontologies. Taking a broader approach, Tawny-OWL [4] provides an environment for building OWL ontologies using Clojure, with all the advantages of using a programming language, including the use of macros.

The present proposal for templates has been developed in close interaction with industrial user communities, and we intend to apply it to various existing enterprise ontologies in the immediate future. This will serve to evaluate, verify and refine the concept, and will help us develop an efficient and reliable set of tools and web services. We believe that templates can be important for construction and use of open, validated modelling patterns, as is required for shared models, and for enabling ontology-based collaboration. In order to create templates that cover typical needs of industrial users, we will work with standardisation bodies and make these templates available through a public repository. This should lower the cost of translating existing data into ontologies, opening up the benefits of ontology-based methods to new users.

To support this work, methods for constructing, structuring and managing templates are necessary. To this end, we intend to further develop the prototype implementation and to develop tools for developing and applying OTTR templates in ontology engineering. We also intend to continue the initial efforts on describing the logical properties of templates [1].

References

1. H. Forssell, D. P. Lupp, M. G. Skjæveland, and E. Thorstensen. Reasonable Macros for Ontology Construction and Maintenance. In *DL Workshop*, 2017.
2. P. Hitzler et al., editors. *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, volume 025. IOS Press, Amsterdam, 2016.
3. L. Iannone, A. L. Rector, and R. Stevens. Embedding Knowledge Patterns into OWL. In *ESWC*, pages 218–232, 2009.
4. P. Lord. The Semantic Web takes Wing: Programming Ontologies with Tawny-OWL. In *OWLED*, 2013.
5. M. J. O'Connor, C. Halaschek-Wiener, and M. A. Musen. M2: A Language for Mapping Spreadsheets to OWL. In *OWLED*, 2010.
6. M. G. Skjæveland, H. Forssell, J. W. Klüwer, D. P. Lupp, E. Thorstensen, and A. Waaler. Pattern-Based Ontology Design and Instantiation with Reasonable Ontology Templates. Accepted for the Workshop on Ontology Design and Patterns (WOP2017), 2017.
7. D. Vrandečić. Explicit knowledge engineering patterns with macros. In *Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005*, 2005.