Web Semantics: Science, Services and Agents on the World Wide Web 40 (2016) 52–64

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Adapting ontologies to best-practice artifacts using transformation patterns: Method, implementation and use cases

Vojtěch Svátek*, Marek Dudáš, Ondřej Zamazal

Department of Information and Knowledge Engineering, University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic

ARTICLE INFO

Article history:
Received 2 November 2015
Received in revised form
16 April 2016
Accepted 25 July 2016
Available online 16 August 2016

Keywords:
Ontology
Transformation
Pattern
Role
E-commerce

ABSTRACT

Reengineering an existing ontology to get it aligned with best practices, represented as design patterns or core ontologies, can be challenging. We demonstrate how the versatile PatOMat framework for pattern-based ontology transformation, together with the GUIPOT Protégé plugin as its front-end, can be used to fulfill this task. Two different use cases are presented. One consists in introducing role-based modeling, mediated by the AgentRole content pattern, into a legacy ontology; it has been applied on the complete OntoFarm collection, containing 16 heterogeneous ontologies on 'conference organization'. The other consists in more lightweight adaptation of legacy ontologies to multiple aspects of style of a core domain ontology; it has been applied on six ontologies that have been converted to the format of GoodRelations, a core ontology for e-commerce. While the former study was carried out by an experienced knowledge engineer, who analyzed the influence of ontology expressiveness and other formal features on the efficiency of transformation, the latter study involved 13 students with limited training, thus mapping, to a large degree, the role of human factor in the transformation.

1. Introduction

The establishment of the OWL language [1] as de facto standard for ontologies on the web is being only gradually followed by the establishment of *best practices* for creating content in this language. Meanwhile, thousands of OWL ontologies have been built for various domains. Preserving valuable knowledge contained in these ontologies while taking into account the best practices that hold either generally or within a specific domain is an important task.

While best practices can often have the form of conventions applied along the whole ontological model, in this paper we are interested in best practices incarnated in compact artifacts that are themselves expressed in the OWL language: either as fully-blown *ontologies* (foundational ones or core ones within a domain) or as smaller, widely usable ontological modules: *ontology content patterns* [2], further just CP for brevity. In both cases we can assume that this *best-practice artifact* (BPA) is to be integrated into the *legacy ontology* (LO), typically as its root portion. However, merely grafting the LO upon the BPA as it is may be unfeasible, since they may differ in terms of their modeling style; LO then has to be transformed to the style of the BPA. Examples of modeling style differences include:

- The use of a plain binary object property (e.g., *boughtFrom*) vs. its reification allowing to involve further arguments (e.g., a *Purchase* class).
- The modeling of a general concept as a class, e.g., *DomesticCat* with physical animals as instances, vs. meta-modeling it as an individual, e.g., individual *DomesticCat* as instance of class *BiologicalSpecies* (note that *BiologicalSpecies* could not be conveniently part of the model in the first modeling style, since it would be a second-order class inexpressible in OWL). A specific variant of this modeling dichotomy is that of choosing between expressing a concept taxonomy as OWL hierarchy or as a SKOS[1] concept scheme.
- The use of an object property (e.g., *country* valued by DBpedia[2] URIs) vs. data property (valued by codes such as "US", "UK" or "DE").

In our previous work on the *PatOMat* project[3] we already addressed the general need for style transformation in ontological engineering. A general *framework*, a simple transformation

* Corresponding author.
E-mail addresses: svatek@vse.cz (V. Svátek), marek.dudas@vse.cz (M. Dudáš), ondrej.zamazal@vse.cz (O. Zamazal).

http://dx.doi.org/10.1016/j.websem.2016.07.002
1570-8268/

[1] http://www.w3.org/2004/02/skos/.
[2] http://dbpedia.org.
[3] http://patomat.vse.cz.

pattern *language*, a set of RESTful *services* and various graphical user-oriented *tools* have been developed. They have also been thoroughly exemplified for two scenarios associated with frequent tasks of ontological engineering on the semantic web:

1. *ontology matching*: if the two ontologies to be matched differ in their modeling style, we can attempt to transform the modeling style of the one so as to make automated matching to the other easier [3];
2. *ontology language profiling*: if an ontology is intractable for a reasoner that only operates over a certain explicit or implicit OWL sub-language (profile), we can change the style of the ontology in terms of replacing the problematic constructs with tractable ones [4].

To the difference of these two transformation scenarios, in which the ontology is to comply with some structure that remains external, the third one discussed here is that of allowing for smooth *inclusion of a BPA* (content pattern or ontology) as direct part of the current ontology, as outlined above. In our very early research published in a workshop paper [5], we demonstrated the possibility of such a transformation on a tiny example comprising a single ontology (dealing with conference organization) wrt. a single content pattern, AgentRole (making role-based modeling explicit in OWL ontologies). The current paper extends the previous one [5] along numerous axes:

- A new *graphical tools* meanwhile developed, the GUIPOT Protégé plugin for interactive transformation (see end of next section), is exploited.
- Adaptation of ontologies to AgentRole is studied for two different kinds of inputs: class structures as in the previous paper [5], but also *property structures*.
- While the previous paper [5] merely discussed different transformation options for adaptation of an ontology to AgentRole, we now present *empirical results* achieved on sixteen ontologies from the OntoFarm collection, dealing with conference organization.
- Also the actual options discussed have been radically changed. The previous paper [5] reflected the effort to closely mimic the five modeling approaches used in the W3C pattern for 'classes as property values' [6]. We however later found the analogy with this pattern as too shallow to warrant such a close alignment, and rather proposed our own, more straightforward, systematization.
- Aside the 'role-modeling' use case, another use case is added, which addresses the integration of a whole *core ontology for e-commerce*, GoodRelations, into six independently-built ontologies mostly describing customer products.[4] This new use case description encompasses a usability study with 13 subjects.

The rest of the paper is structured as follows. Section 2 briefly reviews the PatOMat framework, transformation language and processing services. Section 3 summarizes the ontology matching scenario of PatOMat application from its seminal paper [3] as starting point, and outlines the new scenario of embedding a BPA into a legacy ontology. Section 4 presents the first use case, where the AgentRole content pattern is included as BPA: the AgentRole pattern itself, the legacy ontologies (OntoFarm collection), transformation alternatives, and empirical results. In Section 5 the second use case is presented in a similar structure, although slight alteration of the section structure has been incurred by the different nature of the use case; the BPA is now the GoodRelations ontology and the LOs are six product ontologies. The rest of the paper summarizes and compares both use cases (Section 6), surveys some related work (Section 7), and provides conclusions and future prospects.

[4] An early version of this use case, with one ontology only and much less detailed evaluation, has been described in a recent conference paper [7].

2. *PatOMat* principles and implementation

We only introduce the *PatOMat* transformation framework very briefly. The conference paper [3] provides more details about its initial principles and motivations, and at the project website there is a fully-fledged tutorial for the current version.[5]

The central notion in PatOMat is that of *transformation pattern* (TP). A TP contains two *ontology patterns* (the source OP and the target OP) and the description of transformation between them, called pattern transformation (PT). The representation of OPs is based on the OWL 2 DL profile. However, while an OWL ontology only refers to particular entities, e.g., to class *Person*, in the patterns we also use *placeholders*. Entities are specified (i.e. placeholders are instantiated) at the time of instantiation of a pattern. An OP consists of *entity declarations* (for placeholders as well as specific entities), *axioms*, and *naming detection patterns* (NDPs); the last captures the naming aspect of the OP important for its detection. A PT consists of a set of *transformation links* and a set of *naming transformation patterns* (NTPs). Transformation links are either *logical equivalence relationships* or *extralogical relationships* holding between two entities of different types. NTPs serve for generating new names for old or newly created entities. Naming patterns refer to *passive naming operations* such as detection of a head noun for a noun phrase, as well to *active naming operations* such as derivation of verb form of a noun.

The framework prototype implementation is available either as a *Java library* or as three *core services*, all accessible via the web interface at http://owl.vse.cz:8080/patomat/. The Java library has been directly integrated into three tools: our GUIPOT tool and two third-party ones — the ORE tool[6] for ontology repair [8] and a version of the XDTools framework[7] supporting the eXtreme Design methodology of ontology development [9]. The whole transformation is divided into three steps, which correspond to three core services:

- The *OntologyPatternDetection* service takes the transformation pattern and a particular original ontology on input, and returns the binding of entity placeholders on output, in XML. The structural/logical aspect is captured in the structure of a SPARQL query generated automatically by parsing axioms (in Manchester syntax [10]) from the source OP and transforming them into the Turtle syntax of the graph pattern.[8] Further, the naming aspect is dealt with based on the NDP.
- The *InstructionGenerator* service takes the particular binding of placeholders and the transformation pattern on input, and returns particular transformation instructions on output, also in XML. Transformation instructions are generated according to the transformation pattern and the pattern instance.
- The *OntologyTransformation* service takes the particular transformation instructions and the particular original ontology on input, and returns the transformed ontology on output.

The third service is partly implemented over the OWL-API.[9] We use OWL-API for the operations on *axioms*, for *adding entities*, for *re/naming* entities according to naming transformation patterns and for adding *OWL annotations*.

The transformation framework is now also accessible via a graphical user interface called GUIPOT[10] [11], implemented as

[5] http://owl.vse.cz:8080/patomat/tutorial/.

[6] http://ore.aksw.org/.

[7] http://stlab.istc.cnr.it/stlab/XDTools.

[8] Some specific axioms (e.g. annotations) are also transformed into the SPARQL syntax using a combination of hard-coded rewriting and Manchester syntax parsing.

[9] http://owlapi.sourceforge.net/.

[10] For "Graphical User Interface for Pattern-based Ontology Transformation".