Ontology Patterns - Typology and Experiences from Design Pattern Development

Eva Blomqvist [*]
StLab, ISTC-CNR, via Nomentana 56, 00161 Roma, Italy.
e-mail: eva.blomqvist@istc.cnr.it

April 27, 2010

Abstract

Patterns for ontology engineering have received an increased interest during the past few years. Ontology patterns facilitate knowledge reuse and aid the task of engineering application ontologies. Ontology patterns provide general solutions, which are encoded and stored for reuse purposes. This paper first discusses the nature and characteristics of ontology patterns, and presents a typology of ontology patterns that can be used as an informal vocabulary when presenting and reusing patterns. Secondly, we introduce a catalogue of ontology design patterns, for the domain of product development, which were re-engineered from existing knowledge sources, and we especially list some experiences from this re-engineering process. As future work we envision an increased tools support for pattern development and usage, as well as further experiments using the proposed pattern catalogue.

1 Introduction

Ontology engineering, for the Semantic Web and other application fields, is a tedious and error prone process, requiring expertise in knowledge modelling and logical languages. With the emergence of the Semantic Web [3], Linked Data [4], and the introduction of semantic technologies in different application areas, e.g. Content Management Systems[1], the usage of (logically) light-weight application ontologies has drastically increased. Such ontologies are not heavily axiomatized, but provide just a bit of formal semantics to a data set. Thereby web and system developers in these fields have become more and more interested in ontology engineering, e.g. on the Semantic Web it is no longer the case that most ontologies are carefully constructed by knowledge engineers, instead they are drafted by people who have only a brief knowledge of the underlying logical formalisms of ontology languages (such as the Description Logics-based Web Ontology Language, OWL). An example of this trend is the successful series of VoCamps[2] arranged during the past two years, where researchers and practitioners get together and during one or two days draft vocabularies, i.e., small ontologies, to be used for describing some dataset.

To exploit the full benefits of the Semantic Web, and other semantically-enhanced systems, application ontologies therefore need to be easy to construct. Motivated by this development we are focussing on knowledge reuse through *ontology patterns*. Patterns are recurrent solutions to design problems that can be stored and retrieved from catalogues, and reused when engineering ontologies and semantic applications. However, so far there exist no coherent and uniform classification model for ontology patterns, hence, there is no common vocabulary for discussing patterns.

Patterns have so far mainly been available for addressing syntactic representation and logical modelling problems (e.g. the OWL patterns proposed

[*] The research presented in sections 2 and 3 was financed by, and conducted at, a previous affiliation of the author; the Information Engineering research group at Jönköping University, Sweden.

[1] See the IKS project at http://www.iks-project.eu/

[2] http://vocamp.org/wiki/WhatIsVoCamp

by the W3C[3], and logical patterns as in the catalogue maintained by the University of Manchester[4]), and even when present they are still quite scarce and often very general, in terms of domain applicability. General patterns are highly reusable, but in addition hard to match to the problem at hand, and reuse in a correct manner. These drawbacks support the need for pattern catalogues tailored for more specific domains, such as product development application ontologies, which is the target of this paper. The ideal procedure for pattern development is an organic growth of a pattern catalogue from the joint experiences of a community. However, this process may take several years. In order to kick-start the usage of patterns we propose to also re-engineer other patterns (e.g. data model patterns) into ontology patterns. For data model patterns domain specific catalogues already exist and are freely available, although not formalized.

This paper starts by introducing some basic notions of ontology engineering in section 1.1. Section 2 proposes a pattern typology and some general characteristics for describing patterns. In section 3 we address the problem of developing content ontology design patterns (Content ODPs) based on existing knowledge sources, and their initial evaluation and refinement. Related work is mentioned in section 4, and as an introduction to ongoing and future work section 5 briefly presents two tools for ontology pattern development and reuse, before we conclude with the discussion in section 6.

1.1 Ontology Engineering

In this paper we focus on the notion of application ontologies [16], i.e. ontologies built and tailored for a specific application scenario, as opposed to for example general domain ontologies or abstract top-level ontologies. Engineering of application ontologies is a continuous process incorporating the complete life-cycle of an ontology[14]; everything from the description of its intended application, requirements engineering, ontology construction, ontology reuse, to deploying the ontology in the application, maintaining, and evolving it. An important part of ontology engineering is the actual modelling of the ontology, which is the main focus of the patterns in this paper. A set of questions (first proposed by the author in [5], based on an overview of the field) can briefly summarize the problem areas on different abstraction levels that need to be addressed in order to build (model) an ontology:

1. What is the purpose of the ontology; how is it to be applied in a software system?
2. What parts are to form the ontology; how should the architecture be formed?
3. What should the ontology contain; what concepts, relations, and axioms?
4. How should the ontology be represented syntactically?

The first question pertains to the requirements and the interface of the ontology. An application ontology has a well-defined set of requirements and should be tailored to the way it will be used in a software system. The second question concerns the overall architecture of the ontology, e.g. the different knowledge domains to include, the different views needed etc. The third question addresses the detailed design and content of the ontology, i.e., how the individual requirements will be realized inside the overall structure given by the ontology architecture. Finally, the fourth question deals with representation, e.g., the logical language to represent the ontology, efficiency of reasoning, and usability (in terms of understandability by humans).

2 Pattern Characteristics and Typology

We generally define *ontology pattern* as:

Definition 1. *An ontology pattern is a set of ontological elements, structures or construction principles that intend to solve a specific engineering problem and that recurs, either exactly replicated or in an adapted form, within some set of ontologies, or is envisioned to recur within some future set of ontologies.*

Based on this definition a pattern has to provide a solution to a specific engineering problem or category of problems (*problem* focus), and recur in existing solutions or envisioned solutions (*reuse*

[3]http://www.w3.org/2001/sw/BestPractices/OEP/

[4]http://www.gong.manchester.ac.uk/odp/html/index.html

focus). The nature of the 'engineering problem' is not important, however, it has to involve ontologies in some way or another, e.g. it can be any problem where an ontology is seen as the solution or the construction and maintenance of that ontology. These are considered as necessary and sufficient conditions. A pattern is usually also based on 'best practices', but this is not a necessary condition. A pattern that does not follow such best practices, but instead exhibits a common problem or mistake is usually denoted an *anti-pattern*.

2.1 Basic Perspectives on Patterns

Regarding their origin, either patterns are purely experience-based, i.e., produced *bottom-up* through some pattern mining or recognition technique, or they can be carefully designed and constructed, as abstract templates, i.e., produced *top-down*. From the bottom-up perspective, patterns are recurring structures in some set of solutions. In contrast, from the top-down perspective, patterns are templates that represent a consensus view on a specific problem. In ontology engineering both perspectives exist, including hybrid approaches.

2.1.1 Structure or Content Focus

Structural patterns deal with the logical structure of the ontological elements, but not with the actual ontology represented by these. A structural pattern has an empty signature, i.e., on the design level no actual named classes and properties are proposed by the pattern. An example of such a structural pattern (with an empty signature) is the logical macro 'disjoint union' (which in OWL 2 has its own language construct but in OWL 1 could merely be described as an abstract pattern). The pattern expresses the notion of an exhaustive partition of a class into a set of subclasses, which are all mutually disjoint. In contrast, content patterns deal with ontological modelling solutions, hence, they are domain specific, where the scope of the domain depends on their level of ontological abstraction. An example of a content pattern (with a non-empty signature) is the *Situation*[5] pattern, which is a instantiation of the structural n-ary relation pattern within the domain of situations, i.e. it contains a class 'Situation' which is the projection of the n-ary relation, and two properties 'isSettingFor'/'hasSetting' to indicate the things involved in the situation. Since the pattern contains actual named classes and properties it has a non-empty signature, i.e. it proposes a vocabulary for situations. Structural and content design patterns have been discussed in detail in [12].

[5]http://ontologydesignpatterns.org/wiki/Submissions:Situation

2.1.2 Abstraction and Granularity

Granularity is concerned with the scope of the pattern, e.g. treating some small part of an ontology in detail or treating a complete ontology. Whatever level of granularity is chosen, patterns are focused on solving *one specific problem*, on that specific level of granularity. Abstraction is about hiding the details of some structure in order to describe another aspect of the structure in a more convenient manner, thereby 'abstracting' from the details. Abstraction is more than a change of granularity however; a change in abstraction means switching perspective to view completely different aspects, e.g. by using a new form of representation.

2.2 Pattern Typology

We define four levels of abstraction, including applicable levels of granularity, of ontology patterns. A very preliminary proposal for granularity levels was suggested in [7]. In this paper we present a more elaborate classification where (i) the levels have only one differentiating notion, while in [7] abstraction and granularity were used simultaneously, and (ii) the levels have been detailed and given intensional definitions. The abstraction levels, and corresponding levels of granularity, of ontology engineering patterns provide a top-down framework for classifying and describing ontology engineering patterns; a pattern typology. The framework can be illustrated as in Table 1. The levels are described in detail below and each correspond to one of the questions in section 1.1.

2.2.1 Abstraction Level: Application Patterns

To address the first question of ontology engineering, as listed in section 1.1, application patterns are intended to address problems concerned with the

Table 1: Ontology pattern typology.

Abstraction	Granularity
Application patterns	Overall ontology
Architecture patterns	Overall ontology Ontology module
Design patterns	Overall ontology Ontology module Logical elements
Syntactic patterns	Overall ontology Ontology module Logical elements Element syntax

overall scope and purpose of the ontology. There is only one level of granularity possible, i.e. viewing the complete ontology as a unit. In [1] the notion of ontology *application pattern* was defined as follows:

Definition 2. *An ontology application pattern is a software architecture pattern describing a software system that utilises ontologies to create some of its functionality. The pattern describes properties of the ontology, or ontologies, in the system, and the connection between the ontology and the rest of the system.*

Examples of ontology application patterns can be found in approaches such as by Harmelen et al.[24], which define common reasoning patterns that can be realized by ontologies to support some functionality of a software system.

2.2.2 Abstraction Level: Architecture Patterns

To address the second question in section 1.1, architecture patterns are concerned with the overall organisation of the ontology. An ontology *architecture pattern* is defined as:

Definition 3. *An ontology architecture pattern is a pattern describing the overall structure of the ontology, prescribing certain construction principles and restricting the selection of design patterns that can be used to implement the ontology.*

Note that on this level of abstraction, the details of the ontology (concepts and relations) are not considered, nor how to realize the ontology in some logical language. Ontology architecture patterns can be of two different levels of granularity, either treating the complete ontology or only part of the ontology, e.g. one ontology module. To the best of our knowledge, there are so far no ontology architecture patterns proposed in literature.

Analogous to software architectures, also ontologies can benefit from more detailed descriptions of typical architectures. In software engineering a reference architecture can be defined as a set of architecture patterns applied to a specific well-known problem [2], however, note that this is still not a concrete software architecture, i.e. it is a problem decomposition mapped onto abstract components. We analogously define the notion of ontology reference architecture as a domain specific instance of one or more ontology architecture patterns:

Definition 4. *An ontology reference architecture is a domain-specific ontology architecture pattern containing restrictions and requirements on both the structure and content of the complete ontology.*

2.2.3 Abstraction Level: Design Patterns

The third question in section 1.1 concerns the level of design patterns, addressing the actual modelling of the ontology. We define an ontology *design pattern* as:

Definition 5. *An ontology design pattern is a set of ontological elements, structures or construction principles that solve a clearly defined particular modelling problem.*

Ontology design patterns describe solutions on the abstraction level of logical ontology modelling languages, on three levels of granularity; 1) treating individual logical elements, e.g. logical axioms or macros, 2) treating a smaller part of the ontology, e.g. one module, or 3) treating the complete ontology, e.g. restricting the overall structure on the logical level. The latter is not to be confused with the architecture patterns previously described. On the abstraction level of architecture patterns the overall structure is in focus, without considering the logical realisation of that structure on the modelling level. While on the design level, an example on the granularity level dealing with the complete ontology could be 'taxonomy', i.e. restricting the logical constructs of the complete ontology but on the abstraction level of logical constructs.

So far, most of the ontology patterns proposed are on the level of design patterns. For example, the structural patterns developed by the University of Manchester[6] and the W3C[7], and the content patterns collected in the ODP Portal[8].

2.2.4 Abstraction Level: Syntactic Patterns

The final question of section 1.1 concerns representation; e.g. naming of ontology elements or the ontology itself, and how the elements and axioms are represented in some machine processable syntax. We define a *syntactic pattern* as:

Definition 6. *A syntactic ontology pattern is the realisation of other ontology patterns, or parts or combinations of ontology patterns, in a specific representation syntax.*

This definition adds an additional lowest level of granularity for this type of pattern, where single strings and character combinations are the core of the pattern. Most ontology representation syntaxes have their own common constructs, or in software engineering terms, their own language idioms. For example, lexico-syntactic patterns can be considered as syntactic patterns of ontologies, e.g. Hearst patterns [18] representing subclass relations. Other examples are naming conventions [23], e.g. using the camel convention for naming classes and properties, or using verbs for property names.

3 Pattern Catalogue

During the course of our research we have developed a set of content ontology design patterns (Content ODPs), on the granularity level of solving small design problems (modules) within an ontology. This means that the patterns themselves are small ontologies, with additional annotations describing them, such as the general requirements that each pattern solves (usually expressed in the form of Competency Questions [15]). The patterns have been re-engineered from different sources, mainly data model patterns. This section describes the re-engineering process, and the refinement process leading up to the submission of a set of patterns, i.e. a small product development pattern language (c.f. pattern languages in software engineering), to the ODP Portal[9], and the experiences collected during this process.

3.1 Re-engineering

The initial re-engineering steps have already been discussed in [5], hence, we only give a brief overview of the process in this section. The initial set of Content ODPs developed consisted of 26 patterns. The sources were, in addition to data model patterns [17, 21, 20]: software analysis patterns [9], top level ontologies [11], goal structures [22], and cognitive patterns [13]. The set of patterns was constructed with a specific domain in mind, i.e. product development application ontologies (hence not a specific industry domain), intending to create a small pattern language for this domain. Initially a simple translation approach was applied, defining a one-to-one mapping between constructs in the original representation of the sources and ontology element types, e.g. an entity in a data model pattern was mapped to a concept in the ontology language.

The initial set of patterns was tested during the development of a requirements engineering ontology in the context of the research project SEMCO, see [6]. A subset of the patterns (randomly selected) were evaluated together with domain experts. Based on the initial feedback the patterns were updated (details in [6]), and some were removed from the catalogue due to their low understandability. The remaining patterns were translated into OWL, since this language had by this time emerged as the W3C recommendation. This additionally meant removing a few patterns, since not all translated well into OWL, i.e. some inherently assumed an information modelling paradigm not easily translatable into Description Logics.

The remaining patterns were now tested in a rerun of the SEMCO experiment, as well as two other research projects, see [5], and compared to a set of more general patterns from the ODP portal. The interesting conclusion was that even though the

[6]http://www.gong.manchester.ac.uk/odp/html/index.html

[7]http://www.w3.org/2001/sw/BestPractices/OEP/

[8]http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs

[9]http://ontologydesignpatterns.org

patterns in the domain-specific catalogue were intended for the product development domain, some patterns (and parts of patterns) were actually applicable also in new domains (in this case university education and agriculture).

3.2 OWL 'Best Practices' and ODPs

Based on experiences from these experiments a subset of the initial catalogue, 19 patterns, were considered to have been very valuable, hence worth exposing to the ontology pattern community. This set was selected to be published in the ODP Portal. Before publishing however, we decided to apply some of the best practices encoded in other ODPs already present in the portal, in order to (i) increase the model quality (i.e. by applying modelling best practices) of the re-engineered pattern candidates in our catalogue, and to (ii) align our ODPs to existing ones, i.e. reuse existing patterns where applicable instead of redefining constructs.

This process was performed as an ODP-driven ontology evaluation, taking each candidate ODP from our catalogue at a time and identifying (1) modelling issues that did not follow OWL best practices, and (2) overlap with existing ODPs in the ODP portal. For each issue (1) we re-modelled our candidate ODP to apply the existing best practice. This included changes such as adding inverse relations, adding comments and labels, as well as changing the model to remove particularities introduced by the source pattern, e.g. an extra class representing a many-to-many relation in a database setting (in an OWL model this class has no added value). For each issue (2) we replaced the overlapping part of our candidate ODP by importing the existing ODP and if necessary specializing it, i.e. introduce the original terminology rather than the abstract one of the imported ODP.

One major issue we found, related to (1), i.e. modelling best practices, was the size and overlap of our candidate ODPs. In accordance with cognitive principles ODPs should be small enough so that all aspects can be visualized at the same time, and the main intent grasped more or less immediately [12]. Some of our candidate ODPs had between 20 and 40 classes, and covered several aspects, rather than *one* specific modelling problem. Additionally, several candidate ODPs covered common concepts, hence, we decide to extract these common parts.

Table 2: The pattern candidates.

Pattern name	Key terms
Action	action, plan
Analysis Modelling	analysis approach, modelling
Communication Event	event, contact mechanism, relationship
Employee Department	employee, department
Engineering Change	change, impact, status, specification
Information Acquisition	elicit, record, documentation
Organisation	organisation, informal, legal
Part specification	product, part, specfication
Party	organisation, person, classification
Person	name, birth date, gender
Planning Scheduling	plan, allocate, order
Position	party, position, fulfillment
Product	product, good, service
Product Association	product, complement, substitute
Product Category	product, category, classification
Product Feature	feature, applicability, feature interaction
Requirement	requirement, product, internal, customer
Requirement Description	requirement, product, feature, deliverable
Requirements Analysis	analysis, modelling, acquisition
Validation Testing	criteria, strategy, metric
Work Effort	effort, requirement

One such 'new' pattern was the Product pattern, representing information about products. This structure was previously present in several of the other patterns, but in the final revision the product information was collected in a separate pattern, which is then imported by other patterns where needed. Similarly, information about requirements is now represented separately, since it recurred in several patterns. The final set of 21 patterns are listed in Table 3.2. Due to space restrictions we are not able to go into details of all the patterns, however an example is presented in section 3.4. The 'key terms' in the table are concept labels, or parts of labels, from the pattern that can be seen as example keywords indicating the topic of the pattern. These patterns are now being submitted to the ODP Portal repository of patterns, to await scrutiny by the pattern community and the quality committee of the ODP Portal.

3.3 Experiences

From the re-engineering effort, and the subsequent evaluations and refinement of the patterns, we have gained a number of valuable experiences. One experience is related to the nature of the sources of re-engineering. Since none of the sources were expressed in a formal language, but rather descriptions and example models in a book, the direct mapping method that we applied created some less than optimal solutions. For example, the data model sources commonly contained entities such as 'other x', indicating a concept that would contain 'the rest' in an otherwise non-exhaustive partition. Such entities are relevant when transforming for example an ER-model into a relational database, since there needs to be a table storing these 'other x'. While, when using ontologies, this restriction is not present, an instance can simply have the type of the superclass, if this is not defined as an exhaustive partition, hence such 'other x' entities should not be transformed into concepts during re-engineering, but only be seen as an indication of a non-exhaustive partition.

When considering data model patterns they are commonly tailored for a smooth transformation to a relational database, hence they often contain 'extra' entities representing the many-to-many relations that require an additional table in the relational model. This is an additional problem that is not present in OWL/RDF, where data is stored in triples. Thereby, identifying these superfluous entities and avoiding to transform them to concepts is essential for arriving at an appropriate model. Additionally, when using ontologies the definition of certain concepts is more intuitive than when using a relational model, hence concepts such as gender should in an ontology be axiomatized while in a relational model this will most often be a simple character string programmatically restricted to accept certain values. Cases where ontologies give an opportunity to define certain concepts more explicitly than allowed by other models, e.g. OWL axioms instead of literals, should be identified.

Some pattern candidates were, in the last refinement step, split into several patterns, to decrease redundancy. To identify what are the natural borders and relations between patterns is an important step. When studying the existing patterns in the ODP Portal, some reuse opportunities were also discovered, where general patterns, such as 'sequence'[10] and 'partOf'[11], were reusable by the patterns in the catalogue. In the current version of the pattern catalogue, several patterns reuse more general patterns by importing and specializing them. This is important from two respects; (1) to align the ODP candidates to existing well-known Content ODPs, and (2) to reduce redundancy in the catalogue of proposed ODPs by reusing existing solutions instead of re-defining them.

[10]http://ontologydesignpatterns.org/wiki/Submissions:Sequence

[11]http://ontologydesignpatterns.org/wiki/Submissions:PartOf

3.4 Example Pattern

Due to space restrictions we cannot present the complete catalogue in detail, however we here give a representative example, i.e. a Content ODP named **Action**, in order to illustrate the nature of the patterns. It represents the relations between different types of actions, the state of the actions, and defines the relation between a plan and a set of proposed actions. Figure 1 shows a diagram of the Action Content ODP. In addition to the diagrammatic representation Content ODPs are commonly described using a number of catalogue entry fields (c.f. software pattern templates), such as *name*, *intent*, *consequences*, and *building block* (linking to an OWL realization of the pattern).

4 Related Work

In parallel with this work another typology of ontology patterns was developed, as described in [12]. The typologies, and the terminologies, are related and complementary. They represent two perspectives of ontology patterns. The classification schema proposed in this paper takes a top-down approach, aiming to cover the complete development process of an ontology, in terms of the drill-down from highly abstract requirements to how the ontology is represented. In [12] the authors instead take a bottom-up approach, starting from what activities during the ontology life-cycle are actually performed using patterns, or following best practices. Hence, the terminologies are overlapping, and most of the types of [12] can be classified under one of the main categories presented in this paper, as long as the focus is on constructing the actual ontology. Related (non-design) activities are not covered by our approach. Our approach focuses on a coherent classification, where the differentiating notions between levels and subdivisions are uniform, rather than solely related to current practice (as in [12]).

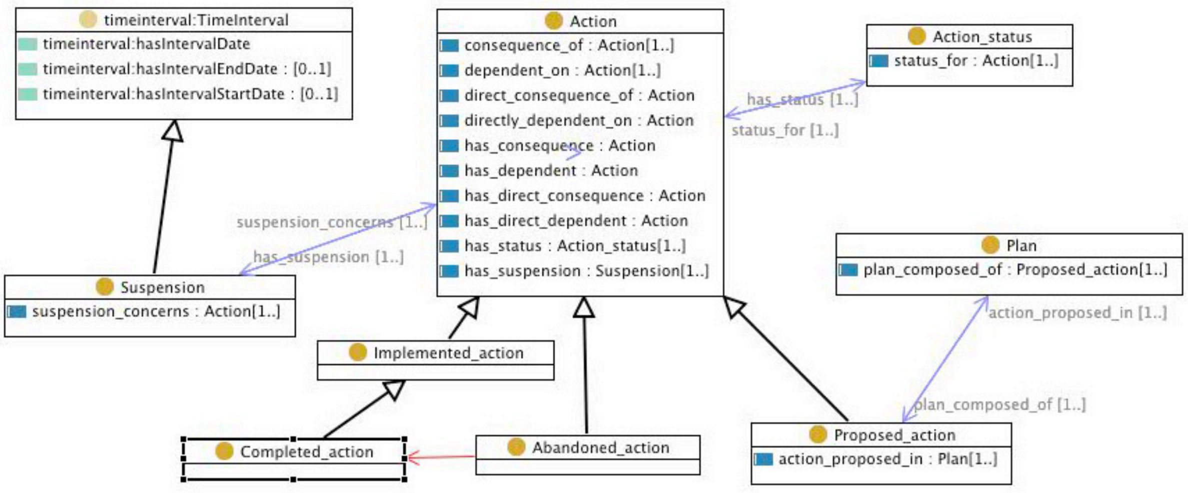
timeinterval:TimeInterval
timeinterval:hasIntervalDate
timeinterval:hasIntervalEndDate : [0..1]
timeinterval:hasIntervalStartDate : [0..1]
Action
consequence_of : Action[1..]
dependent_on : Action[1..]
direct_consequence_of : Action
directly_dependent_on : Action
has_consequence : Action
has_dependent : Action
has_direct_consequence : Action
has_direct_dependent : Action
has_status : Action_status[1..]
has_suspension : Suspension[1..]
Action_status
status_for : Action[1..]
has_status [1..]
status_for [1..]
suspension_concerns [1..]
has_suspension [1..]
Suspension
suspension_concerns : Action[1..]
Plan
plan_composed_of : Proposed_action[1..]
action_proposed_in [1..]
plan_composed_of [1..]
Implemented_action
Completed_action
Abandoned_action
Proposed_action
action_proposed_in : Plan[1..]

Figure 1: The Action Content ODP's graphical representation in UML (based on the OWL model).

The main difference in terminology is concerned with the term 'design'. While [12] denotes all patterns 'ontology *design* patterns', referring to the focus on *designing* ontologies, our notion of ontology design patterns has a narrower scope, and refers to patterns related to the actual logical design of the ontology, i.e. solving logical modelling problems. Also, the notion of ontology architecture pattern is slightly different; in this paper it is used only for patterns that are focused on the overall structure of the ontology, disregarding logical representation, while [12] in addition uses it for logical restrictions on the design level that concern the complete ontology, i.e. mixing two levels of abstraction.

Related work on collecting catalogues of ontology patterns include the ODP Portal[12] where patterns are collected in a Wiki-based interface, letting any community member contribute new patterns and review other users' patterns. Other pattern catalogues also exist, such as the logical patterns collected by the W3C[13] and by the University of Manchester[14]. A recent pattern re-engineering effort was described in [10], concerning the transformation of a component library into ODPs, however, we are not aware of any other similar efforts for bootstrapping ODPs.

5 Current Tool Support

Some ontology patterns are suitable for description as abstract templates that can be reused as general 'ideas' when engineering ontologies, while other types of patterns can be more formally represented and reused as concrete building-blocks. In this section we briefly describe two tools for supporting the development and reuse of ontology patterns; the ODP Portal[15] intending to support the complete range of ontology patterns, and the XD Tools Eclipse plugin mainly focused on support for Content ODPs as reusable OWL building blocks.

The ODP Portal is a semantic wiki portal for collecting, distributing and discussing ontology patterns. Technically it is based on the Semantic Media Wiki[16] framework, and plugins such as the Evaluation Wiki Flow [8]. The portal is divided into areas devoted to different types of patterns, where the most prominent so far is the Content ODP area. Each area contains a catalogue of submissions and a certified catalogue. All members of the portal community are free to upload patterns in the submissions catalogue, while the certified catalogue is moderated by the portal's quality committee consisting of ontology engineering and pattern experts. Patterns are certified through a

[12]http://ontologydesignpatterns.org
[13]http://www.w3.org/2001/sw/BestPractices/OEP/
[14]http://www.gong.manchester.ac.uk/odp/html/index.html
[15]http://www.ontologydesignpatterns.org
[16]http://semantic-mediawiki.org/wiki/Semantic_MediaWiki

peer review process, following a certification request from the pattern author. Important features of the portal also include the possibility to provide open reviews, i.e. any community member can spontaneously review any pattern, and the possibility to post modelling issues and discuss possible solutions and patterns. The portal additionally provides pattern training opportunities, and has been used to support events such as the pattern track at WOP2009[17] (Workshop on Ontology Patterns co-located with ISWC2009).

The eXtreme Design Tools[18] (XD Tools) is an Eclipse plugin supporting the XD methodology for ontology design pattern reuse, as described in [19]. It is compatible with ontology engineering environments such as the NeOn toolkit[19] and TopBraid Composer[20]. The XD Tools currently focuses on supporting the reuse of content ontology design patterns (Content ODPs). Content ODPs are commonly posted as reusable OWL building-blocks (in the ODP Portal, see above) that can be specialized into ontology modules realizing particular requirements of a concrete application ontology. The XD Tools supports the specialization process through a wizard, enforcing best practices such as adding labels and comments to all entities and adding inverse properties when applicable. Additionally XD Tools includes functionality for ontology analysis, based on best practices and 'rules of thumb' for ontology design, as well as an annotation wizard for describing the ontology modules built (based on ontology annotation patterns imported into the tool).

6 Conclusions and Future Work

In this paper we have presented a top-down ontology pattern typology, based on the notion of ontology patterns as a development aid while engineering an application ontology. The main difference to other proposed typologies is the coherence of the different levels, i.e. they are uniformly distinguished. The proposed typology is also consistent with common practice in software engineering where the terminology is defined in a similar way, based on the abstraction level of the patterns. We believe that having a coherent terminology and a set of characteristics with which to describe ontology patterns is essential, and will facilitate communication between both researchers and ontology engineers. Future work with respect to the pattern typology is to further study patterns on the different levels, and exemplify them, e.g. patterns are so far are not present on the architecture level.

We also focused on a specific level, i.e. Content ODPs, and showed how such patterns can be re-engineered from sources similar to ontologies, in order to populate a pattern catalogue. In our case the catalogue was designed for the product development application ontology domain. The re-engineering effort has given us a set of valuable experiences that can be used as guidelines when re-enginereing similar knowledge sources. Future work includes to conduct more evaluation experiments using the patterns in the catalogue. The patterns will also be scrutinized by the ontology patterns community through the ODP Portal.

Finally, according to the motivation presented at the beginning of this paper we need to make it easier to construct good application ontologies, even for non-experts and common web developers. We believe that having patterns on all the levels of the presented typology will ease the building process, however more tool support is still needed. In this paper we presented two examples of current tools that particularly support the reuse of Content ODPs, however in the future ontology patterns on different levels of abstraction can support the introduction of ontology engineering environments more similar to CASE tools in software engineering, where the developer is guided through a structured process and best practices are enforced.

Acknowledgements

This work was supported by Jönköping University, Sweden, through research projects SEMCO (Swedish KK-foundation grant no. 2003/0241), MediaIlLOG (grant from the foundation Carl-Olof och Jenz Hamrins Stiftelse), and DEON (financed by STINT). The tools presented in section 5 were developed by the STLab ISTC-CNR, Italy, in the context of the EU-FP7 IP NeOn.

[17]http://ontologydesignpatterns.org/wiki/WOP2009:Main
[18]Available from: http://stlab.istc.cnr.it/stlab/XDTools
[19]http://www.neon-toolkit.org
[20]http://www.topquadrant.com/products/TB_Composer.html

References

[1] Thomas Albertsen and Eva Blomqvist. Describing ontology applications. In *Proceedings of the 4th European Semantic Web Conference (ESWC07)*, 2007.

[2] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Addison-Wesley, second edition edition, 2003.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American Magazine*, 2001.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal On Semantic Web and Information Systems*, 5(3):1–22, 2009.

[5] Eva Blomqvist. *Semi-automatic Ontology Construction based on Patterns.* PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology, 2009.

[6] Eva Blomqvist and Annika Öhgren. Constructing an enterprise ontology for an automotive supplier. In *Engineering Applications of Artificial Intelligence*, volume 21, April 2008.

[7] Eva Blomqvist and Kurt Sandkuhl. Patterns in Ontology Engineering: Classification of Ontology Patterns. In *Proceedings of the International Conference on Enterprise Information Systems 2005*, Miami Beach, Florida, May 24-28 2005.

[8] Enrico Daga, Valentina Presutti, and Alberto Salvati. http://ontologydesignpatterns.org and evaluation wikiflow. In *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008), Rome, Italy, December 15-17,2008*, CEUR Workshop Proceedings, 2008.

[9] Martin Fowler. *Analysis Patterns - Reusable Object Models.* Addison-Wesley, 1997.

[10] Aldo Gangemi and Vinay K. Chaudhri. Representing the component library into ontology design patterns. In *Proc. of the Workshop on Ontology Patterns (WOP 2009)*, volume 516. CEUR Workshop Proceedings,, 2009.

[11] Aldo Gangemi and Peter Mika. Understanding the Semantic Web through Descriptions and Situations. In *Proc. of the International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2003)*, Catania, Italy, 2003.

[12] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on Ontologies, 2nd Ed.*, International Handbooks on Information Systems. Springer, 2009.

[13] Karen Gardner, Alexander Rush, Michael Crist, Robert Konitzer, and Bobbin Teegarden. *Cognitive Patterns - Problem-solving Frameworks for Object Technology.* Cambridge University Press, 1998.

[14] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering.* Springer, 2004.

[15] Michael Gruninger and Mark S. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, 1994.

[16] Nicola Guarino. Ontology and Information Systems. In *Proceedings of FOIS'98*, pages 3–15, 1998.

[17] David C. Hay. *Data Model Patterns - Conventions of Thought.* Dorset House Publishing, 1996.

[18] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–545, Nantes, France, July 1992.

[19] Valentina Presutti, Enrico Daga, Aldo Gangemi, and Eva Blomqvist. extreme design with content ontology design patterns. In Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, and Vojtech Svatek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009*, volume 516. CEUR Workshop Proceedings, 2009.

[20] Len Silverston. *The Data Model Resource Book - A Library of Universal Data Models by Industry Types*, volume 2. John Wiley & Sons, 2001.

[21] Len Silverston. *The Data Model Resource Book - A Library of Universal Data Models for All Enterprises*, volume 1. John Wiley & Sons, 2001.

[22] Alistair Sutcliffe. *The Domain Theory - Patterns for Knowledge and Software Reuse.* Lawrence Erlbaum Associates, 2002.

[23] Ondrej Sváb-Zamazal and Vojtech Svátek. Analysing ontological structures through name pattern tracking. In Aldo Gangemi and Jerome Euzenat, editors, *Proceedings of EKAW 2008*, volume 5268 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2008.

[24] Frank van Harmelen, Annette ten Teije, and Holger Wache. Knowledge engineering rediscovered: towards reasoning patterns for the semantic web. In *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA*, pages 81–88. ACM, 2009.