Semi-automated Structural Adaptation of Advanced E-Commerce Ontologies

Marek Dudáš[1], Vojtěch Svátek[1], László Török[2],
Ondřej Zamazal[1], Bene Rodriguez-Castro[2], and Martin Hepp[2]

[1] University of Economics Prague, Nám. W. Churchilla 4, 13067 Praha 3, Czech Rep.
{xdudm12|svatek|ondrej.zamazal}@vse.cz
[2] Univ.der Bundeswehr Munich, W.-Heisenberg-Weg 39, 85579 Neubiberg, Germany
{laszlo.toeroek|bene.rodriguez|martin.hepp}@unibw.de

Abstract. Most ontologies used in e-commerce are nowadays taxonomies with simple structure and loose semantics. One exception is the OPDM collection of ontologies, which express rich information about product categories and their parameters for a number of domains. Yet, having been created by different designers and with specific bias, such ontologies could still benefit from semi-automatic post-processing. We demonstrate how the versatile *PatOMat* framework for pattern-based ontology transformation can be exploited for suppressing incoherence within the collection and for adapting the ontologies for an unforeseen purpose.

Key words: ontology, e-commerce, GoodRelations, transformation, ontology pattern, linked data

1 Introduction

The idea that well-designed, structurally rich ontologies would allow to partially automate e-commerce operations has been around for years [1]. Nevertheless, even nowadays, most ontologies exploited in this field are plain taxonomies with imprecise semantics. Proposals for sophisticated modeling remain at the level of academic prototypes, or, at most, are used in closed B2B settings [5].

The *GoodRelations* (GR) ontology [3] has been conceived, as an attempt to balance expressiveness and practical usability, with size comparable to popular linked data vocabularies[1], OWL ontology language[2] expressivity and stress on favorable learning curve thanks to a cookbook with a number of recipes.[3] As 'vertical' extensions to GR, ontologies for specific product/service categories then started to be developed, most recently within the *Ontology-Based Product Data Management* (OPDM) project.[4] This family of ontologies already enjoyed

[1] http://lov.okfn.org/dataset/lov/
[2] http://www.w3.org/TR/owl2-overview/
[3] http://wiki.goodrelations-vocabulary.org/Cookbook
[4] http://www.opdm-project.org/

industrial adoption, such as the car sales ontology used by a major automotive manufacturer.[5]

In this paper we focus on two aspects of such product ontologies for which further enhancement is possible. First, the rapid pace of creation of the ontologies and involvement of multiple designers in parallel occasionally leads to *incoherence* in modeling patterns and naming conventions, both within a single ontology and across a set of them. Second, some of their features are compromises between the best practices for publishing linked data [2] and somewhat different requirements imposed by the e-commerce and web engineering worlds, given they are to be used in direct integration with web-based product catalogs. Therefore they need to be adapted in order to be used in a 'canonical' linked data setting.

As either kind of structural adaptation potentially involves a wide scope of restructuring and renaming operations, it can benefit from application of a user-friendly ontology transformation framework. Such framework has been recently developed under the name of *PatOMat* [6, 8]. In the rest of the paper we first describe the material, i.e., the GoodRelations ontology and the product ontologies based on it (Section 2). Then we present the incoherence problems of both types discovered in the product ontologies (Section 3). Next, the principles of the PatOMat framework and its user interfaces are briefly reviewed (Section 4), and its application on the OPDM ontologies is described (Section 5). Finally, the paper is wrapped up (Section 6).

2 GoodRelations and Product Ontologies

GoodRelations (further GR) is a generic ontology for e-commerce, which offers conceptual elements to capture facts that are most relevant for exchanging arbitrary goods and services. The core model revolves around the abstraction that an *agent* offers to transfer certain *rights* related to a *product* or *service* [3]. This model is independent of a particular e-commerce domain, since the agent can be any commercial entity making the offer, and rights transferring can range from simple *sale* to *rental* or *leasing*. GR includes generic conceptual elements for products and services and their properties (including prices, delivery or warranty conditions etc.), but no domain-specific product classes or taxonomies.

A premier use case for GR is adding semantic annotation to of e-commerce *web sites*. Aside the website-level application, there are also domain-specific extensions of GR that can be used within e-commerce *business information systems* as a common data schema that all software services support. Product data available in many systems is often unstructured or incomplete. As sophisticated automated business processes require precise, highly structured data, they are likely to benefit from ontologies capturing data about products from particular domains. *OPDM ontologies*, designed to fulfil this need, extend a subset of GR: domain-specific *product classes* are subclasses of `gr:ProductOrService`, *product*

[5] `http://www.w3.org/2001/sw/sweo/public/UseCases/Volkswagen/`

properties are subproperties of `gr:quantitativeProductOrServiceProperty` or its 'quantitative' or 'datatype' counterparts,[6] and a few *generic properties* such as color, dimension or weight are directly reused from the GR ontology. The ontologies are self-contained, and capture the most frequently occurring properties of each particular product type.

3 Incoherence Problems in OPDM ontologies

3.1 Incoherence Types Considered

When an OWL ontology is being developed, there is often more than one option how to model a specific concept or structure, due to high expressiveness of the language. Modeling incoherence may thus arise when such modeling options differ for concepts/structure of similar nature. The fact that OPDM ontologies are all grafted upon the GR ontology somewhat alleviates this problem. Nevertheless, there is still space for incoherence; both at *structural* level, e.g., using a datatype property instead of object property, or at the level of *naming conventions*, such as arbitrarily switching between synonymous lexical constructs.

Another way of incoherence classification is according to the situation in which a particular part of an ontology is considered 'incoherent'. Due to the large number of OPDM ontologies and involvement of multiple designers, *intrinsic incoherence* may easily occur, which is a term we suggest for unintentional heterogeneous modeling occurring either within a single ontology or within a collection of ontologies typically presented together (such as the OPDM collection). On the other hand, if the ontologies are to be used outside the original context, it is likely that one will run into what we call *export-based extrinsic incoherence*. Finally, we could also consider *import-based extrinsic incoherence*, which occurs when legacy ontologies have to be adapted to a 'canonical' modeling style (here, the style pre-supposed by GR).[7] In the rest of this section we discuss examples[8] of different types of incoherence in the context of OPDM ontologies.[9]

3.2 Intrinsinc Incoherence

Intrinsic structural incoherence. One example of intrinsic structural incoherence is related to modeling the support of various media data types (e.g., GIF, JPEG, AVI etc.) available in an electronic device. There are several ontologies in the

[6] We omit their full names for typographic reasons – excessive length.

[7] Pre-cursor work on resolving import-based extrinsic incoherence (though not labeled by this term) at a generic level – with 'canonical' modeling defined by ontology content design patterns – is described in [9].

[8] A longer version of this article with more examples is available at `http://nb.vse.cz/~svabo/patomat/tp/opdm/ecweb13su.pdf`.

[9] In all examples, local entities from the individual OPDM ontologies are without prefix, while the GR ontology entities are presented with their usual `gr` prefix.

OPDM project that cover the described concept (ontologies of computers, cameras, bluray players, portable media players etc.), and as the OPDM ontologies are not modular and are designed to be used independently, the same concept has been designed separately in each ontology. In most of the ontologies there is a class `MediaFormat`, with instances JPEG, GIF, AVI etc., as well as an object property `playbackFormat`, which has the class `MediaType` as its range. In one of the ontologies, however, a different approach is used: there is a boolean data property for each of the media data types. So, for example, the fact that a hypothetical portable media player supports AVI would be expressed as `player playbackFormat AVI` in the former case and as `player AVI true` in the latter. We will refer to this incoherence pattern as to 'boolean vs. instance'.

3.3 Extrinsic Structural Incoherence

An example of extrinsic structural incoherence comes from considerations of using OPDM ontologies in an 'orthodox' linked data environment. A very relevant opportunity for advanced product ontologies is, for example, their use by an application for *public contracts* management. The *Public Contracts Ontology*[10] designed within the EU LOD2 project, as well as the processing tools that provision RDF data according to this ontology [4], strictly adhere to the linked data principles, which suggest using object properties rather than data properties.[11] Each OPDM ontology is meant to be used independently, outside the linked data cloud, and barriers for their usage by practitioners (unfamiliar with semantic web technologies) is lowered as much as possible, hence most of the properties are datatype properties. This makes them easy to populate with 'instances' in the form of literals; however, in the linked data environment, the benefits of *interlinking* could not be exploited.

4 PatOMat Framework for Ontology Transformation

The central notion in the *PatOMat* framework[12] 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, except that *placeholders* are allowed in addition to concrete OWL entities. An OP consists of *entity declarations* (of placeholders and/or concrete entities), *axioms* and *naming detection patterns* (NDP); the last capture the naming aspect of the OP, which is important for its detection. A PT consists of a set of *transformation links*

[10] `http://code.google.com/p/public-contracts-ontology/`

[11] The use of object properties allows for explicitly referring to resources (ontological instances) from external datasets.

[12] [8] provides details about the initial version of the framework, [6] about the user-oriented tools, and at `http://owl.vse.cz:8080/tutorial/` there is a fully-fledged tutorial for the current version.

and a set of *naming transformation patterns* (NTP). Transformation links are either *logical equivalence relationships* or *extralogical relationships* (holding between two entities of different type, thus also called 'heterogeneous equivalence'). Naming transformation patterns serve for generating names for target entities. Naming patterns range from *passive naming operations*, such as detection of a head noun for a noun phrase, to *active naming operations*, such as derivation of verb form of a noun. Syntactically, the patterns are expressed according to an XML schema[13] However, the patterns needn't be edited manually, as a graphical editor is available for their authoring.[14] The framework prototype implementation is available either as a *Java library* or as three *RESTful services*.[15] The Java library is used by the GUIPOT tool[16] and other transformation GUIs.

The framework has already been used in multiple use cases, such as:

- Adaptation of the style of an ontology to another one to which it is to be *matched* [8]
- Adaptation of the style of a legacy ontology to a best-practice content pattern being *imported* into it [9]
- *Repair* use cases, including downgrading of an ontology to a less expressive dialect of OWL [7] or entity naming canonicalization [10].

5 Pattern-Based Transformation of OPDM Ontologies

5.1 Selected Transformation in Depth

Transformation patterns were designed for all of the previously described incoherence cases. One of them[17] is presented in this section in detail.

Transformation for 'boolean vs. instance' This incoherence case requires a transformation of boolean data properties to instances of a new `MediaFormat` class, while also adding a property such as `playbackFormat`, whose range is this class. It can be achieved using the transformation pattern in Fig. 1.[18] The source pattern thereof fits all boolean (as specified in the first axiom) subproperties of `gr:datatypeProductOrServiceProperty` (specified in the second axiom), of which those representing media types have to be selected (currently, manually). The rest of the transformation is performed automatically according to the target ontology pattern and the pattern transformation parts of the transformation pattern, as shown below. The role of the two axioms concerning annotations (labels and comments) is to transfer them to the target transformed ontology. The

[13] `http://nb.vse.cz/~svabo/patomat/tp/tp-schema.xsd`
[14] `http://owl.vse.cz:8080/tpe/`
[15] All accessible via the web interface at `http://owl.vse.cz:8080/`.
[16] `http://owl.vse.cz:8080/GUIPOT/`
[17] All patterns are in full extent at `http://nb.vse.cz/~svabo/patomat/tp/opdm/`.
[18] The `|` symbols are not part of the code: they only mark elements that are referred to in the explanatory text.

purpose of the last axiom in the source pattern is to keep the information about the domain of the transformed data property (i.e., some product class) in the placeholder `?pc`. It will be used to set the domain of the newly created object property `playbackFormat`, whose range will be the newly created `MediaFormat` class; its instances arise from the transformed data properties. All the datatype properties `?m` selected in the previous step are transformed into instances `?OP2_m` of class `MediaFormat`, which is created as a new entity. The selected properties `?m` are removed from the ontology and replaced with instances `?OP2_m`. Axioms describing `?m` are also removed except labels and comments (as mentioned above), which are connected to the newly created instances `?OP2_m`. The `playbackFormat` object property (represented by placeholder `?OP2_p`) is created, its domain set to `?OP2_pc` – the domain of the transformed data property – and its range to `?OP2_C` – the newly created `MediaClass`.

5.2 Transformation Pattern Application Using GUIPOT

As one of the user-oriented add-ons [6] to the PatOMat framework we developed the Graphical User Interface for Pattern-based Ontology Transformation (GUIPOT), as means for comfortable application of transformation patterns. GUIPOT is a plugin for Protégé.

After loading a transformation pattern it displays a list of pattern instances of the source OP detected in the given ontology: see the upper-center of the screen in Fig. 2, for an application of the 'boolean vs. instance' pattern. By selecting one or more instances, the detected entities are highlighted in the hierarchical view of the ontology in the left part of the plugin window. The right part of the window shows the ontology after the transformation with entities that were affected (changed or added) by the transformation marked with red arrows.

6 Conclusions and Future Work

The presented research leverages on several years of research on both e-commerce ontology principles and ontology transformation techniques. It aims to provide collections of product ontologies with better internal coherence as well as external reusability, in particular, in the linked data world.

In the future, we also plan to address *import-based extrinsic incoherence*, i.e., adaptation of various legacy ontologies to GR-based modeling. Presumably, the design of ontologies for *novel domains* of products and services (such as the building industry, which plays an important role in public procurement) will also bring into light novel kinds of pattern, thus leading to enrichment of the transformation pattern library. The proliferation of specific transformation patterns will also need to be backed by a user-friendly *pattern portal* integrated with the mainstream ontology pattern portal.[19]

[19] `http://ontologydesignpatterns.org`

```
<op1>
    <entity_declarations>
     <placeholder type="DatatypeProperty">?m</placeholder>
     <placeholder type="Literal">?a1</placeholder>
     <placeholder type="Literal">?a2</placeholder>
     <placeholder type="Class">?pc</placeholder>
     <entity type="Class">&xsd;boolean</entity>
     <entity type="DatatypeProperty">
          &gr;datatypeProductOrServiceProperty</entity>
     <entity type="AnnotationProperty">&rdfs;label</entity>
     <entity type="AnnotationProperty">&rdfs;comment</entity>
    </entity_declarations>
    <axioms>
|    <axiom>DataProperty: ?m Range: boolean</axiom>
|    <axiom>DataProperty: ?m SubPropertyOf:
          datatypeProductOrServiceProperty</axiom>
|    <axiom>DataProperty: ?m Annotations: label ?a1</axiom>
|    <axiom>DataProperty: ?m Annotations: comment ?a2</axiom>
|    <axiom>DataProperty: ?m Domain: ?pc</axiom>
    </axioms>
 </op1>
<op2>
    <entity_declarations>
     <placeholder type="Individual">?OP2_m</placeholder>
     <placeholder type="Class">?OP2_C</placeholder>
     <placeholder type="ObjectProperty">?OP2_p</placeholder>
     <placeholder type="Literal">?OP2_a1</placeholder>
     <placeholder type="Literal">?OP2_a2</placeholder>
     <placeholder type="Class">?OP2_pc</placeholder>
     <entity type="ObjectProperty">
          &gr;qualitativeProductOrServiceProperty</entity>
    </entity_declarations>
    <axioms>
|    <axiom>Individual: ?OP2_m Types: ?OP2_C</axiom>
     <axiom>ObjectProperty: ?OP2_p SubPropertyOf:
          qualitativeProductOrServiceProperty</axiom>
     <axiom>Individual: ?OP2_m Annotations: label ?OP2_a1</axiom>
     <axiom>Individual: ?OP2_m Annotations: comment ?OP2_a2</axiom>
     <axiom>ObjectProperty: ?OP2_p Domain: ?OP2_pc</axiom>
     <axiom>ObjectProperty: ?OP2_p Range: ?OP2_C</axiom>
    </axioms>
 </op2>
 <pt>
    <eqHet op1="?m" op2="?OP2_m"/> <eq op1="?a1" op2="?OP2_a1" />
    <eq op1="?a2" op2="?OP2_a2" /> <eq op1="?pc" op2="?OP2_pc" />
    <ntp entity="?OP2_C">MediaFormat</ntp>
    <ntp entity="?OP2_p">playbackFormat</ntp>
    <ntp entity="?OP2_a1">"+?a1+"</ntp>
    <ntp entity="?OP2_a2">"+?a2+"</ntp>
 </pt>
```

Fig. 1. Pattern for transforming (media type) boolean properties to instances

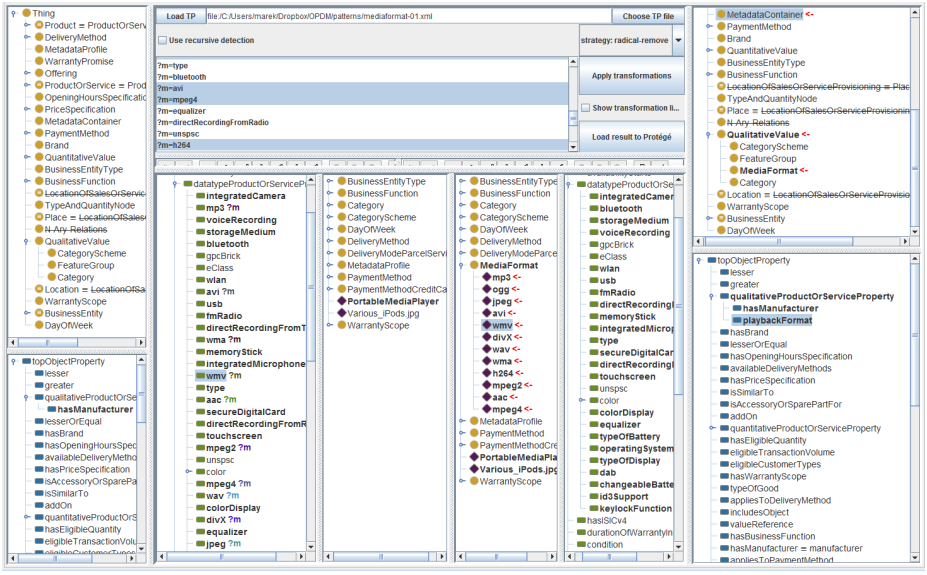

Fig. 2. Processing of 'boolean vs. instance' pattern by GUIPOT

References

1. Ding Y., Fensel D., Klein M., Omelayenko B., Schulten E.: The role of ontologies in e-Commerce. In: Handbook on Ontologies, Springer, 2004.
2. Heath T., Bizer C.: Linked Data: Evolving the Web into a Global Data Space (1st edition). Morgan & Claypool, 2011.
3. Hepp M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: Proc. EKAW2008, 2008, Springer LNCS, Vol.5268, 332-347.
4. Klímek J., Knap T., Mynarz J., Nečaský M., Svátek V.: Framework for Creating Linked Data in the Domain of Public Sector Contracts. Deliverable 9a.1.1 of the EU FP7 LOD2 project. Online, `http://lod2.eu/Deliverable/D9a.1.1.html`
5. Lee T., Lee I.-h., Lee S., Lee S.-g., Kim D., Chun J., Lee H., Shim J.: Building an operational product ontology system. *El. Commerce Res. and App.* 5, 16-28 (2006).
6. Šváb-Zamazal O., Dudáš M., Svátek V.: User-Friedly Pattern-Based Transformation of OWL Ontologies. In: Proc. EKAW 2012, Galway, Springer-Verlag, LNCS 7603.
7. Šváb-Zamazal O., Schlicht A., Stuckenschmidt H., Svátek V.: Constructs Replacing and Complexity Downgrading via a Generic OWL Ontology Transformation Framework. In: Sofsem 2013, Springer, LNCS 7741.
8. Šváb-Zamazal O., Svátek V., Iannone L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: EKAW-2010, Lisbon, Portugal, 2010.
9. Svátek V., Šváb-Zamazal O., Vacura M.: Adapting Ontologies to Content Patterns using Transformation Patterns. In: Workshop on Ontology Patterns (WOP 2010) collocated with ISWC'10, Shanghai, China, November 8, 2010. Online `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-671/`.
10. Zamazal O., Bühmann L., Svátek V.: Checking and Repairing Ontological Naming Patterns using ORE and PatOMat. In: WoDOOM'13, workshop at ESWC 2013.