

Towards a Catalog of *OWL-based* Ontology Design Patterns

Aldo Gangemi¹, Asunción Gómez-Pérez², Valentina Presutti¹,
M. Carmen Suárez-Figueroa²

¹ Laboratory for Applied Ontology, ISTC-CNR, Rome, Italy
{aldo.gangemi, valentina.presutti}@istc.cnr.it

² Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo s/n.
28660 Boadilla del Monte. Madrid. Spain
{asun, mcsuarez}@fi.upm.es

Abstract. Taking in consideration the works on software patterns in Software Engineering and with the aim of introducing ontology design patterns in the Ontology Engineering field for helping software engineers and ontology practitioners to develop ontologies faster, cheaper and better, in this paper we propose a general template for describing ontology design patterns (based on the well-accepted format for software engineering patterns). We also provide a first version of a catalog of ontology design patterns (exemplified with an OWL DL¹ implementation) organized into three different categories: logical patterns, architectural patterns and content patterns.

Keywords: Ontology design patterns, OWL DL, catalog.

1 Introduction

The term “*pattern*” [6] appears in English in the 14th century and derives from Middle Latin “*patronus*” (meaning “*patron*”, and, metonymically, “*exemplar*”, that is something proposed for imitation). In the 1970’s, the architect and mathematician Christopher Alexander introduced the term “*design pattern*” [1] for shared guidelines that help solve design problems.

In Software Engineering, a *design pattern* is defined as a simple and elegant solution to specific problems in object-oriented software design [4, 5]. A design pattern provides a common vocabulary for designers to communicate, document, and explore design alternatives. It is a generic description of how to solve a modelling problem that can be used in many different situations. A *design pattern* reflects the experience, knowledge and insights of developers who have successfully used these patterns in their own work, and provides a ready-made solution that can be adapted to different problems if necessary.

The design patterns in Software Engineering are described using a commonly and accepted format [5] that includes, for instance, the design problem, the context, the

¹ <http://www.w3.org/TR/owl-features/>

proposed solution, and examples. Moreover, the design patterns are organized in pattern catalogs [3], which are collections of related patterns subdivided into different categories.

Recently, in the Ontological Engineering field, ontology design patterns have emerged as a way for helping ontology practitioners to model OWL ontologies [6, 7]. Such works present modelling solutions, implemented in OWL, to some well-known ontology modelling problems. However, these ontology design patterns are difficult to understand for those practitioners (eg. software engineers) that are not familiar with OWL syntax, but are used to model using other software engineering techniques. So, a graphical notation (eg. UML based) would help software engineers and ontology practitioners in the task of understanding the patterns, and would increment the ontology design patterns reuse in the ontology building process. Another important limitation of the works done up to now is that there is no consensus on a format or scheme to represent ontology design patterns, as in Software Engineering. The Ontological Engineering field also miss an ontology design pattern catalog with the goal of helping software engineering and ontology practitioners in the activity of modelling their ontologies.

Taking into account that both Ontology Engineering and Software Engineering produce domain models, some principles about software design patterns [4] can be reused, adapted and extended for the construction of ontology design patterns. Thus, *ontology design patterns* are modelling solutions to well-known design problems in Ontology Engineering, they are based on best practices for solving modelling problems and they make ontology design easier for software engineers and ontology practitioners.

For these reasons, in this paper, we present an agreed format or scheme for describing ontology design patterns (based on the well-accepted format for software engineering patterns) produced by partners working in the NeOn project². This template includes UML diagrams and OWL abstract syntax code for the solution proposed in the ontology design pattern, in order to facilitate the understanding and reuse of ontology design patterns by non OWL experts. We also provide a catalog of ontology design patterns (for OWL DL³) organized into three different categories: logical patterns, architectural patterns and content patterns.

This paper is organized as follows: section two describes the state of the art of ontology design patterns. Section three presents the catalog of ontology design patterns (for OWL DL). Sections four, five, and six deal respectively with the three different categories of patterns: logical, architectural, and content patterns. The final section presents the conclusions drawn from the present work.

2 State of the Art of Ontology Design Patterns

In the field of ontology design patterns, a distinction can be made between logical and conceptual design patterns [6]. In the first case, the W3C Semantic Web Best

² <http://www.neon-project.org/>

³ <http://www.w3.org/TR/owl-features/>

Practices and Deployment Working Group (SWBPD)⁴ states that best practices are necessary to provide some hand-on support for developers and users of the Semantic Web. This group defines best practices as ‘a consensus-based guidance designed to facilitate Semantic Web deployment within RDF and OWL’; and the group proposes patterns for solving design problems for OWL (OWL design patterns), independently of a particular conceptualization, addressing logical problems. In the second case, [6] presents patterns for solving (in OWL or another logical language) design problems for the domain classes and properties that populate an ontology, addressing content problems.

In [7] some well known Semantic Web best practices (in particular those related to W3C activities⁵) are analyzed to present them in a so-called cook-book style. This cook-book style makes easier for teams to check whether the best practices are related to the modeling problems they have and, if so, how to apply them. The so-called cook-book style covers the following aspects of a concrete best practice: the problem(s); the solutions, that is, ingredients, required materials (ontology expressive power), and examples; and finally, the tips (discussions and pros and cons).

The experience in ontology engineering suggests recurrence of typical conceptual patterns [6] emerging out of different ontology projects (although the projects themselves are aimed at different tasks and involve experts having heterogeneous backgrounds). These emerging patterns include e.g. *participation* pattern (involving objects taking part in events) and the *role* \leftrightarrow *task* pattern (dealing with the temporary *roles* that objects can play, and with the *tasks* that events allow to execute).

In [6], the notion of ‘Content Ontology Design Patterns’ (‘CODEPs’) is introduced, and its difference with other sibling notions is discussed. Some examples of ‘CODEPs’ are given, and their usefulness in acquiring, developing, and refining ontologies from either experts or from documents is commented. ‘CODEPs’, for example, can be exploited as a tool to annotate ‘focused’ fragments of a reference ontology, i.e. the parts of an ontology containing the types and relations that underlay ‘expert reasoning’ in given fields or communities. They can be applied at different degrees of abstraction, and can be specialized or composed. ‘CODEPs’ are expressible in OWL DL, and their high reusability and formal and pragmatic nature make them suitable not only for isolated ontology engineering practices, but also (and especially) for distributed, collaborative environments like intranets, the Web or the Grid.

3 Catalog of Ontology Design Patterns

In this paper, we propose a catalogue of *ontology design patterns*; to exemplify them we focus on an OWL-based implementation of such patterns. This catalog presents a collection of *ontology design patterns* organized into three different categories [8]:

- **Logical ontology design patterns (LPs)**: elements of the OWL DL metamodel⁶ or compositions of such elements. An LP is a content-independent structure, i.e. an

⁴ <http://www.w3.org/2001/sw/BestPractices/>

⁵ <http://www.w3.org/2001/sw/BestPractices/>

⁶ <http://www.w3.org/TR/owl-features/>

untyped structure expressed only with a logical vocabulary. In the case of OWL DL, it can only be instantiated by elements in the owl namespace. An LP can be applied more than once in the same ontology to solve similar modelling problems. An LP affects only a specific and delimited part of an ontology, i.e. it does not affect the overall shape of an ontology.

- ❑ **Architectural ontology design patterns (APs):** LPs or compositions of them that are used exclusively in the design of an ontology. An AP is also a content-independent structure. In other words, an AP is supposed to characterize the overall structure of an ontology. In simple terms, an AP dictates ‘how an ontology should look like’.
- ❑ **Content ontology design patterns (CPs):** instances of an LP (or compositions of LPs). A CP is a typed structure expressed with a domain specific (non logical) vocabulary. A CP represents and solves a domain modelling problem and affects the (limited) part of the ontology dealing with that domain modelling problem [6].

Table 1 presents the current catalog which includes 18 LPs, 3 APs, and 6 CPs. Such ontology design patterns are described in detail in [8]. The catalog will be continuously updated at the NeOn website⁷.

Table 1. Catalog of ontology design patterns

<i>Logical Patterns</i>	
Primitive class	Union of classes
Defined class	Individual
Subclass-of relation	Disjoint classes
Multi-inheritance	Covering axiom
Equivalence relation	Defining n-ary relations: introducing a new class
Object property	Defining n-ary relations: using a list for arguments
Subproperty-of relation between object properties	Representing specified values in OWL: values as sets of individuals
Datatype property	Representing specified values in OWL: values as subclasses
Existential restriction	
Universal restriction	
<i>Architectural Patterns</i>	
Taxonomy structure	Lightweight ontology
	Modular architecture
<i>Content Patterns</i>	

⁷ <http://www.neon-project.org/>

Participation	Simple part-whole relations: part-whole relation Simple part-whole relations: part-whole class hierarchy
Description-situation	
Role-task	
Plan-execution	

Ontology design patterns of the above categories are described using a unified template, inspired by previous work on software engineering patterns [5], with the following slots [8]:

- ❑ *General Information*, which includes name, identifier and ontology design pattern category. These slots are mandatory.
- ❑ *Use Case*, which includes the problem to be addressed.
- ❑ *Ontology Design Pattern*, which includes the proposed solution in different formats. Here the content changes depending on the category of ontology design patterns.
- ❑ *Relations to other ontology design patterns*, which refers to possible relationships (use, specialize, etc.) with other patterns in the catalog. This slot is optional.
- ❑ *Comments*, which refers to remarks for clarifying the use of the pattern. This slot is also optional.

4 Logical Ontology Design Patterns

Logical ontology design patterns (LPs) are elements of the OWL DL metamodel or compositions of such elements. An LP is a content-independent structure expressed only with a logical vocabulary.

For describing LPs in the catalog, we propose the specific template shown in Table 2, which is based on the general one presented in Section 3.

Table 2. Template for logical patterns

Slot	Value
<i>General Information</i>	
<i>Name</i>	Name of the ontology design pattern
<i>Identifier</i>	An acronym composed of: category + pattern + number
<i>Ontology Design Pattern Category</i>	Logical Pattern (LP)
<i>Use Case</i>	
<i>General</i>	Description in natural language of the general problem addressed by the ontology design pattern.

<i>Examples</i>	Description in natural language of some examples for the general problem.
Ontology Design Pattern	
<i>Informal</i>	
<i>General</i>	Description in natural language of the general solution provided by the ontology design pattern, using elements from the OWL DL Metamodel.
<i>Examples</i>	Description in natural language of the solution applied to the examples.
<i>Graphical</i>	
<i>(UML) Diagram for the General Solution</i>	Graphical representation of the general solution provided, taking into account the UML Profile proposed in [2].
<i>(UML) Diagram for Examples</i>	Graphical representation of the solution provided, using examples and taking into account the UML Profile proposed in [2].
<i>Formalization</i>	
<i>General</i>	Formalization of the pattern in terms of the OWL DL Metamodel.
<i>Examples</i>	Formalization of the examples (using abstract syntax for OWL code).
Relationships	
<i>Relations to other ontology design pattern</i>	Description of any relation to other ontology design pattern (use, specialize, etc.).
Comments	
<i>Comments</i>	Remarks for clarifying the use of the ontology design pattern.

As an example of LP in the presented catalog of ontology design pattern, Table 3 shows the pattern for *defining n-ary relations* by means of creating a new class and *n* new object properties that represent an n-ary relation⁸.

Table 3. Logical pattern for modelling n-ary relations: introducing a new class for the relation

Slot	Value
General Information	
<i>Name</i>	N-ary Relation: New Class
<i>Identifier</i>	LP-NR -01

⁸ <http://www.w3.org/TR/swbp-n-aryRelations/>.

<i>Ontology Design Pattern Category</i>	Logical Pattern (LP)
<i>Use Case</i>	
<i>General</i>	<p>Express that:</p> <p>A binary relationship really needs a further argument.</p> <p>Two binary relationships always go together and should be represented as one n-ary relation.</p> <p>A relationship is really amongst several things.</p>
<i>Examples</i>	Suppose that someone wants to express that 'business plans' have 'business tasks' with a concrete 'duration'.
<i>Ontology Design Pattern</i>	
<i>Informal</i>	
<i>General</i>	<p>Create a new class and n new object properties.</p> <p>Therefore, instantiate the classes <code>Class</code> and <code>ObjectProperty</code>.</p>
<i>Examples</i>	<p>Create the classes 'BusinessPlan', 'BusinessTask', 'hasBT_Relation', and 'Duration'.</p> <p>In the definition of the class 'BusinessPlan', specify an object property 'hasBusinessTask' with the range restriction going to 'hasBT_Relation' class.</p> <p>Define 'task_Value' and 'hasDuration' as functional object properties.</p> <p>In the definition of the class 'hasBT_Relation', specify two object properties 'task_Value' and 'hasDuration' with the range restriction going to the classes 'BusinessTask' and 'Duration', respectively.</p>
<i>Graphical</i>	
<i>(UML) Diagram for the General Solution</i>	<div style="text-align: center;"> <p>The diagram consists of a rectangular box labeled 'Class' and a diamond-shaped box labeled '<owl:ObjectProperty>'. The diamond is positioned below the rectangle.</p> </div>
<i>Formalization</i>	
<i>General</i>	<pre>Class(Class partial OntologyElement) Class(Property partial OntologyElement) Class(ObjectProperty partial Property)</pre>
<i>Examples</i>	<pre>Class(BusinessPlan partial restriction(hasBusinessTask</pre>

	<pre> allValuesFrom(hBT_Relation)) owl:Thing) Class(hBT_Relation partial owl:Thing restriction(task_Value someValuesFrom(BusinessTask)) restriction(hasDuration allValuesFrom(Duration))) Class(BusinessTask partial owl:Thing) Class(Duration partial owl:Thing) ObjectProperty(task_Value Functional domain(owl:Thing)) ObjectProperty(hasDuration Functional domain(owl:Thing)) </pre>
Relationships	
<i>Relations to other ontology design patterns</i>	Possible use of this LP in APs and CPs.

5 Architectural Ontology Design Patterns

Architectural ontology design patterns (APs) are LPs or compositions of them that are used exclusively in the design of an ontology.

For describing APs in the catalog, we also propose a specific template, based on the general template presented in Section 3. As we mentioned in such section, the content of the slot called ‘Ontology Design Pattern’ depends on the pattern category. For this reason, Table 4 only shows such slot of the specific template for APs; the remainder of the slots is similar to those presented in Table 2.

Table 4. Template for architectural patterns: ontology design pattern slot

Slot	Value
Ontology Design Pattern	
<i>Informal</i>	
<i>General</i>	Description in natural language of the general solution provided by the ontology design pattern, in terms of the OWL DL Metamodel.
<i>Examples</i>	Description in natural language of the instantiated solution for the examples.
<i>Graphical</i>	
<i>(UML) Diagram for the General Solution</i>	Graphical representation of the general solution provided, taking into account the UML Profile proposed in [2].

<i>(UML) Diagram for Examples</i>	Graphical representation of the solution provided, using examples and taking into account the UML Profile proposed in [2]. This slot could be optional.
-----------------------------------	---

6 Content Ontology Design Patterns

Content ontology design patterns (CPs) are instances of an LP (or compositions of LPs). A CP is a typed structure expressed with a domain specific (non logical) vocabulary.

For describing CPs in the catalog, we propose a specific template, based on the general template presented in Section 3. As we mentioned in that section, the content of the slot called ‘Ontology Design Pattern’ depends on the pattern category. For this reason, Table 5 shows only such slot of the specific template for CPs; the remainder of the slots is similar to those presented in Table 2.

Table 5. Template for content patterns: ontology design pattern slot

Slot	Value
<i>Ontology Design Pattern</i>	
<i>Informal</i>	
<i>General</i>	Description in natural language of the general solution provided by the ontology design pattern, in terms of the OWL DL Metamodel. In this case we focus on a generic domain.
<i>Examples</i>	Description in natural language of the solution provided using examples. In this case we focus on a specific domain. This slot could be optional.
<i>Graphical</i>	
<i>(UML) Diagram for the General Solution</i>	Graphical representation of the general solution provided, taking into account the UML Profile proposed in [2].
<i>(UML) Diagram for Examples</i>	Graphical representation of the solution provided, using examples and taking into account the UML Profile proposed in [2]. This slot could be optional.
<i>Formalization</i>	
<i>General</i>	Formalization of the pattern in terms of the most general classes and properties in OWL abstract syntax.
<i>Examples</i>	Formalization of specialized solution for the examples (using abstract syntax for OWL code). This slot could be optional.

7 Conclusions

In this paper we briefly mention the work done in design patterns within the software engineering field and we present a short summary of research in design patterns within the ontology engineering field.

As in Software Engineering, a well-accepted and commonly used format for describing patterns is also needed in Ontology Engineering. For this reason, in this paper we propose a general template for representing ontology design patterns (for OWL DL) based on software engineering experiences. The main idea underlying such a template is to facilitate the understanding and reuse of ontology design patterns by teams that model ontologies.

Moreover, to help teams in the task of reusing ontology design patterns we provide a catalog of patterns (exemplified with OWL DL) classified into three different categories: (1) logical patterns (untyped structures expressed only with logical vocabulary that solve modelling problems); (2) architectural patterns (untyped structures expressed through a combination of LPs and related to ‘how the ontology looks like’); and (3) content patterns (typed structures expressed with a domain specific (non logical) vocabulary). For each category we propose a specific template based on the general pattern presented in this paper.

Acknowledgments. This work has been supported by the NeOn project (IST-2005-027595). We are very grateful to our NeOn partners for their revisions and comments. We are also very grateful to Rosario Plaza for her revisions and comments.

References

1. Alexander, C. *The timeless way of building*. Oxford University Press, New York (1979).
2. Brockmans, S.; Haase, P. *A Metamodel and UML Profile for Networked Ontologies. A Complete Reference*. Technical report, Universität Karlsruhe, April 2006. Available at: <http://www.aifb.uni-karlsruhe.de/WBS/sbr/publications/ontology-metamodelling.pdf>.
3. Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley and Sons Ltd, Chichester, UK (1996).
4. Devedzic, V. *Understanding Ontological Engineering*. Communications of the Association for Computing Machinery, 45(4):136–144, 2002.
5. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA (1995).
6. Gangemi, A. *Ontology Design Patterns for Semantic Web Content*. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference, Galway, Ireland, 2005. Springer.
7. Pan, J.Z.; Lancieri, L.; Maynard, D.; Gandon, F.; Cuel, R.; Leger, A. *Knowledge Web Deliverable D1.4.2.v2. Success Stories and Best Practices*. January 2007. Available at: <http://www.csd.abdn.ac.uk/~jpan/pub/TR/D142v2-final.pdf>.
8. Suárez-Figueroa, M.C.; Brockmans, S.; Gangemi, A.; Gómez-Pérez, A.; Lehmann, J.; Lewen, H.; Presutti, V.; Sabou, M. NeOn. Deliverable 5.1.1. NeOn Modelling Components. February 2007. Available at: <http://www.neon-project.org/>.