Workshop on Ontology Patterns

WOP 2010

Papers and Patterns from the ISWC workshop

Introduction

Since the beginning of the semantic web initiative, ontologies have been referred to as the key tool for implementing the semantic web vision. The way they have been studied in, and brought to, this field has assumed that they are the same kind of ontologies used in artificial intelligence (AI) or studied in philosophy.

However, by looking at the most popular web ontologies we can conclude that this is debatable: On one hand semantic web ontologies have inherited some things from AI and philosophical ontologies, on the other hand web ontologies exhibit another level of simplicity, scalability, and modularity, as well as including contributions from, and being used by, the masses (see also EKAW 2010[1] theme).

Additional evidence of the need for a new ontology design paradigm comes from the growing interest in linked data on the web, which is an amazing breakthrough for the semantic web. The inherent basis of linked data is the data-driven paradigm, as opposed to a concept-driven one. This makes linked data authoring and publication easy but its consumption less than straightforward, especially considering the ontologies currently used for representing them: this constitutes a potential limitation on the possibility of a concrete realization of the semantic web vision in the near future. Ontology Patterns, and their related technologies, could be key for bridging the gap between linked data and ontologies, because they are conceived with simplicity, scalability, and modularity, as well as contributions from, and usage by, the masses in mind, without giving up the inheritance from AI and philosophical ontologies. Hence, Ontology Patterns could drive the next breakthrough for the semantic web.

The aim of WOP is to give researchers and practitioners a stage where to share their latest findings and emerging issues, as well as building a common language for ontology patterns. Furthermore, the WOP community is supported by the ontologydesignpatterns.org initiative, and uses it as its main mean of communication, e.g. for pattern submission, reviewing and discussion outside the workshop schedule. A workshop should be a practical and interaction-rich event, and for this reason WOP had three parts: regular papers, posters/demos, and "pattern writing", with a focus on the latter. The inspiration for this model comes from the pattern writing workshops for software patterns. The aim is to promote development and review of actual patterns, rather than papers describing patterns. Related events are also VoCamps for writing vocabularies for the Semantic Web.

[1] http://ekaw2010.inesc-id.pt/

We received 6 submissions for the paper and poster track of the workshop. The program committee selected three submissions for oral presentation and one submission as position paper. 6 ontology patterns were submitted to the workshop, of which 5 patterns were selected for presentation in the poster session, such patterns are described in these proceedings as extended abstracts. The workshop also included a pattern writing session on proposed modeling issues, which can be found on the workshop web site. Finally, we had a session on late breaking news short presentations. Further information about the Workshop on Ontology Patterns can be found at: http://ontologydesignpatterns.org/wiki/WOP2010.

Acknowledgments

We thank all members of the steering committee, program committee, authors and local organizers for their efforts and support. We appreciate support from the FP6 NeON Integrated Project, the FP7 IKS Integrated Project and the DEON project (STINT IG 2008-2011).

Eva Blomqvist
Vinay K. Chaudhri
Oscar Corcho
Valentina Presutti
Kurt Sandkuhl

October 2010

Organization

WOP2010 Chairs

Paper and poster chairs:
Valentina Presutti, STLab ISTC-CNR (IT)
Vinay K. Chaudhri, SRI International (US)

Pattern chairs:
Eva Blomqvist, Jönköping University (SE)
Oscar Corcho, Universidad Politécnica de Madrid (ES)

Proceedings chair:
Kurt Sandkuhl, Jönköping University (SE) and University of Rostock (DE)

Steering Committee

Eva Blomqvist, ISTC-CNR (IT)
Aldo Gangemi, ISTC-CNR (IT)
Natasha Noy, Stanford University (US)
Valentina Presutti, ISTC-CNR (IT)
Alan Rector, University of Manchester (UK)
Francois Scharffe, INRIA (FR)
Steffen Staab, University of Koblenz (DE)
Chris Welty, IBM Watson Research Center (US)

Program Committee

Alessandro Adamou, ISTC-CNR (IT)
Marie-Aude Aufaure, Ecole Centrale Paris (FR)
Vinay Chaudhri,SRI International (US)
Mathieu D'Aquin, Open University (UK)
Enrico Daga, ISTC-CNR (IT)
Violeta Damjanovic, Salzburg Research (AT)
Rim Djedidi, Paris-Sud University (FR)
Leigh Dodds, Talis (UK)
Henrik Eriksson, Linköping University (SE)
Aldo Gangemi, ISTC-CNR (IT)
Jose-Manuel Gomez, Universidad Politécnica de Madrid (ES)
Gerd Groener, University of Koblenz (DE)
Holger Lewen, AIFB University of Karlsruhe (DE)
Natasha Noy, Stanford University (US)
Wim Peters, University of Sheffield (UK)

Alan Rector, University of Manchester (UK)
Alan Ruttenberg, Science Commons Cambridge, MA (US)
Marta Sabou, Open University (UK)
Kurt Sandkuhl, Jönköping University (SE)
Francois Scharffe, INRIA (FR)
Steffen Staab, University of Koblenz (DE)
Mari Carmen Suárez-Figueroa. Universidad Politécnica de Madrid (ES)
Vojtech Svatek, University of Economics, Prague (CZ
Tania Tudorache. Stanford University (US)
Boris Villazón-Terrazas, Universidad Politécnica de Madrid (ES)
Chris Welty, IBM Watson Research Center (US)

Pattern Program Committee

Alessandro Adamou, ISTC-CNR (IT).
Rim Djedidi, University Paris Nord 13 (FR).
Gerd Groener, University of Koblenz-Landau (DE).
Enrico Motta, KMI Open University, Milton Keynes (UK).
Olaf Noppens, University of Ulm (DE).
Andrea Nuzzolese, ISTC-CNR (IT).
Catherine ROUSSEY, LIRIS-CNRS Cemagref (FR).
Alan Ruttenberg, Creative Commons (US).
Mari Carmen Suárez-Figueroa. Universidad Politécnica de Madrid (ES).
Vojtech Svatek, University of Economics, Prague (CZ).
Pierre-Yves Vandenbussche, INSERM (FR).
Boris Villazón-Terrazas, Universidad Politécnica de Madrid (ES).

Table of Contents

Keynote Talk

Archeology for Ontology Patterns

Chris Welty

IBM Watson Research
Hawthorne, NY 12540, USA
cawelty@gmail.com

Abstract

Ontology Patterns for the semantic web are closest in spirit to software patterns, e.g. [1]. They are, or should be, motivated by design experience, not philosophical tradition. The software pattern community was launched into prominence as the result of an effort in "software archeology": digging through existing software, observing and cataloging different solution methods, generalizing and classifying them in a sensible framework, and publishing the result. In this talk I will argue for an archeological and less theoretical approach to ontology patterns, with examples.

References

1. Gamma, Helm, Johnson, and Vlissides (1995) Design Patterns. Addison-Wesley, 1995.

Further Information

For further information, please visit:
http://ontologydesignpatterns.org/wiki/WOP:2010/KeynoteTalk

Research Papers

The State of Ontology Pattern Research
A Systematic Review of ISWC, ESWC and ASWC 2005–2009

Karl Hammar and Kurt Sandkuhl

School of Engineering at Jönköping University
P.O. Box 1026
551 11 Jönköping, Sweden
haka, saku@jth.hj.se

Abstract. While the use of patterns in computer science is well established, research into ontology design patterns is a fairly recent development. We believe that it is important to develop an overview of the state of research in this new field, in order to stake out possibilities for future research and in order to provide an introduction for researchers new to the topic. This paper presents a systematic literature review of all papers published at the three large semantic web conferences and their associated workshops in the last five years. Our findings indicate among other things that a lot of papers in this field are lacking in empirical validation, that ontology design patterns tend to be one of the main focuses of papers that mention them, and that although research on using patterns is being performed, studying patterns as artifacts of their own is less common.

Keywords: Ontology design patterns, literature review, content classification

1 Introduction

The use of patterns in computer science-related work already has some tradition. Since the seminal work on software design patterns by the "gang of four" [8], many pattern approaches have been proposed and found useful for capturing engineering practices or documenting organizational knowledge. Examples can be found for basically all phases of software engineering, such as analysis patterns [6], data model patterns [10], software architecture patterns [7, 4] or test patterns. The pattern idea has been adapted in other areas of computer science, such as workflow patterns [5], groupware patterns [12] or patterns for specific programming languages [1].

Ontology design patterns are a rather new development. Ontology design patterns in its current interpretation was introduced by Gangemi [9] and by Blomqvist and Sandkuhl [3]. Blomqvist defines the term as *"a set of ontological elements, structures or construction principles that solve a clearly defined particular modeling problem"* [2]. Despite quite a few well-defined ontology construction methods and a number of reusable ontologies offered on the Internet,

efficient ontology development continues to be a challenge, since this still requires both a lot of domain knowledge/experience and knowledge of the underlying logical theory. Ontology Design Patterns (ODP) are considered a promising contribution to this challenge. They are considered as encodings of best practices, which help to reduce the need for extensive experience when developing ontologies.

When a new research area is emerging, as is the case for ontology design patterns, both academia and industry are interested in the state of affairs, but for different reasons. Academia's interest often relates to identifying the open problems and most pressing challenges to be explored and solved. Industry's prime interest is often in new technologies promising better operational efficiency and effectiveness, or competitive advantages for particular products or services. In order to decide which technologies qualify, they need to understand the choices available, their advantages and disadvantages, and potentials risks. This paper presents a literature survey intending to identify existing research in ontology design patterns, structure this research, and evaluate the way it has been tested and validated.

The remaining part of the paper is structured as follows: after an introduction to the research questions and description of study design (Sections 2 and 3 respectively), the paper presents some key results of the survey (Section 4). Section 5 discusses threats to validity and analyses the results, attempting to answer the research questions posed, and pointing out possibilities for future work.

2 Research Questions

The point of performing a literature survey is of course to get a handle on the current state of affairs within the topic area. But what does this really boil down to, what data should we gather and which questions should we ask? We turn to another information gathering profession for guidance, by adopting and adapting the Five Ws (and one H) formula for gathering information from the journalism field. This is a set of questions that need to be answered for any piece of reporting to be considered complete. The questions to pose are Who, What, When, Where, Why, and How.

Since, in the case of academic research, the Who and the Where often converge (few scholars work independently of research institutions) and since we do not want to perform reviews of individual scholars, the Who question is dropped from the list. The Why question is also left out - while it is indeed an important question to answer (why ontology pattern research is done at all), it seems out of scope of a survey such as this one. We are left with four questions to answer. We map these questions to concrete answerable research questions for the survey:

1. What kind of research on ontology patterns is being performed? (What)
2. How has research in the field developed over time? (When)
3. Where is ontology pattern research performed? (Where)
4. How is research in ontology patterns being done? (How?)

3 Method

The method used to perform the literature review is illustrated in Fig. 1. It consisted of first finding the relevant research papers (i.e. papers that discuss ontology patterns), then classifying and sorting them based on various metrics and measures in order to obtain the data needed to answer the research questions. The following sections discuss each of these steps in more detail.

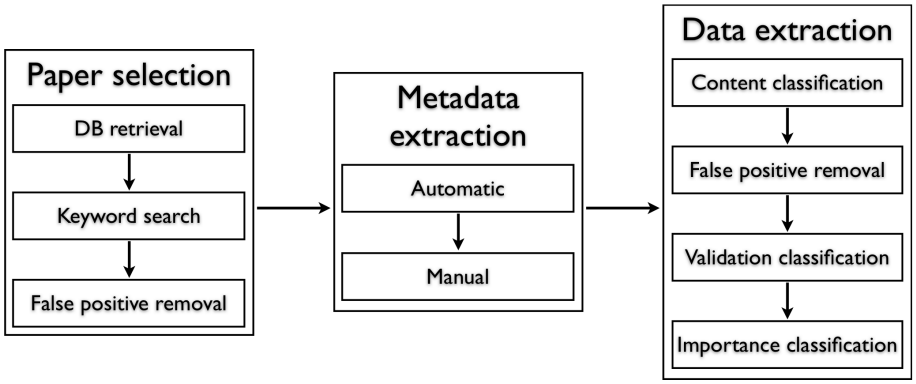

Fig. 1. Method overview.

3.1 Paper Selection

For the purpose of this survey, we have studied conference and workshop proceedings of ISWC, ASWC and ESWC from the last five years (2005-2009). The reasons for choosing these delimitations are as follows:

- The conferences in question are three of the most well-established and prestigious semantic web conferences in the academia. Papers that have been accepted to them are therefore likely to be of a high quality and representative of the general direction in which the semantic web research field is heading.
- As mentioned in section 1, the current interpretation of ontology design patterns was introduced in [9] and [3], both of which were published in 2005. We thus consider this a natural starting year for the survey. 2009 was selected as the ending year for the simple reason that the 2010 conferences have not all been held at the time of writing.
- After initially performing the selection process detailed below with only the main conference proceedings as source, it was found that the number of papers returned were rather few. In order to gain more data, the scope was widened to include the workshop proceedings also, on the intuition that

the workshops while being narrower in focus may still be relevant markers of trends and directions in ontology pattern research (see for instance the WOP workshops that deal specifically with ontology patterns).

The proceedings were retrieved from various Internet databases including SpringerLink[1] and CEUR-WS[2]. A total of 2462 papers were retrieved, divided into 861 conference papers and 1601 workshop papers. Unfortunately, not all workshop proceedings could be located and retrieved, leaving the following workshops' proceedings missing from the survey:

ISWC
SemUbiCare-2007
ESWC
Framework 6 Project Collaboration for the Future Semantic Web-2005
ASWC
All workshops from ASWC 2006
All workshops except DIST from ASWC 2008
All workshops from ASWC 2009

In order to find the subset of papers dealing with ontology patterns, all of these papers were subjected to a full-text search. All papers containing the phrases *ontology patterns* or *ontology **word** patterns* (***word*** denoting any one single word) were selected for further analysis. Thereafter, in order to weed out false positives, papers mentioning patterns only in the reference list were removed from the dataset. This left us with a total of 54 papers, 19 from conferences and 35 from workshops.

3.2 Metadata Extraction

The following metadata posts were determined to be useful in answering the research questions:

- Publication year - necessary (but not sufficient) for answering RQ 2
- Author provenance (research institution) - sufficiently answers RQ 3
- Author count - necessary (but not sufficient) for answering RQ 4
- Institution count - necessary (but not sufficient) for answering RQ 4

These pieces of information were retrieved from the dataset. The publication year was retrieved automatically by way of database queries and web spidering scripts when downloading the source material, whereas the research institutions and institution/author counts were ascertained by studying the papers manually.

Research institutions were counted and ranked by the number of mentions they received in the headers of papers in the dataset. However, due to time constraints, further study of the specific organizational structure of the various

[1] http://www.springerlink.com
[2] http://ceur-ws.org/

institutions was not performed. Consequently, different departments of the same university were counted as belonging to the same unit, as were different branches of a company or other research organization. This gives us a course-grained view of what universities and organizations are involved in this type of research.

3.3 Data Extraction

Answering research questions 1 and 4 (the What and How questions) required more detailed reading and analysis of the papers. We determined that the What question could most accurately be answered by categorizing the papers based on how they contribute to or make use of ontology pattern research. This categorization is described later in this section and in Table 1. The How question, due to its rather fuzzy nature, is harder to answer in a concrete manner. We decided to focus on three measurable aspects: (1) how scientists cooperate, (2) how the theories they propose are validated and tested, and (3) how central ontology patterns are to the content of the studied papers (whether they are a core part of the performed research or merely used in passing). Aspect 1 can be ascertained by authorship metadata, see section 3.2. Aspects 2 and 3 are discussed in the following sections.

Table 1. Content categories and definitions.

Category	Definition
New pattern presentation	The paper presents a new ontology pattern.
Pattern usage method	The paper presents a new general approach or method of using patterns to achieve some goal(s).
Pattern creation method	The paper presents a new approach for isolating or creating ontology design patterns (including re-engineering from other knowledge representation forms).
Patterns used	The paper describes a case where patterns have been used for achieving some goal(s)
Evaluation	The paper focuses on evaluating patterns or pattern usage/creation methods
Pattern typology	The paper discusses or suggests different types or groupings of patterns.
Pattern features	The paper discusses specific features of ontology patterns. This includes features of not only the reusable reference implementation, but also the documentation and metadata associated with it.
Pattern identification	The paper deals with finding instances of patterns in an existing ontology.
Pattern languages	The paper discusses languages or formalisms for representing or displaying patterns.
Antipatterns	The paper concerns antipatterns or worst practices.
Not relevant	The paper is a false positive - it does not deal with ontology pattern research, but only mentions the term in passing.

Content Classification We read and tagged the papers with one or more labels categorizing how they related to or made use of ontology patterns. Since we did not know beforehand what categories would best describe the collected material, the labels used could not be decided ahead of time. Instead, the label were selected in an exploratory fashion as the readings of the papers progressed. However, to try to ensure as unbiased a classification as possible, each label was paired with a definition covering papers belonging to the category. The labels and definitions are listed in Table 1. The papers were studied twice, once before and once after deciding upon the categories.

In tagging the papers, we attempted to err on the side of caution. For instance, if a pattern is used or mentioned in a paper without reference to original creator, we have not assumed that the author of the paper has created this pattern for this particular project. Rather, new patterns or new methods need to be explicitly defined as new contributions before we add any of the relevant tags to the paper.

The papers that were classified as belonging to the *Not relevant* category were then pruned from the dataset, leaving 47 papers, of which 16 were conference papers and 31 workshop papers.

Validation Classification In order to survey how ontology pattern research is validated, two procedures were followed. To begin with, the papers were categorized according to in what manner validation or testing of the proposed ideas and theories had been performed. For this purpose, the following four categories were used:

- No validation - there is no mention of any validation of the ideas presented.
- Anecdotal validation - the paper mentions that the research has been validated by use in an experiment or in a project but it provides no detail on how this validation was performed or its results.
- Validation by example - one or more examples are presented in the text, validating the concepts presented in a theoretical manner.
- Empirical validation - some sort of experimental procedure or case study has been performed.

Each paper was assigned to one validation category only. In the cases where a paper matched more than one category, the category mapping to a higher level of validation was selected, i.e. empiricism trumps example which in turn trumps anecdote.

Note that there is a difference between examples used in passing in text to motivate some particular design choice or to illustrate syntax, and examples made explicit and presented as validation or test of theory. The latter is grounds for inclusion in the category *Validation by example*, the former is not. In the studied papers it was also common to take one example or use case and follow along with it from the beginning of the paper (where it occurs as problem description) to the end (presenting a solution to said problem). This is considered

to be thorough enough of an example for such a paper to be included in the *Validation by example* category.

Having categorized the papers by validation technique, a further study of validation quality was performed against the papers categorized as belonging to the *Empirical validation* group. For this study we made use of some of the metrics and corresponding measurement criteria developed in [11]:

- Context description quality - the degree to which the context (organization where the evaluation is performed, type of project, project goals, etc.) of the study is described.
- Study design description quality - the degree to which the design (the variables measured, the treatments, the control, etc.) of the study is described.
- Validity description quality - the degree to which the validity (claims of generalizability, sources of error etc.) of the study is discussed.

ODP Importance Classification In order to learn how important the use of or research into ontology patterns is to each particular paper, we studied in what section of the paper that patterns were mentioned. The intuition was that this information would give a crude indication as to whether ontology patterns were considered an essential core part of the research (warranting inclusion in the title or abstract) or not. Title inclusion indicate greater importance than abstract inclusion, in turn indicating greater importance than mere inclusion in the body text. For this measure, terms such as "Knowledge patterns", "Semantic patterns", or just plain "patterns" were also considered acceptable synonyms for ontology patterns, provided they were used in a manner indicating a link to ontologies and the use of such patterns in an ontology engineering context.

4 Results

The processes described in section 3 provided us with a large amount of data to analyze, a subset of which is presented in this chapter. The full dataset is too large to include in full, but is available on demand.

Table 2. ODP papers by year

Year	Conferences	Workshops
2005	2	2
2006	2	2
2007	7	3
2008	1	4
2009	4	20[1]

[1] 15 of which are from WOP 2009

Table 3. Institutes by paper count

Institute name	Conference	Workshop
ISTC-CNR	4	2
U. of Economics, Prague	2	4
U. Politecnica de Madrid	1	5
U. of Karlsruhe	0	4
Jönköping U.	3	0
Salzburg Research GmbH	2	1
U. of Innsbruck	1	2
U. of Manchester	1	2
U. of Sheffield	1	2

Table 2 presents the number of papers in the dataset indexed by the year they were published, the idea being that this might give an indication as to whether research interest in the field is expanding. Table 3 lists the research organizations most often listed as affiliations of authors in the dataset (sorted by summarized publication count and alphabetically). The full list is considerably longer at 49 entries in total, but is for brevity here limited to organizations with three or more mentions.

Table 4. Classification of the reviewed papers' connection to ODPs.

Classification	Conferences	Workshops
Antipatterns	0	2
Evaluation	0	2
New pattern presented	3	12
Pattern creation methods	1	1
Pattern features	1	5
Pattern identification	0	2
Pattern languages	1	4
Pattern usage method	4	11
Pattern typology	0	3
Patterns used	8	6

Table 4 contains the results of the process described in section 3.3, that is, the labels denoting categories of pattern-related research and the number of papers tagged with each such label. The results are divided into columns for the conference papers and workshop papers.

Table 5. Validation levels of reviewed papers.

Source	No validation	Anecdote	Example	Empiricism
Conferences	3	2	5	6
Workshops	3	2	19	7

Tables 5 and 6 present the results of the validation classification performed in section 3.3, i.e. how the results were validated and, in the case of empirical procedures being used for this purpose, how well the experiments or case studies were described. Table 7 presents the institution counts of the papers in the dataset. Table 8, finally, shows the result of the ODP importance classification in section 3.3, i.e. in what parts of the papers that the topic of patterns were addressed.

Table 6. Quality of empirical validations.

Quality indicator	Weak	Medium	Strong
Conference papers			
Context description	4	1	1
Study design description	0	3	3
Validity description	5	1	0
Workshop papers			
Context description	5	2	0
Study design description	1	3	3
Validity description	6	1	0

Table 7. Institution counts

Institutions	Conferences	Workshops
1	12	16
2	2	10
3	2	5

5 Analysis and Discussion

In this section we discuss some limitations to the generalizability of the survey and sources of error. We then analyze the data resulting from the survey, attempting to answer the research questions posed in section 2.

5.1 Sources of Error

There are of course limitations to the validity of this survey. To begin with, the selection of source material (ISWC/ASWC/ESWC 2005-2009) may give a limited view of the developments within the field as a whole. It is our belief that these three conferences are important and well-known enough for this selection to give a good overview of the general state of ontology pattern research, but we cannot guarantee that key influential work has not been presented elsewhere, thus falling below our radar. The reader should therefore keep this limitation of generalizability in mind when reading our results.

Furthermore, in the selection process described in section 3.1, we do not check for false negatives. This means that there may be papers within the source material which are not matched in the full text search and thus not selected for further analysis, even though they actually deal with ontology patterns. In order

Table 8. ODP importance classification of reviewed papers.

Group	Conferences	Workshops	Workshops (w/o WOP)
Title match	7	22	8
Abstract match	3	3	2
Body match	6	6	6

to study how common such false negatives are, a random set of discarded papers would need to be read manually. This is something that due to time constraints, we were unable to do. If the false negatives are evenly distributed over the various metrics that we have studied, the impact of missing these papers in the analysis might not be that great, but without further looking into this we cannot say either way.

Another possible error lies with the fact that we have been unable to gather all of the data from the ASWC workshops. This means that it is very possible that Asian universities are ranked lower than they actually deserve in the analysis of where research is done (RQ3).

Finally, in Section 3.3 there are at least two threats to validity. To begin with, we read and classify papers. This process is of course, by its very nature, prone to bias since judgement as to what categories a certain paper matches is not always entirely clear cut. We have tried to limit this threat as far as possible by defining fairly rigid categories, but the reader should keep in mind that this is not a 100 % objective measure by any means. Secondly, since we selected the categories on basis of the papers that we have read, it is likely that we have not exhausted the whole set of possible categorizations of ontology pattern research. In other words, there may be categories of research that are not present in our findings, for the very reason that they are not present in our data.

5.2 What kind of research on ontology patterns is being performed? (RQ1)

The results indicate that while patterns are used in various different ways in research and new patterns are being presented, there is quite a lot less work being done on how to formalize the creation and/or isolation of patterns. This, in our view, means one of two things. It could indicate that the best ways of creating and finding patterns have been established, and that there is thus little need for more research to be done in those areas. However, due to the youth of the field, we believe this to be less likely.

Instead, we believe it is more likely that the results indicate a lack of sufficient research in these areas. This situation could be problematic if it indicates that the patterns that are used are not firmly grounded in theory or practice. If one adds to this the fact that relatively little work is being done on pattern evaluation, the overall impression is that patterns are being presented and used as tools, but are not being sufficiently studied as artifacts of their own.

Another area that appears to be understudied is antipatterns, "worst practices" or common mistakes. We theorize that this may be because finding such antipatterns necessitates empirical study, which as we report in section 5.5, appears to be less common in the papers we have studied.

The distribution of research categories seems to be rather consistent between the set of conference papers and the set of workshop papers, with exception for the categories *Patterns used* and *Pattern creation methods*. The latter could be a simple statistical anomaly (as we've mentioned there are few papers dealing

with this topic, for which reason small discrepancies stand out more). The former seems a bit more notable however. We have no theory explaining this finding.

5.3 How has research in the field developed over time? (RQ2)

We can see that there is a much larger number of papers matching our search criteria published in 2009 than in 2005, in large part due to the first Workshop on Ontology Patterns being held in 2009. If we remove the WOP papers, we still see a growth in volume.

One interesting note in relation to RQ1 is that all of the work on pattern identification and pattern creation methods that we found was published in 2009. Furthermore, the papers dealing with pattern evaluation are all from 2008-2009, possibly indicating that researchers have noticed these gaps in theory and are attempting to do something about them.

Unfortunately both of these observations are based on so few papers and such a short period of time that it would be very difficult to claim them as a general trends or make predictions based on them.

5.4 Where is ontology pattern research performed? (RQ3)

Keeping in mind the possibility of errors in selection mentioned in section 5.1, our results indicate that ODP work is primarily taking place at European institutions. In Table 3 we can see that all of the top nine institutions publishing more than two papers are located in mainland Europe and the UK. However, even if we include all of the institutions that have two publications in the dataset, we still do not find any non-european organizations. As a matter of fact, out of a total of 49 institutions that had published, only four were located outside of Europe. Out of these four, three were based in the USA and one in New Zealand.

We can also see that while the dataset is dominated by research originating at universities, there are also a number of private corporations, research foundations, and other types of non-university organizations that work in the field. Out of the 49 institutions found, 17 were such non-university organizations (approximately 35 %). These proportions are slightly lower when taking into account the number of publications per institution, with the non-university organizations netting 31 % of all institutional mentions in the dataset.

5.5 How is research in ontology patterns being done? (RQ4)

Studying academic cooperation, one easily accessible metric is the number of authors per paper. However, author counts on their own can be misleading. In some academic cultures students' advisors get authorship, while in others they do not. We instead look at both author counts and institution counts which gives a better indication of actual research cooperation.

Out of a total of 47 papers, 19 (just over 40 %) list more than one affiliated institution and 7 (just under 15 %) list three institutions. These figures, however,

include papers written by only one author. If we look at the subset of 40 papers that were written by more than one author, the figures are 47.5 % and 17.5 % respectively. No matter which way you slice it, these numbers in our opinion indicate a quite healthy degree of cooperation between research institutions in the field.

Interestingly, there seems to be a difference between work published at workshops and work published at the main conferences - of the former, 51.6 % are credited to single institutions, whereas the of the latter, 75 % are. This possibly indicates that the prestige and/or difficulty associated with the higher barrier of entry to full conferences cause researchers to keep such papers "in-house" to a larger degree.

With regards to the validation and testing of ODP research, we find that there may be some work to be done. Nearly one third of the papers published at full conferences contain no validation or only anecdotal validation of the presented work. Another near-third validates the work via examples, but provide no real-world testing to ensure validity. For workshop papers, a smaller proportion of the papers have no validation or anecdotal validation, but on the other hand, a much larger proportion validate only through example. Of the papers that do contain empirical testing, it is uncommon to see discussions on the limits of validity of said testing. This situation may be somewhat problematic. Though not all types of research invite the opportunity to perform experiments or case studies, nor actually require them, we find quite a few papers that could have benefitted from a more thorough testing procedure.

Finally, looking at how central ODPs are to the content of the papers, we find that the most common situation is actually that patterns are mentioned already in the title (see Table 8), indicating that they are quite central to the papers. This remains the case even when we filter out the WOP papers which naturally skew the results. We have no explanation for this unexpected result. One possibility is that this indicates that ontology design patterns are primarily used within the ODP research community and thus written about by people who consider them to be important enough to warrant inclusion in the title.

5.6 Future Work

We can imagine followups to this study a few years down the line, studying the developments on a longer time scale, or using a wider set of source conferences (including for instance EKAW or KCAP). Also, an in-depth analysis of the individual papers could be performed, studying the citation counts of the papers and cross referencing this against the other metrics studied, in order to validate how well the latter map against the real world qualities that give a paper a high citation count.

Another aspect that we would like to explore when time and resources permit is the use of patterns within industry. It is certainly possible that there are patterns and best practices that are established outside of the academia, and if this is the case, it is not too far-fetched to asssume that such patterns are more grounded in empirical knowledge than those discussed in this paper.

With regard to the more important question of future research opportunities in the ODP field we suggest that further work is needed on:

- methods of evaluating the efficiency and effectiveness of ODP,
- developing ODPs for particular usages,
- isolating ODPs from existing ontologies or other information artifacts.

We would also like to see, where possible, a greater focus on evaluating academic results in this field in an empirical manner. This would aid in firmly establishing the credibility and maturity of the ODP research field, which we believe to be necessary if ontology patterns are to be established as a viable solution to the complexity challenges of ontology development, in both the greater research community and in industry.

References

1. Beck, K.: Smalltalk Best Practice Patterns. Volume 1: Coding. Prentice Hall, Englewood Cliffs, NJ (1997)
2. Blomqvist, E.: Semi-automatic ontology construction based on patterns. Ph.D. thesis, Linköpings universitet (2009)
3. Blomqvist, E., Sandkuhl, K.: Patterns in ontology engineering: Classification of ontology patterns. In: Proceedings of the 7th International Conference on Enterprise Information Systems. pp. 413–416 (2005)
4. Buschmann, F., Henney, K., Schmidt, D.: Pattern-oriented software architecture: On patterns and pattern languages. John Wiley & Sons Inc (2007)
5. van Der Aalst, W., Ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and parallel databases 14(1), 5–51 (2003)
6. Fowler, M.: Analysis Patterns: reusable object models. Addison-Wesley (1997)
7. Fowler, M.: Patterns of enterprise application architecture. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA (2002)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-wesley Reading, MA (1995)
9. Gangemi, A.: Ontology design patterns for semantic web content. In: The Semantic Web–ISWC 2005. pp. 262–276. Springer (2005)
10. Hay, D.: Data Model Patterns: Conventions of Thought. Dorset House (1996)
11. Ivarsson, M., Gorschek, T.: Technology transfer decision support in requirements engineering research: a systematic review of REj. Requirements engineering 14(3), 155–175 (2009)
12. Schümmer, T., Lukosch, S.: Patterns for computer-mediated interaction. Wiley (2007)

Adapting Ontologies to Content Patterns using Transformation Patterns

Vojtěch Svátek, Ondřej Šváb-Zamazal, and Miroslav Vacura

Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{svatek|ondrej.zamazal|vacuram}@vse.cz

Abstract. Ontology content patterns are meant to be used not only for new ontologies but also for reengineering of existing ontologies. However, the modelling style of such ontologies often differs from the best-practice pattern that is to be imported to their root portions, which makes the integration of the two models time-consuming and error-prone. We explore how the recently developed PatOMat transformation framework could be applied to ease the adaptation of 'legacy' ontologies to a widely used content pattern from the OntologyDesignPatterns.org library. We also investigate the link between transformation choices and logical patterns as those earlier proposed by the W3C SWBPD Group.

1 Introduction

Ontology content patterns [4] are nowadays considered as a central artifact for promoting best practices and supporting shareability in ontology design. Although, ideally, content patterns (CPs, for brevity) should be taken into account from the very start of the design process, it is a common situation that the authors of the ontology are, at the onset, either unaware of the existence of CPs at all or too novice to choose the right one. The CPs then only enter the design process at a later phase, when at least a prototype of the ontology already exists, and their adoption has the nature of ontology reengineering.

In the easiest setting, generic CPs can be (as kind-of foundational ontology fragments) 'passed under' the current root concepts of the ontology. However, as the same conceptualisation can be expressed using different *modelling styles* in a language such as OWL,[1] the need for style transformations may often arise. A straightforward example of such transformation is change from 'class-centric' to 'property-centric' style, or vice versa, considering that the same notion can be modelled as a class (e.g. 'Purchase') or an object property ('bought_from').

In our previous work on the PatOMat project,[2] we already addressed the general need for style transformation in ontological engineering. A general *framework*, a simple transformation pattern *language*, and a set of RESTful *services*

[1] In our framework we implicitly assume the use of OWL (possibly in version OWL 2 [6]) as ontology language.

[2] http://patomat.vse.cz

(relying on external tools such as OPPL and OWL-API, see Section 2) have been developed, and thoroughly exemplified for an *ontology matching* scenario in [11]: having two ontologies to be matched, we can transform the modelling style of the one so as to make automated matching to the other easier.

The new scenario in this paper is rather one related to *ontology import*. However, compared to the generic scenario of either matching or importing an arbitrary ontology into another one, we consider here a particular, small and widely reusable ontological component—a content pattern—to which an existing 'provisional' ontology is adapted in order to gain both in rigour and comprehensibility.

The rest of the paper is structured as follows. Section 2 briefly reviews the *PatOMat* framework, transformation language and implemented prototype services, as described in [11]. Section 3 summarises the ontology matching scenario from [11], and introduces the new scenario of importing a CP into a prototype ontology. Section 4 presents the specific input setting of our case study: the *AgentRole* pattern from the *OntologyDesignPatterns.org* library, and the *ConfOf* ontology from the *OntoFarm* collection. Section 5 discusses the dimensions of ontology transformation wrt. *AgentRole*, which relate to the source pattern, target pattern, and additional axioms in the source ontology. Section 6 shows an XML serialisation of a (selected) relevant transformation pattern. Finally, Section 7 surveys some related work, and Section 8 wraps up the paper.

2 Overview of the *PatOMat* Transformation Framework

The central notion in the *PatOMat* framework[3] is that of *transformation pattern* (TP). A TP contains two *ontology patterns* (the source OP and the target OP) and the description of transformation betweem them, called pattern transformation (PT). The representation of OPs is based on the OWL 2 DL profile. However, while an OWL ontology refers to particular entities, e.g. to class *Person*, in the patterns we generally use *placeholders*. Entities are specified (i.e. placeholders are instantiated) at the time of instantiation of a pattern. An OP consists of *entity declarations*, *axioms*, and *naming detection patterns*; the last capture 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*. Transformation links are either *logical equivalence relationships* or *extralogical relationships* holding between two entities of different type. Naming transformation patterns serve for generating new names for old or newly created 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. The framework prototype implementation consists of three core services:[4]

– The *OntologyPatternDetection* service takes the transformation pattern and a particular original ontology on input, and returns the binding of entity

[3] [11] provides more details about the framework, and at `http://owl.vse.cz:8080/tutorial/` there is a fully-fledged tutorial for the current version.

[4] All accessible via the web interface at `http://owl.vse.cz:8080/`.

placeholders on output, in XML. The structural/logical aspect is captured in the structure of an automatically generated SPARQL query; the naming aspect is dealt with based on its description within the source pattern.
- 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 based on OPPL [2] and partly on our specific implementation over OWL-API.[5] Currently we use OPPL for the operations on *axioms* and for *adding entities*, and OWL-API for *re/naming* entities according to naming transformation patterns and for adding *OWL annotations*. As far as detection is concerned, the SELECT part of OPPL could be used to some extent; our naming constraints are however out of the scope of OPPL. Furthermore, in contrast to OPPL, we *decompose* the process of transformation into parts, which enables user intervention within the whole workflow.

3 *PatOMat* in Use: Matching vs. CP Importing Scenario

As mentioned in the Introduction, the initially considered scenario for pattern-based ontology transformation was the *ontology matching scenario*, schematically depicted in Fig. 1 (the 'structural changes' shown are merely illustrative and don't claim to correspond to meaningful transformations). The transformation step here precedes the actual matching step in the whole workflow. A given ontology (possibly just a fragment thereof), which is to be matched to a second ontology, is first matched to the source OP of a TP; the choice of the fragment as well as of the TP is however guided by an analysis of the second ontology (such that the target OP of the TP should use the same modelling style as the second ontology). The transformed ontology is built in the style of the target OP of the TP. This makes the subsequent matching to the second ontology easier; e.g. simple and fast string matching methods can be used instead of sophisticated and fragile matching methods.

The *content pattern import scenario* (adapting a prototype ontology, via transformation, to the style of a CP), depicted in Fig. 2, differs from the previous one in the overall workflow, in the sense that the import literally precedes the transformation. Namely, in the first step, the CP is included in the ontology as a separate structure, merely subordinated to *owl:Thing*. Then the transformation takes place; however, only TPs specifically tailored to the given CP are considered. In the transformed ontology,[6] shaped to the style of the target OP of the TP, the CP is already integrated into the ontology, as part of its root structure.

[5] `http://owlapi.sourceforge.net/`

[6] For illustration, Fig. 2 also contains an entity B, which is not matched by the TP and thus implicitly copied to the transformed ontology.

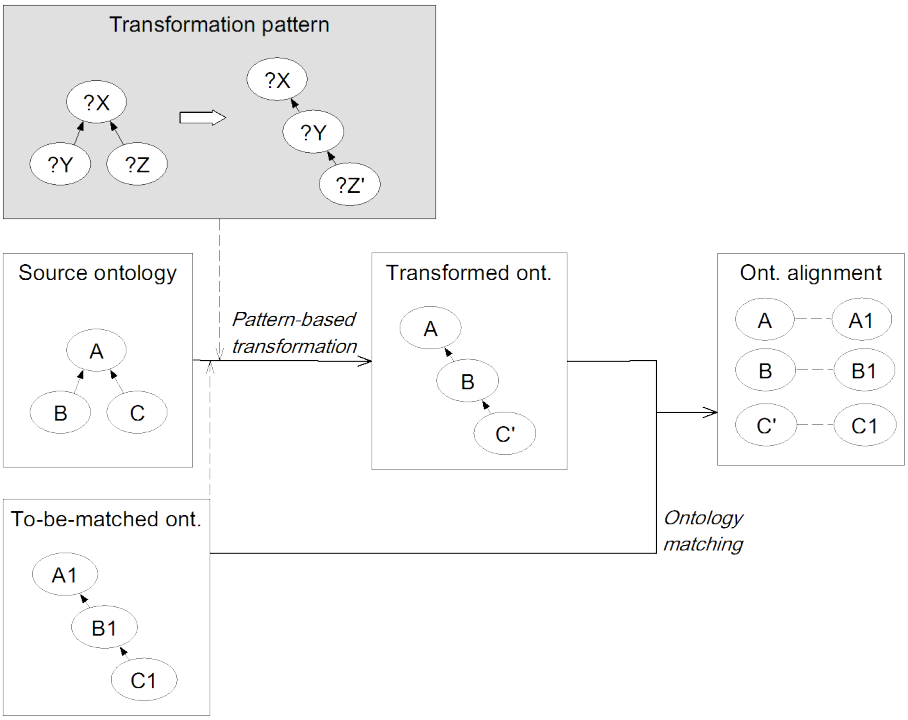

Fig. 1. Schematic depiction of transformation for ontology matching

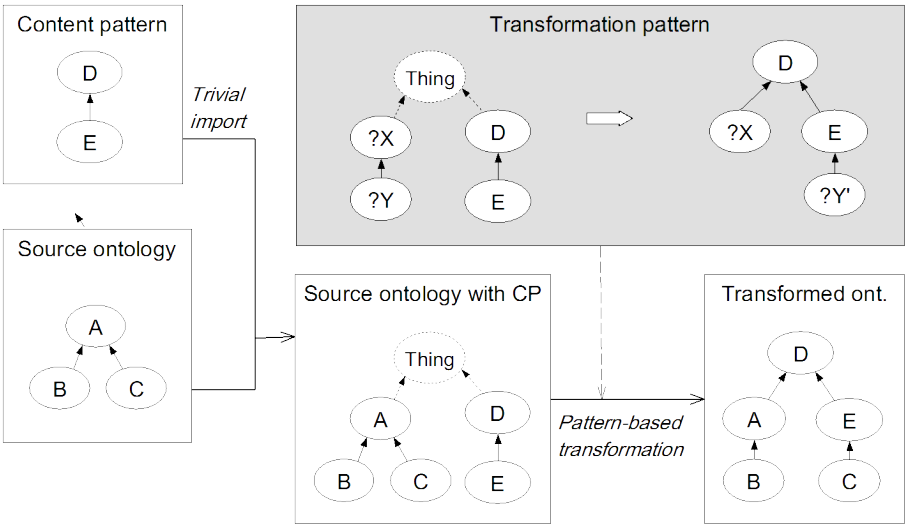

Fig. 2. Schematic depiction of transformation for CP import

Even if a part of the input to the transformation, the content pattern, is fixed, there is potentially great variability in the ways different ontologies can be adapted to the pattern. The variability goes along (at least) three different dimensions: the style of the *source pattern* occurrence proper, the style of the *target pattern* (apart from the imported content pattern itself), and the existence of *additional axioms* external to the source pattern that refer to entities from the pattern. We exemplify each of these dimensions in Section 5.

4 Input Settings for the Import Scenario Case Study

4.1 Role Modelling Approaches and *AgentRole* Pattern

The notion of *role*[7] has repeatedly appeared in the history of knowledge modelling. We will not examine this phenomenon in depth here (for thorough discussion refer e.g. to [3], Ch.7), but will only present the most obvious modelling options in the context of OWL, partially borrowed from [10].

For expressing general notions in OWL, essentially, one has two atomic entity types available. An ontology designer not deeply acquainted with the notion of ontological role will typically model a role implicitly, as

- a *class* that is a subclass of a class expressing a natural concept (e.g. *Teacher* as subclass of *Person*), such that the 'role' class is endowed with restrictions over one or more properties (e.g. an instance of *Teacher* has to be *teacherOf* some *Student* and *teacherAt* some *University*), or
- just as one or multiple separate *properties* (such as *teacherOf*, *teacherAt*) representing aspects of the role.

It should be mentioned that Sunagawa [10] also suggests a sophisticated OWL pattern for capturing all important aspects of role playing: role concepts (i.e. roles proper), natural concepts (that play roles), and role holders (that hold roles but inherit properties from natural concepts). We do not however consider this pattern here, as it is very unlikely for it to have been engineered 'just by chance' in the kind of prototype ontologies we focus on in our approach.

The wiki-based web portal at *OntologyDesignPatterns.org* contains a vast number of patterns of various types, all related to ontology design and exploitation. The most represented category is that of content patterns as immediately reusable ontology building blocks: there are 80 such patterns (though none yet certified) at the time of writing this paper. As the library aims to lower the threshold for even less experienced ontology designers, most CPs are relatively simple, yet grounded in well-designed foundational ontologies such as the lightweight version of DOLCE.[8] The *AgentRole* pattern, depicted in UML-like notation in Fig. 3, is an example of such a simple content pattern leveraging

[7] This general notion should not be confused with the term 'role' used for binary relation in description logics (as formal underpinning of OWL).

[8] `http://www.loa-cnr.it/ontologies/DUL.owl`

on the modular structure of imports. The *AgentRole* pattern, displayed with its elements in solid, specializes the *ObjectRole* pattern (entities from the 'or' namespace), which in turn specializes the *Classification* pattern (entities from the 'class' namespace), both displayed with their elements in dashed. Following from the most generic model, the *Classification* pattern merely allows to state the relationship between an entity and the concept to which this entity is somehow classified; this corresponds to an informal 'reification' of the *subClassOf* relationship. *ObjectRole*, in turn, already deals with role playing, understood as a specific type of classification;[9] entities playing roles can be any objects (i.e. no arbitrary things such as properties any more). Finally, *AgentRole* introduces an even more specific class of such role-playing objects, called *Agent*, which is declared as disjoint with class Role.

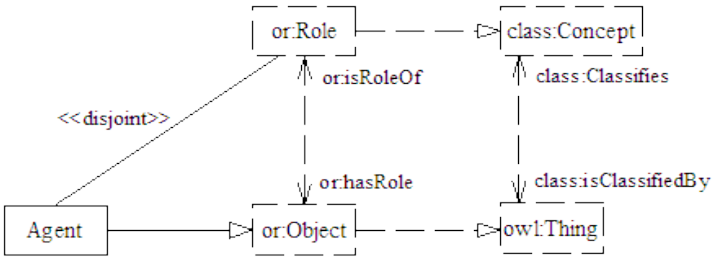

Fig. 3. *AgentRole* pattern and its imports

4.2 Example Input Ontology

As core example of input ontology to be adapted, we will use an existing ontology from the *OntoFarm*[10] collection, the *ConfOf* ontology. It models the domain of 'organizing conferences' from the point of view of a particular review (and other activity) support software: the *ConfTool*.[11] We will however also discuss patterns alternative to those used in *ConfOf*.

One 'role-playing' fragment in *ConfOf* is depicted in Fig. 4. The notion of authorship is modelled as the *Author* class being subclass of *Person*. The *Author* class has a pair of existential and universal restriction over the *writes* property, and also appears in the domain of this property and in the range of its inverse, *writtenBy*.

[9] There is a subproperty relationship (not depicted in the diagram) between *isRoleOf* and *Classifies*, and *hasRole* and *isClassifiedBy*, respectively.

[10] For an overview on the *OntoFarm* project see `http://nb.vse.cz/~svatek/ontofarm.html`.

[11] `http://www.conftool.net`

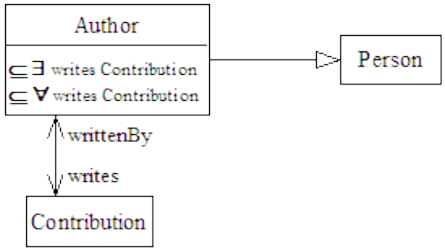

Fig. 4. Fragment of *ConfOf* ontology dealing with authorship

5 Transformation Process Alternatives

In this section we will demonstrate the aspects of the 'CP import' scenario of pattern-based ontology transformation, outlined in Section 3, on the concrete setting presented in Section 4.

5.1 Source Pattern

As natural candidates for the source pattern we can consider the simple role patterns from Section 4.1: we can call them 'class-oriented' and 'property-oriented' pattern, respectively. Obviously, *ConfOf* sticks to the 'class-oriented' role pattern. Another ontology could however, for example, avoid the Author class and model the author role merely in terms of properties such as *writes* and *written* by (or e.g. *authorOf* and *hasAuthor*), with domain/range set to *Person*, this leading to the 'property-oriented' pattern.

In addition, due to the above-mentioned sequencing of operations, the *content pattern* is already present in the ontology when the ontology is submitted to transformation, and thus has to be a part of the source pattern. However, it is unconnected to the rest of the ontology (and source pattern) yet.

5.2 Target Pattern

Obviously, the content pattern is transferred to the target pattern without change. However, the way the rest of the source pattern occurrence is shaped and linked to the content pattern may vary.

Note that, in the particular context of *AgentRole* pattern and the class-oriented role modelling style of *ConfOf*, the transformation of the notion of 'author' from a (seemingly) natural concept to a role amounts to transition from the 'instance of' relationship (being a language primitive in OWL DL) to the *hasRole* relationship (i.e. object property). By consequence, the fact of a person having the author role, which was previously expressed as e.g. "John rdf:type Author", now has to connect the individual John to some 'author role' entity through the *hasRole* property. Assuming we formalise the author role as class in

the target pattern, we arrive to an instance of the "defining classes as property values" problem, treated as a logical pattern in [7].

From the set of five 'modelling approaches' (actual logical patterns, in fact) of this published pattern we will only consider those expressible in OWL DL. This eliminates "Approach 1: Use classes directly as property values". The remaining are, in turn (in the original terminology of [7]):

- Approach 2: Create special instances of the class to be used as property values
- Approach 3: Create a parallel hierarchy of instances as property values
- Approach 4: Create a special restriction in lieu of using a specific value
- Approach 5: Use classes directly as annotation property values

We will not analyze the pros and cons of different approaches wrt. particular situations, as in [7]. Instead we will try to reveal important aspects of transformation of the *ConfOf* fragment to the given approach. It should be noted that the application of the approaches is not always as obvious as in the "books-about-animals" example used through [7], as the nature of the underlying *dc:subject* property is somewhat different from the *hasRole* property in our example; we occasionally comment on such differences.

Transformation to Approach 2. It may lead, assuming some naming transformations, to the situation depicted in Fig. 5 (we omit the namespace prefixes and don't distinguish imports, for better readability). The *Author* class was removed, while there is a new class, with the same name but different meaning, subordinated to *Role*; there is now also now a distinguished instance of the latter class, called *AuthorRole*[12] (in rounded rectangle) as part of the ontology. While populating the transformed ontology, an instance of *Person* can be connected by the *hasRole* property with the *AuthorRole* individual.

ConfOf models authorship in a simple way such that there are no subclasses of *Author*. However, if there were such subclasses (e.g. *PosterAuthor*), they would be transformed to subclasses of the new *Author* class; each such class would also have a corresponding individual (e.g. *PosterAuthorRole*) as *direct* instance.

Transformation to Approach 3. The transformation corresponds to the setting in Fig. 6. The difference from Approach 2 is that *Author* has no longer the semantics of role and thus is not subclass of *Role*. As *AuthorRole* is still instance of *Role*, we link it to *Author* using the annotation property *rdfs:seeAlso*.

The most important distinction is however not visible in this simple diagram. Namely, if there were a subclass system under *Author* in the original ontology, it would have to be transferred to the instance level. Instead of subclasses, there

[12] In the "books-about-animals" example in [7], the class is assumed to correspond to an animal species and the instance to the 'topic' of this animal. Both in the original example and in the example used in this paper, the nature of such instances—'roles' and 'topics', respectively—is thus not very coherent with the semantics of the class (whose name, be it e.g. *Lion*, or *Author*, is rather appropriate for a natural concept).

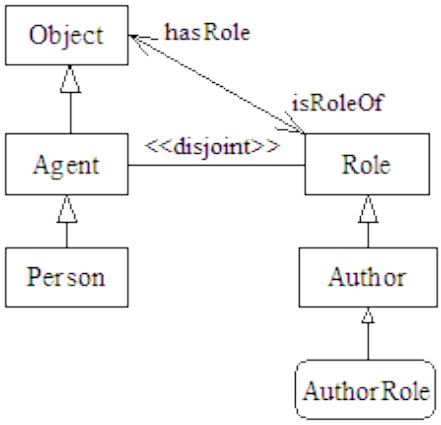

Fig. 5. Target of transformation using Approach 2

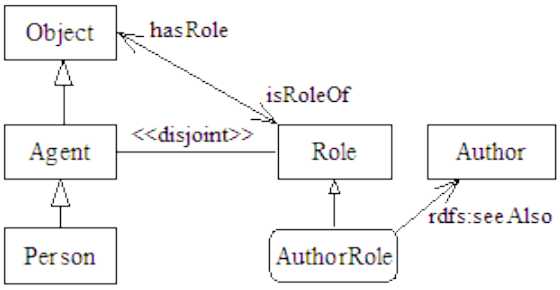

Fig. 6. Target of transformation using Approach 3

would be individuals such as *PosterAuthorRole*, which would be all direct instances of *Role*, and their specialization relationship would be modelled by a dedicated object property such as *subRoleOf*.

Transformation to Approach 4. This is similar to Approach 2 (including the treatment of a hypothetical subclass system), except that

- the original *Author* class is not removed
- consequently, the new class (subclass of *Role*) is not named *Author* but *AuthorRole*
- no special instances are generated for the new (*AuthorRole*) class
- instead, an additional restriction (in bold) is imposed on the *Author* class, relating it to the *AuthorRole* class.

The result is in Fig. 7.

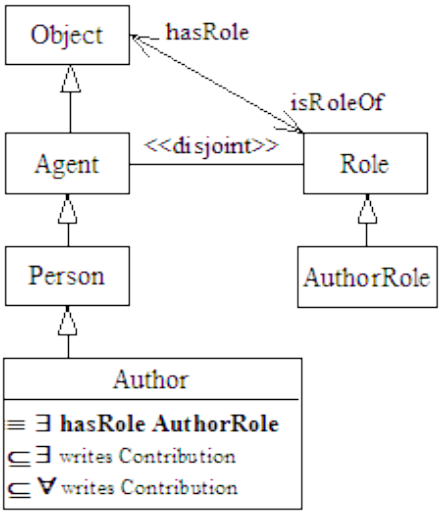

Fig. 7. Target of transformation using Approach 4

Transformation to Approach 5. The transformation corresponds to the setting in Fig. 8. In this approach, similarly to Approach 3, *Author* is not subclass of *Role*, and represents, together with its hypothetical subclasses, a branch of the ontology separate from *Person*, too. The *Role* class and the original *hasRole* and *isRoleOf* object properties, although imported with the *AgentRole* pattern, are not used at all. Instead, a new pair of *annotation properties*, borrowing the name of these two object properties, is defined. They can be used to directly connect instances of *Person* to class *Author* (or its subclass), as using classes as values of annotation properties is possible within OWL DL. Obviously, the downside is unavailability of information in annotations to conventional DL reasoners.

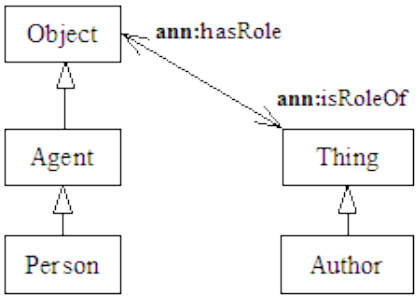

Fig. 8. Target of transformation using Approach 5

Feedback to the W3C Pattern. The context of transformation probably makes obvious that the five approaches from [7] (including Approach 1, which amounts to using a class directly as value of the property, thus lifting the ontology to the OWL Full dialect) are not the only possible choices for expressing the given conceptualization. Systematic variation of the (declaratively expressed) transformation pattern can give rise to multiple new approaches, which deserve to be further systematized, in a new round of the best practice identification process.

5.3 Additional Axioms

The implemented version of the transformation framework does not make distinction between the source pattern proper (i.e. structures whose occurrence is essential for the presence of the pattern) and additional, 'external' axioms that only 'touch' the pattern by referring to one of its entities. Such axioms may or may not be present for an ontology fragment to match the source pattern, but if they are present then they have to be considered by the transformation. Considering them as mandatory makes the whole transformation pattern over-specific and hard to understand. Therefore, for the new (not yet implemented) version of the transformation framework, we will decompose the transformation patterns into a mandatory part (containing the source and target ontology patterns proper, and their pattern transformation) and an optional part (containing the additional axioms, and their pattern transformation).

Specifically, when looking at Fig. 4, we see that the original *Author* class is used both in local (existential and universal) and global (domain and range) restrictions. If we use e.g. Approach 2, the *Author* class is however removed. The easiest but most lossy way of dealing with axioms that referred to it would be to drop the local restrictions together with the class, and to let the global restrictions refer to the immediate superclass (i.e., *Person*). However, we can also (at the cost of complexity overhead) replace the removed named class with the composed class expression $Person \sqcap \exists\ hasRole.Author$ in these axioms. While this is straightforward for the global restrictions (just setting the domain/range of the respective properties to be this class expression), for modelling the local

restrictions in OWL 2 DL we would have to employ an explicit anonymous class in order to link the two composed concept expressions, e.g.:

$$\_1 \equiv Person \sqcap \exists\ hasRole.Author$$
$$\_1 \sqsubseteq \exists\ writes.Contribution$$

Detailed discussion of the impact of such manipulations is however beyond the scope of this paper.

6 XML Serialisation of Transformation Patterns

Fig. 9 shows the XML serialisation of the transformation pattern for Approach 2. The codes for all four mentioned approaches are available in the transformation pattern library accessible from `http://nb.vse.cz/~svabo/patomat/tp/`.

As mentioned in Section 2, the transformation pattern consists of two ontology patterns, the source and the target one, plus a pattern transformation. Both OPs contain the given CP (*AgentRole*), with concrete classes, in their 'axioms' part (last six axioms in each OP).

In addition, the *source OP* declares two placeholders for classes, ?A and ?B, such that the second is subclass of the first (such as *Author* is of *Person*): this is the only axiom with placeholders. There is no naming detection pattern, for simplicity (as none is needed in this simple example). Additional axioms, mentioned in Section 5.3, are not yet considered either.

The *target pattern* declares two placeholders for classes, ?C and ?D, and one for individual, ?b, which is instance of the latter class. The first two of its axioms however already interlink the imported CP with placeholder classes (this corresponds to subordinating *Person* to *Agent* and *Author* to *Role*). The third axiom states that ?b is instance of ?D (such as *AuthorRole* to *Author*).

The *pattern transformation*, finally, declares the logical equivalence[13] transformation links between the classes from the source and target OP, and a simple naming pattern transformation for creating the name of the new individual by concatenating the name of class ?B with the string 'Role'.

7 Related Work

We are unaware of any style transformation approach used in connection with *content patterns* as in our current research. As regards our framework as such, several generic approaches to *ontology transformation* have recently been published. We refer here to two that look most relevant to our work (aside pure OPPL, which we ourselves use as an external component of our framework).

In [9] the authors consider *ontology translation* from the Model Driven Engineering perspective. The basic shape of our transformation pattern is very

[13] For simplicity we use an equivalence link even for ?B vs. ?D, although the equivalence between the class *Person* before and after transformation is arguable; removing a class and creating a new one would be sounder.

```
<tp>
  <op1>
    <entity_declarations>
      <placeholder type="Class">?A</placeholder>
      <placeholder type="Class">?B</placeholder>
    </entity_declarations>
    <axioms>
      <axiom>?B subClassOf ?A</axiom>
      <axiom>or:isRoleOf range or:Object</axiom>
      <axiom>or:hasRole domain or:Object</axiom>
      <axiom>or:hasRole range or:Role</axiom>
      <axiom>or:isRoleOf domain or:Role</axiom>
      <axiom>:Agent subClassOf or:Object</axiom>
      <axiom>:Agent disjointWith or:Role</axiom>
    </axioms>
  </op1>
  <op2>
    <entity_declarations>
      <placeholder type="Class">?C</placeholder>
      <placeholder type="Class">?D</placeholder>
      <placeholder type="Individual">?b</placeholder>
    </entity_declarations>
    <axioms>
      <axiom>?C subClassOf :Agent</axiom>
      <axiom>?D subClassOf :Role</axiom>
      <axiom>?b a ?D</axiom>
      <axiom>or:isRoleOf range or:Object</axiom>
      <axiom>or:hasRole domain or:Object</axiom>
      <axiom>or:hasRole range or:Role</axiom>
      <axiom>or:isRoleOf domain or:Role</axiom>
      <axiom>:Agent subClassOf or:Object</axiom>
      <axiom>:Agent disjointWith or:Role</axiom>
    </axioms>
  </op2>
  <pt>
    <eq op1="?A" op2="?C"/>
    <eq op1="?B" op2="?D" />
    <ntp entity="?b">?B+Role</ntp>
  </pt>
</tp>
```

Fig. 9. Simple transformation pattern for Approach 2

similar to their metamodel. They consider an *input pattern*, i.e. a query, an *output pattern* for creating the output, as well as variables binding the elements. However, the transformation is considered at the *data level* rather than at the *schema level* as (primarily) in our approach.

In comparison with the previous work the authors of [5] leverage the ontology translation problem to the generic meta-model. This work has been done from the *model management* perspective, which implies a generality of this approach. There are important differences to our approach. Although they consider transformations of ontologies (expressed in OWL DL), these transformations are directed into the generic meta-model or into any other meta-model such as that of UML or XML Schema. In contrast, in our approach we stay within one meta-model, the OWL language, and we consider transformation as a way of translating a certain representation into its modelling alternatives.

8 Conclusions and Future Work

We demonstrated that transformation patterns are a useful mediator for adequately importing content patterns into ontologies, especially into prototype ones that have been designed ad hoc, are not yet widely used, and now are to be put (through the content patterns) onto a more solid and shared foundation. Although the scenario used in this paper is quite narrow, we believe that it uncovers recurrent issues related to ontology style heterogeneity in general.

Aside the *AgentRole* pattern and its generalizations, we plan to explore other *content patterns* from *OntologyDesignPatterns.org*. Analogously, we will experiment with different *input ontologies*, which will require new transformation operations such as changing between properties and classes ('de/objectification'), as e.g. in the 'n-ary relations' logical pattern [8]. Obviously, the *cost/benefit ratio* of the CP import functionality should also eventually be examined with respect to the size of the ontology to be transformed and amount of data that already refer to it (and have to be transformed too, either at query time or in bulk).

The most critical future work, which is not specific for content pattern import but generic for pattern-based transformation, is however support for *recursive* and *optional* parts of patterns. This will lead to much more efficient transformation, as the transformation process, triggered at a 'root' entity (such as the *Author* class in *ConfOf*) could be propagated down the subclass (for different specific types of authors) or subproperty links, and yield an analogous, though stylewise different structure in the target ontology.

Another generic functionality we also plan to achieve in long term is systematic, pattern-based swapping of the information lost during style transformation into *OWL 2 annotations*, see e.g. [1] for initial considerations.

From the research point of view, a challenging task is to *automatically suggest* fragments of ontologies that should be (transformed and) subordinated to a content pattern. For the *AgentRole* pattern, for example, this challenge amounts to recognising classes or properties that implicitly express a role. In [12] we already formulated and in [13] made an initial evaluation (with promising result)

of a heuristic for detection of a 'role' class, through the occurrence of its name in (a naming pattern context of) a property having this class as domain. Much more complex heuristics, possibly inductively learnt, would however be needed for efficient recommendation.

This research has been partially supported by the CSF grant no. P202/10/1825, "PatOMat – Automation of Ontology Pattern Detection and Exploitation".
The authors are indebted to Eva Blomqvist, Aldo Gangemi and Valentina Presutti for inspiring discussions and hints regarding the ODP.org content patterns.

References

1. Annotation System. OWL WG, Work-in-Progress document, `http://www.w3.org/2007/OWL/wiki/Annotation_System`.
2. Egaña M., Stevens R., Antezana E.: Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language. In: OWLED. 2008. (z related work:)
3. Guizzardi G.: *Ontological Foundations for Structural Conceptual Models*, PhD Thesis, University of Twente, The Netherlands. Published as the book Ontological Foundations for Structural Conceptual Models, Telematica Instituut Fundamental Research Series No. 15, ISBN 90-75176-81-3 ISSN 1388-1795; No. 015; CTIT PhD-thesis, ISSN 1381-3617; No. 05-74.
4. Presutti V., Gangemi A.: Content ontology design patterns as practical building blocks for web ontologies.: In Proceedings of ER2008. Barcelona, Spain, 2008.
5. Kensche D., Quix C., Chatti M., Jarke M.: GeRoMe: A Generic Role Based Metamodel for Model Management. In: *Journal on Data Semantics*, Vol.8, p.82–117, 2007.
6. Motik B., Patel-Schneider P.F., Parsia B. (eds.): OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax. W3C Recommendation 27 October 2009, online at `http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/`.
7. Noy N. (ed.): Representing Classes As Property Values on the Semantic Web. W3C Working Group Note 5 April 2005, online at `http://www.w3.org/TR/swbp-classes-as-values/`.
8. Noy N., Rector A. (eds.): Defining N-ary Relations on the Semantic Web. W3C Working Group Note 12 April 2006, online at `http://www.w3.org/TR/swbp-n-aryRelations/`.
9. Parreiras F. S., Staab S., Schenk S., Winter A.: Model Driven Specification of Ontology Translations. In: 27th International Conference on Conceptual Modeling (ER-2008). n. 5231, p. 484–497. 2008.
10. Sunagawa E., Kozaki K., Kitamura Y., Mizoguchi R.: Role organization model in Hozo. In: Proc. EKAW'06, Podebrady, Czech Republic. Springer, LNCS.
11. Šváb-Zamazal O., Svátek V., Iannone L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: EKAW-2010, Lisbon, Portugal, 2010.
12. Svátek V., Šváb-Zamazal O., Presutti V.: Ontology Naming Pattern Sauce for (Human and Computer) Gourmets. In: Workshop on Ontology Patterns at ISWC'09, Washington DC, 2009. Online `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-516/`
13. Svátek V., Šváb-Zamazal O.: Entity Naming in Semantic Web Ontologies: Design Patterns and Empirical Observations. In: Znalosti 2010, 9^{th}Czecho-Slovak Annual Knowledge Technology Conference, Jindřichův Hradec 2010, Czech Republic.

Reusing Ontology Design Patterns in a Context Ontology Network

María Poveda-Villalón, Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez
Ontology Engineering Group. Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo s/n.
28660 Boadilla del Monte. Madrid. Spain
mpoveda@delicias.dia.fi.upm.es, {mcsuarez, asun}@fi.upm.es

Abstract. Reusing knowledge resources, specifically Ontology Design Patterns (ODPs), has became a popular technique within the ontology engineering field. Such a reuse allows speeding up the ontology development process, saving time and money, and promoting the application of good practices. Recently methods and tools to support the reuse of ODPs have emerged. In addition, the existence of detailed examples of real use cases that reuse ODPs favours the adoption and application of such methods. Thus, our objective in this paper is to show an example of how to apply a method for reusing ODPs during the development of a context ontology network to model context-related knowledge that allows adapting applications based on user context. Besides, in this paper we present the main drawbacks found during the application of the reuse method as well as some proposals to overcome them.

Keywords: ontology design pattern, ontology development, context ontology

1 Introduction

With the goal of speeding up the ontology development process, ontology practitioners are starting to reuse [11, 12] as much as possible knowledge resources (ontologies, ontology modules, ontology statements, and ontology design patterns as well as non-ontological resources). This approach also allows ontology developers to save time and money and to promote the application of good practices.

Regarding the ODPs, their reuse has proven different benefits [1] like (a) to make easier the ontology development or (b) to produce ontologies of better quality. The reuse of ODPs should be supported by methods and tools to minimize the reuse effort and to maximize their benefits. In this sense, methods and tools to support the reuse of ODPs have recently emerged.

However, methods and tools are not enough to guarantee a successful reuse of ODPs. In this regard, the existence of detailed examples of how ODPs are reused in real use cases would be of great help. This kind of guided examples would favour the adoption of ODPs and their related methodological and technological support.

Thus, our main objective in this paper is to present a guided example of how we have carried out the reuse of ODPs during the development of an ontology network about user's contextual information, called mIO! ontology network. This ontology

network is a context ontology in the mobile environment that aims to represent contextual knowledge about the user that can influence his interaction with mobile devices. In addition, we also show in this paper some lessons learned during the application of the method for reusing ODPs as well as some proposals for overcoming the drawbacks found in the method.

The remainder of the paper is structured as follows: Section 2 summarizes the state of the art on methodological and technological aspects of the ODPs reuse. Section 3 describes in detail how we have followed the guidelines to reuse ODPs as well as how we have adapted such guidelines during the development of the mIO! ontology network. Section 4 shows the lessons learned during the ODPs reuse. Finally, Section 5 concludes and shows some lines of future work.

2 Ontology Design Patterns and how to reuse them

The idea of applying patterns for modelling ontologies was proposed by [2]. Since then, relevant works on patterns have been the Semantic Web Best Practices and Deployment Working Group[1], the Ontology Design Patterns Public Catalogue[2], and the Ontology Design Patterns Portal[3]. According to [4] Ontology Design Patterns (ODPs) are modeling solutions to solve a recurrent ontology design problem. ODPs are a way of encoding best practices, based on experiences and knowledge of 'good' solutions. In [5], authors distinguish the following six different types of ODPs: reasoning, structural, content, presentation, lexico-syntactic, and correspondence.

Generally patterns are perceived as having three kinds of benefits [5]: (1) reuse benefits, (2) guidance benefits, and (3) communication benefits. The ontology design patterns reuse refers to the activity of using available ontology design patterns in the solution to different modelling problems during the development of new ontologies [15]. The goal of the ODPs reuse is to facilitate the solution of modelling issues and to improve interoperability through using well-proven solutions and best practices, in the form of patterns. In this sense, there are experiments that prove that the reuse of ODPs makes the ontology development process easier and improves the quality of the resulting ontologies [1].

Even when such benefits have been established, there is the need to provide methodological and technological support to the pattern usage with the goal of (a) minimising the reuse effort and (b) maximising the ODPs benefits. In this regard, methods and tools have recently appeared to guide and support the ODPs reuse.

On the one hand, a method for reuse any type of ODP, called XD (for eXtreme Design), is presented in [3, 9]. In this method the ODPs reuse activity is divided into eight tasks (1. Identify requirements to be addressed; 2. Identify available patterns; 3. Divide and transform the problem, select a partial problem; 4. Match selected partial problem to patterns; 5. Select patterns; 6. Apply (reuse) selected patterns and compose them; 7. Evaluate and revise with respect to partial problem; and 8. Integrate partial solutions). The method has as input the Ontology Requirements Specification

[1] http://www.w3.org/2001/sw/BestPractices/OEP/
[2] http://www.gong.manchester.ac.uk/odp/html/index.html
[3] http://ontologydesignpatterns.org

Document (ORSD) and a set of available ODPs, and as output an ontology network including the ODPs reused. In summary, the workflow starts with the identification of the requirements to be addressed by reusing ODPs and the available ODPs registries and libraries. Next, an iterative set of tasks takes place in which (a) each problem is divided into sub-problems, (b) a sub-problem is selected and matched with the ODPs, (c) the ODPs are selected and applied, and (d) the result is evaluated. This cycle is repeated until all the sub-problems are addressed. Finally, the method concludes by integrating all the partial solutions resulting in an ontology network which contains the ODPs selected. The XD method could be specialized to address the reuse of concrete types of ODPs, for example, content patterns [3, 9, 13].

It is well known that the adoption of any emerging methodology or method is facilitated by providing detailed examples of how to apply them to real and complete use cases. However, as far as we know, there are no complete examples of how to apply the XD method to the reuse of any type of ODP. Thus, we present in this paper a detailed example of how we have reused different types of ODPs during the development of a context ontology network.

On the other hand, there are also tools that provide a technological support for the ODPs reuse activity. The plug-in for NeOn Toolkit[4] called "XD Tools"[5] allows the content ODPs reuse accordingly to this variant of the XD method. Whereas, the ontology editor Protégé 3.4.4[6] allows the automatic reuse[7] of the following logical patterns and good practices: "Value Partition", "Enumeration", "N-ary Relation", "OWL List", and "RDF List".

3 ODPs Reuse in the mIO! Ontology Network

As already mentioned, the main purpose of this context ontology, called mIO! ontology network, is to represent knowledge related to the user context, e.g., information on location and time, user information and its current or planned activities, as well as devices located in his surroundings. The ontology aims at solving the challenge of configuring, discovering, executing, and enhancing services based on the user context.

The development of this ontology network about contextual information has been carried out following the NeOn Methodology [12]. During the development of such an ontology network we have reused different knowledge resources (ontologies, non-ontological resources, and ODPs) as explained in [8]. In this paper, we focus exclusively on how we have performed the reuse of ODPs.

To reuse ODPs we have applied the general method described in [13] rather than the one explained in [9], since we are not interested exclusively in content patterns but in any type of ODP. For the same reason we have manually put into practice method instead of using the XD Tools. It is important to add that in some cases during the application of the method we have had to adapt and/or extend the guidelines provided.

[4] http://neon-toolkit.org
[5] http://neon-toolkit.org/wiki/XDTools
[6] http://protege.stanford.edu/download/registered.html#p3
[7] This reuse is performed in a guided way by means of a wizard.

Thus, in this section we describe (a) how we have applied the general XD method[8] for reusing different types of ODPs and (b) how we have adapted and/or extended the guidelines provided by the method.

Task 1. Identify requirements to be addressed

To identify the requirement to be addressed by means of the reuse of ODPs, we have chosen the third question proposed in [13] based on our knowledge about ODPs. The other two questions proposed in [13] could result in selecting requirements that cannot be solved by reusing ODPs and for this reason we have discarded such questions. Therefore, the selected question is *"What requirements can be associated with existing patterns types?"*.

The main drawback at this point is the fact that to be able to answer this question developers should be versed in ODPs types. In this sense, based on our knowledge about ODPs, we have not selected those requirements that could be addressed by existing ODPs types but those that apparently can be solved by reusing existing ODPs.

Table 1 shows both the non-functional and the functional requirements extracted from the ORSD[9] that will be addressed by reusing ODPs. The functional requirements belong to the following subdomains as we can observe in the identifier[10] field: Interface (INT), Device (DSP), Time (TMP), Location (LOC), and User (USR). These functional requirements can be written both as Competency Questions (CQs) and their corresponding responses [6] and as affirmative sentences in natural language (NL) as explained in [8].

Table 1. Requirements to be addressed by reusing ODPs

Non-functional requirements	
• The ontology should be modular.	
Functional requirements	
Requirement identifier	**Requirement**
CQ identifier	**CQ and its response**
DSP_PC7	What devices exist? Display, touchscreen, balise, keyboard, trackball, pulse oximeter, glucose meter, emvironmental temperature sensor, environmental humidity/pressure sensor, Anemometer, CO2 level sensor, printer, camera, GPS Receiver, short distance communication module (NFC. ZigBee, Bluetooth), long distance communication module (GPRS, UMTS, WiFi), processing module, memory module, loudspeaker, microphone and reading/writing module NFC
DSP_PC9	What are the components of a display? A display is composed by: - input interfaces - presentation surface - control interfaces - power system

[8] To simplify the example description, we present a unique process addressing simultaneously all the requirements.

[9] The whole ORSD for the mIO! ontology network is available in [7].

[10] The abbreviations come from the Spanish name of each subdomain: *Intefaz (INT), Dispositivo (DSP), Tiempo (TMP), Localización (LOC),* and *Usuario (USR).*

CQ identifier	CQ and its response
DSP_PC49	What are the components of a CO2 level sensor? A CO2 level sensor is composed by: - source - power system - detector - amplifier - output /download data port (optional)
DSP_PC61	What are the components of a printer? A printer is composed by: - leaf storage receptacle - printhead - ink container - storage device - processing device - communication interface - screen
DSP_PC71	What are the components of a camera? A camera is composed by: - lens - image capture device - image processing device - storage device - communication interface - positioning device - lighting device - screen - microphone
DSP_PC83	What are the components of a GPS receiver? A GPS receiver is composed by: - antenna - signal processor - processing module - communication interface - screen - speaker - microphone
DSP_PC96	What are the components of a speaker? A speaker is composed by: - communication interface - amplifier - decoder - signal processing module - active element - casing
DSP_PC102	What are the components of a microphone? A microphone is composed by: - diaphragm - coil - permanent magnet A condenser microphone is composed by: - diaphragm of lightweight and flexible membrane - rigid backplate - cable to the preamplifier - bias voltage feeder
TMP_PC4	What are the days of the week? Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday
LOC_PC14	When was the last time that the user was in the location X? The day Z from xx: xx: xx to yy: yy: yy
NL sentence identifier	**Affirmative NL Sentence**
INT_CA3	The interfaces types are: - conversational - gestural - graphic - natural language - command line - multi screen - touch - textual - vocal - web
USR_CA7	A user will be in a specific location at any given time. The possible physical movement of the user is associated with this aspect

Task 2. Identify available patterns

To carry out this task, first we have identified different libraries and repositories in which patterns could be found. Next, based on our knowledge about ODPs and on

their descriptions[11] and uses we have sketched preliminary correspondences between the requirements selected in Task 1 and the available ODPs. Note that there are patterns that belong to several libraries or repositories.

The ODPs resources and the ODPs suitable to be reused are:

- **Ontology Design Patterns (ODP) Portal**[12]: this Wiki aims to gather ODPs in a repository. The patterns are reviewed by a quality committee. Besides, users can submit new patterns or modelling issues. In this portal we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Componency
  - N-Ary Participation
- **W3C Semantic Web Best Practices and Deployment (SWBPD)**[13]**:** this repository contains pattern descriptions and some examples of modelling patterns. The repository also includes other types of best practices as guidelines, repositories of vocabularies and ontologies, learning material, and demos. In this repository we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - N-ary Relations. Pattern 1: Introducing a new class for a relation
  - Specified Values in OWL: "value partitions" and "value sets"
- **NeOn D2.5.1 [10]:** *A Library of Ontology Design Patterns*: *reusable solutions for collaborative design of networked ontologies*. This catalogue contains content patterns classified by domains**.** In this resource we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Componency (CP-COM-01)
  - N-Ary Participation (CP-NPAR-01)
- **NeOn D5.1.1 [14]**: *NeOn Modelling Components.* This catalogue provides descriptions about logical, architectural, and some content patterns. In this resource we have found the following suitable patterns to be reused within the mIO! ontology network development:
  - Modular Architecture (AP-MD-01)
  - Taxonomy (AP-TX-01)
  - Part-Whole Relation (CP-PW-01)
  - Specified Values: Set of Individuals (LP-SV-01)
  - Specified Values: Subclasses (LP-SV-02)
  - N-ary Relation: New Class (LP-NR-01)

To illustrate how we have mentally sketched a correspondence between requirements and possible ODPs we present here some examples.

Let us take the DSP_PC9 requirement as the starting point. We can observe that in the requirement appear the terms “components”, “composed by” and a list of the parts

[11] These descriptions usually include (a) the name of the pattern, (b) a graphical description, (c) an enumeration of its elements, and (d) a set of requirements (in the form of CQs) or use cases (in the form of sentences in natural language) the pattern is intended to address. Sometimes, the code in an ontology implementation language is also provided.

[12] http://ontologydesignpatterns.org/

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

that form a display. These characteristics of the requirement are clues to think about the "Part-Whole Relation" or "Componency" patterns due to the information included in the description and uses cases of these patterns. The same reasoning can be applied to the requirements DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102.

In the case of the requirement TMP_PC4 the days of the week are enumerated. It is well known that this is a unique enumeration and the days of the week are different among them. These features can lead a developer to think in the "Specified Values: Set of Individuals" and the "Specified Values: Subclasses" patterns.

Task 3. Divide and transform the problem, select a partial problem

At this point the method [13] suggests transformations like (a) writing CQs to represent requirements stated only in example scenario sentences if not already present in the ORSD and (b) grouping similar CQs that may be solved together.

In our case, the selected requirements are quite simple so it was not necessary to divide them. In fact, some of them have been grouped during the transformation because of their similarity. Besides, we have only transformed the functional requirements since the non-functional one represents a "manageable piece" itself.

To perform the transformation of the problem we propose the following approach. We have taken into account that the functional requirements selected in Task 1 can be written as CQs and their responses or as affirmative sentences in natural language as Table 1 shown. In addition, we have observed that some suitable patterns to be reused (e.g. the Taxonomy (AP-TX-01) pattern [14]) have as part of their descriptions general use cases in form of sentences in natural language instead of CQs. Bearing in mind all these facts, we have transformed the requirements depending on the descriptions fields of the patterns suitable to be reused. Therefore, some CQs have been transformed into sentences in natural language and vice-versa. Others requirements have been transformed according to the suitable patterns descriptions but maintaining their original format. Table 2 summarizes the transformation types proposed in our approach and shows the specific transformation cases that have occurred during the mIO! use case. Finally, the resulting transformations are shown in Table 3.

Table 2. Requirement transformation cases

		The requirement was transformed into:	
		Sentence in NL	**Competency Question**
The requirement was written as:	**Sentence in NL**	INT_CA3	USR_CA7
	Competency Question	DSP_PC7 TMP_PC4	DSP_PC9 DSP_PC49 DSP_PC61 DSP_PC71 DSP_PC83 DSP_PC96 DSP_PC102 LOC_PC14

It should be noted that there are no guidelines to carry out the requirements transformation. In our case, we have carried out the transformation based on how general are the original requirements in the ORSD. Having in mind this approach, we have abstracted the requirements in order to make them similar to the ODPs descriptions. For example, in the requirements DSP_PC9, DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102 a list of components is shown for each concrete device. During the transformation, we have abstracted these requirements writing them as a unique CQ that could be applied to any of them and that could be match with the CQs addressed by some available patterns.

Table 3. Transformation of subproblems

Requirement identifier	Transformation
INT_CA3	Perform categorization/classification of interface types at different extents of granularity.
DSP_PC7	Perform categorization/classification of devices at different extents of granularity.
DSP_PC9	What are the components of this device? *Response*: A list containing the parts of the device.
DSP_PC49	
DSP_PC61	
DSP_PC71	
DSP_PC83	
DSP_PC96	
DSP_PC102	
TMP_PC4	List the days that make up the week.
LOC_PC14	What is the location of a particular user at a particular time? *Response*: A structure containing the location of the user in a given point of time.
USR_CA7	

It is important to mention here that there are two important drawbacks at this point of the reuse. First, we cannot transform the requirements only into CQs because there are ODPs that only have in their descriptions use cases written as sentences in natural language, as we have already noted. Second, ontology developers should be well versed in ODPs to discern what type of transformation they should carry out. Therefore, we have realized the high importance of making a finer identification of the most suitable ODPs in Task 2.

Task 4. Match selected partial problem to patterns

As it can be observed in Table 3 none of the requirements has been divided into sub-problems, but they have been simply transformed. Therefore, it was not necessary to assign a new numbering for the partial problems since they correspond to the identifier of each requirement.

The matching between the problems to be addressed by reusing ODPs and the ODPs available in libraries and repositories, identified in Task 2, is shown in Table 4.

Table 4. Matching between partial problems and ODPs suitable to be reused

Non-functional requirements		
Requirement		**Suitable pattern(s)**
• The ontology should be modular.		• Modular Architecture (AP-MD-01)
Functional requirements		
Requirement identifier	**Transformation**	**Suitable pattern(s)**
INT_CA3	Perform categorization/classification of interface types at different extents of granularity.	• Taxonomy (AP-TX-01)
DSP_PC7	Perform categorization/classification of devices at different extents of granularity.	• Taxonomy (AP-TX-01)
DSP_PC9	What are the components of this device?	• Componency • Componency (CP-COM-01) • Part-Whole Relation (CP-PW-01)
DSP_PC49		
DSP_PC61		
DSP_PC71		
DSP_PC83		
DSP_PC96		
DSP_PC102		
TMP_PC4	List the days that make up the week.	• Specified Values in OWL: "value partitions" and "value sets" • Specified Values: Set of Individuals (LP-SV-01) • Specified Values: Subclasses (LP-SV-02)
LOC_PC14	What is the location of a particular user at a particular time?	• N-Ary Participation • N-ary Relations. Pattern 1: Introducing a new class for a relation • N-ary Participation (CP-NPAR-01) • N-ary Relation: New Class (LP-NR-01)
USR_CA7		

It is important to mention that at this point the method does not provide detailed guidelines about how to match the requirements with the available patterns. As we have explained in Task 1, we have selected those requirements that could be covered by at least one available ODP. Based on our knowledge about ODPs and on our experience reusing them, we can propose the following heuristics to quickly identify a suitable ODP to be reused while reading the requirements:

- If a requirement mentions something about a list of values, we could infer that the “Specified Values in OWL: "value partitions" and "value sets"”, the “Specified Values: Set of Individuals (LP-SV-01)” or the “Specified Values: Subclasses (LP-SV-02)” patterns could be reused.
- If a requirement is about types and subtypes of a given concept, we could infer that the “Taxonomy (AP-TX-01)” pattern could be reused.
- If a requirement contains more than two concepts related, we could infer that the “N-Ary Participation”, “N-ary Relations. Pattern 1: Introducing a new class for a relation”, “N-ary Participation (CP-NPAR-01)” or the “N-ary Relation: New Class (LP-NR-01)” patterns could be reused.

- If a requirement is about parts of something, we could infer that the "Componency", "Componency (CP-COM-01)" or "Part-Whole Relation (CP-PW-01)" patterns could be reused.
- If a requirement is about the need for modularity within the ontology, we could infer that the "Modular Architecture (AP-MD-01)" pattern could be reused.

Task 5. Select patterns

At this point, we must select the most appropriate pattern in those cases identified in Task 4 in which there are several suitable patterns to cover the requirements. Since there are no detailed guidelines to perform this task, we have taken the following decisions for requirements related to more than one potential ODP:

- For the requirements DSP_PC9, DSP_PC49, DSP_PC61, DSP_PC71, DSP_PC83, DSP_PC96, and DSP_PC102, there is no need to represent transitive relationships. For this reason, we have selected the "Componency (CP-COM-01)" pattern instead of the "Part-Whole Relation (CP-PW-01)" pattern. It is important to mention that we have reused the "Componency" pattern from the ODP Portal instead of the one included in the NeOn D2.5.1 [10], because the ODP Portal provides an OWL file that contains the source code for the pattern.
- For the TMP_PC4 requirement is needed to represent a set of individuals whose enumeration is equivalent to the parent class. Therefore, we have selected the "Specified Values: Set of Individuals (LP-SV-01)" pattern[14] instead of the "Specified Values: Subclasses (LP-SV-02)" pattern.
- For the requirements LOC_PC14 and USR_CA7 there is no need to represent events or situations. For this reason, we have selected the "N-ary Relation: New Class (LP-NR-01)" pattern[15] instead of the "N-ary Participation" pattern or the "N-ary Participation (CP-NPAR-01)" one.

It is worth mentioning that for the non-functional requirement and for the requirements INT_CA3 and DSP_PC7, the ODPs related to those requirements in Table 4 have been directly selected.

Task 6. Apply (reuse) selected patterns and compose them

In this task we have observed that ODPs can take different roles in different stages of an ontology development process depending on their types and the problem that they address. For example, we have distinguished the following situations:

- Reusing ODPs to define the architecture of the ontology network during the conceptualization activity through the "Modular Architecture (AP-MD-01)" pattern.
- Reusing ODPs in the implementation activity to complete the knowledge represented; for example, we can enrich a mereological relationship through the "Componency (CP-COM-01)" pattern.

[14] In this case the use of the "Specified Values: Set of Individuals (LP-SV-01)" pattern and the "Specified Values in OWL: "value partitions" and "value sets"" patterns would be equivalent.

[15] In this case the use of the "N-ary Relation: New Class (LP-NR-01)" pattern and the "N-ary Relations. Pattern 1: Introducing a new class for a relation" patterns would be equivalent.

- Reusing ODPs in the implementation activity to represent logical structures that are not supported by the ontology language, for example, the "N-ary Relation: New Class (LP-NR-01)" pattern.
- Reusing ODPs in the implementation activity to join concepts from different ontologies, for example, thorough the "N-ary Relation: New Class (LP-NR-01)" pattern.

In addition, here we show the result of applying the patterns selected in Task 5 to the mIO! ontology network[16].

The reuse of the "Modular Architecture (AP-MD-01)" pattern has given rise to the modular structure that forms the miO! ontology network. Such a structure is shown in Fig. 1[17].

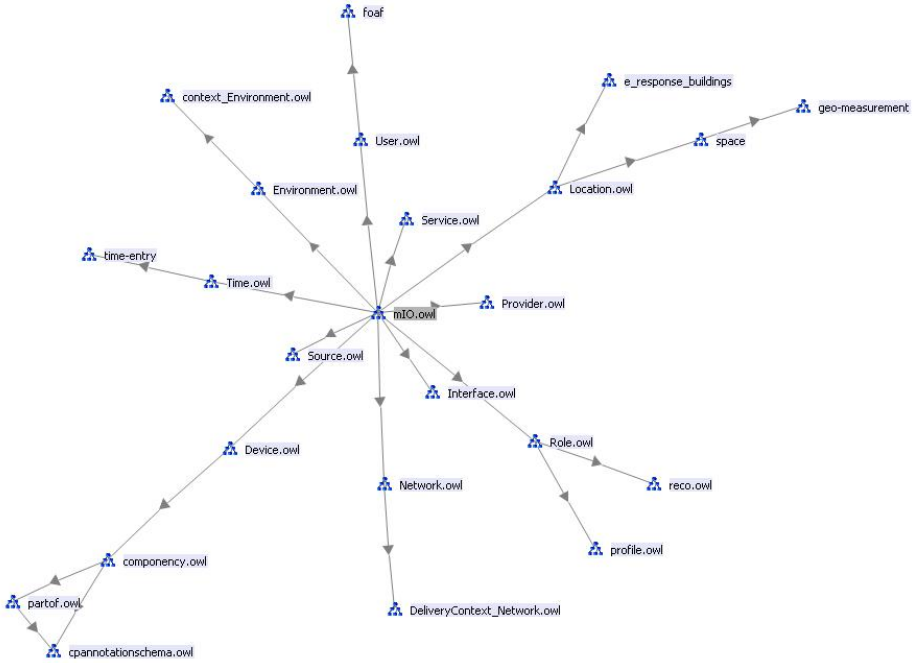

Fig. 1. "Modular Architecture" pattern applied to the mIO! ontology network

The reuse of the "Componency (CP-COM-01)" pattern has been carried out within the Device.owl ontology by importing (See [10] for more information about this operation) (Fig. 2-a) the componency.owl pattern and specializing (See [10] for more information about this operation) (Fig. 2-b) the pattern.

[16] The screenshots shown have been taken from NeOn Toolkit's default tabs and the "Relationship Browser" (http://www.neon-toolkit.org/wiki/2.3.1/Relationship_Browser) plug-in.

[17] The blue dotted triangles represent ontologies, whereas the arrows with grey solid triangles represent the "import" relationship between ontologies. These relationships must be understood as follows: the mIO.owl ontology imports the Service.owl one.

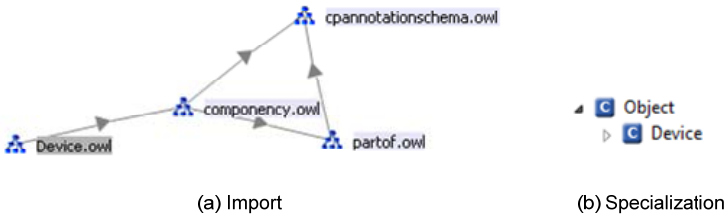

Fig. 2. "Componency" pattern applied to Device subdomain

The result of reusing the "Specified Values: Set of Individuals (LP-SV-01)" pattern during the modelling of the days of the week is shown in Fig. 3[18].

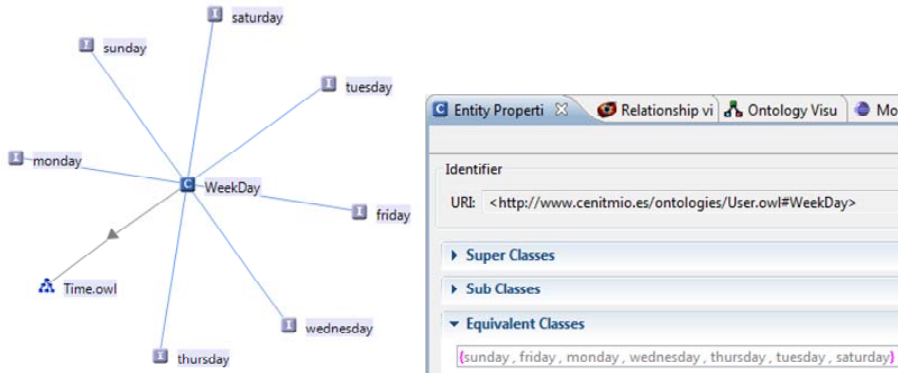

Fig. 3. "Specified Values: Set of Individuals" pattern applied to the days of the week

Fig. 4[19] depicts the result of reusing the "N-ary Relation: New Class (LP-NR-01)" pattern to model the location of a user at given point of time. In the model shown in Fig. 4 has been taken into account both locations in a geo-political entity, such as a country or a city, and locations specified by coordinates.

Finally, it is worth mentioning that Task 7 (Evaluate and revise with respect to partial problem) and Task 8 (Integrate partial solutions) have not been carried out as authors propose in [13] since none of the requirements has been divided into sub-problems. In addition, the reuse of patterns has been carried out by a single development team so there is no need to integrate solutions developed in parallel by different teams. Finally, the evaluation of the obtained model has been carried out during the evaluation of the whole ontology network.

[18] The days of the week are represented by boxes with an "I" to indicate that they are instances, whereas the class "WeekDay" is represented by a box with a "C". In addition the lines that link the class with the instances indicate that they belong to the class.

[19] The ellipses represent classes and the lines with a triangle represent relationships among classes.

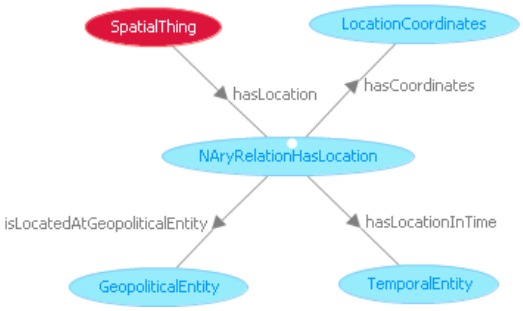

Fig. 4. "N-ary Relation: New Class" pattern applied to locations at given point of time

4 Lessons Learned

After the application of the general XD method to reuse ODPs during the development of the mIO! ontology network, we have realized (a) the usefulness of following a method to guide the ODPs reuse during the ontology building and (b) the advantages of reusing ODPs, ensuring the use of good practices in the ontology.

However, we have also realized the difficulty of applying the abovementioned method because of the lack of detailed guidelines in some of the tasks. For this reason in this section we report (a) some of the lessons we have learned during the application of the method as well as (b) some proposals that could be valuable for any enhancement of the method and/or for any further development that reuses ODPs.

During the execution of *Task 1*, we have realized that a beginner could select requirements that cannot be solved by means of reusing ODPs when he/she answers the questions proposed in [13] for such a task. Thus, ontology developers must have some experience with ODPs to make a more direct identification of the requirements that will be addressed. Such an experience is needed to select only those requirements that can be fulfilled by reusing ODPs.

We can also add that in those cases in which a beginner selects requirements that cannot be solved by means of ODP reuse, such a developer would have at least the following two options: (a) to propose a new pattern to cover such requirements and (b) to cover such requirements with other knowledge resources (such as ontologies or non-ontological resources) as proposed in [12, 16].

In the case of *Task 2*, we also consider that experience with ODPs is required again. In this task some types of patterns, as content patterns, can be identified as suitable to be reused by means of tools (e.g., XD Tools); however, there are other types of patterns that cannot be identified using tools due to (a) such patterns have no CQs in their description to match with the requirements to be addressed or (b) the patterns are not available at on-line libraries. Thus, in these cases, the ontology developer should have large knowledge about both ODPs and repositories to identify manually the patterns.

Besides, in this task it could happen that for a given requirement there are no patterns related or that the developers cannot find patterns suitable to be reused. In that case, we propose the following alternatives that can be included in the method:

(a) to posting a modelling issue[20] within the ODP Portal related to the given requirement; (b) to look for others resources suitable to be reused such as ontologies or non-ontological resources as explained in [12] and [16] respectively; and (c) to manually face the problem and to submit a proposed pattern[21] that covers the given requirement to the ODP Portal.

To carry out *Task 3* the method [13] suggests transforming the requirements (written as example scenario sentences) into CQs. However, in our case we have also had to transform some of the requirements into sentences in natural language. This transformation was needed to match the requirements to be addressed with some ODPs use cases, as already explained in Section 3.

Regarding to *Task 4*, based on our experience applying the method, this task only seems necessary if at least one requirement has been divided in Task 3. In other cases, Task 4 can be considered similar to Task 2.

We can also mention that after performing *Task 5* it is possible that no pattern matches with the problem to be addressed. In this case, we propose to follow the same options already presented for Task 2.

Finally, we have observed that ODPs can be applied at different points of an ontology development. For example, in the mIO! ontology network case the "Modular Architecture (AP-MD-01)" pattern was applied during the conceptualization activity whereas the rest of patterns were applied during the implementation activity.

5 Conclusions and Future Lines of Work

This paper presents an example of how to apply the general XD method to reuse different types of ODPs (logical, architectural, and content patterns). This application has been performed with a real use case within the development of a context ontology network, called mIO! ontology network[22]. This guided example could be used together with the method in further ontology developments, what would favour the adoption of ODPs and of the abovementioned method.

During the process we have taken advantage of following a guided method that sets and orders the tasks to carry out during the reuse of ODPs. The method also provides some examples or criteria to take into account as well as a list of catalogues where to find ODPs. Once the method was applied, we have realized that time was saved in the conceptualization and implementation activities.

However, we have also identified some points during the reuse process where the developers' experience on ODPs seems quite important (tasks 1 and 2). In addition, we have discovered some drawbacks in the general XD method that could be solved by extending and improving the guidelines for tasks 1, 2, 3, and 5, as already explained in Section 4.

As future work we have plan to apply the XD method together with the XD Tools in a collaborative ontology development within a real use case. The idea of this new application of the method is to focus on requirements that have to be divided into

[20] http://ontologydesignpatterns.org/wiki/Community:PostModelingIssue
[21] http://ontologydesignpatterns.org/wiki/Submissions:SubmitAPattern
[22] http://www.oeg-upm.net/index.php/es/ontologies/82-mio-ontologies

different sub-problems. Our final aim is to analyze (a) what tasks the XD Tools make easier and (b) what tasks still need more detailed guidelines.

Acknowledgments. This work has been partially supported by the Spanish project *mIO!* (CENIT-2008-1019).

References

1. Blomqvist, E., Gangemi, A., Presutti, V. *Experiments on Pattern-based Ontology Design*. In Proceedings of K-CAP 2009, pp. 41-48. 2009.
2. Clark, P., Thompson, J., & Porter, B. W. *Knowledge Patterns*. In KR2000: Principles of Knowledge Representation and Reasoning. pp. 591-600. 2000.
3. Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., Villazón-Terrazas, B. *NeOn D2.5.2 Pattern based ontology design: methodology and software support.* NeOn project. http://www.neon-project.org. 2010.
4. Gangemi, A., Gomez-Perez, A., Presutti, V., Suarez-Figueroa, M.C. *Towards a Catalog of OWL-based Ontology Design Patterns*. In proceedings of CAEPIA. 2007.
5. Gangemi, A.; Presutti, V. *Ontology Design Patterns*. Handbook on Ontologies (Second Edition). Springer. International Handbooks on Information Systems. 2009.
6. Gruninger, M., Fox, M. S. *The role of competency questions in enterprise engineering*. In Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, Trondheim, Norway, 1994.
7. Poveda, M. *Metodología NeOn Aplicada a la Representación del Contexto*. Master Thesis. Spain. *Universidad Politécnica de Madrid*. September, 2010.
8. Poveda, M., Suárez-Figueroa, M.C., García-Castro, R., Gómez-Pérez, A. *A Context Ontology for Mobile Environments*. Proceedings of CIAO 2010. Lisbon, Portugal. 11 October 2010.
9. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E. *eXtreme Design with Content Ontology Design Patterns*. In Proceedings of WOP 2009. Washington D.C., USA, 25 October, 2009, Vol. 516 CEUR Workshop Proceedings. 2009.
10. Presutti, V., Gangemi, A., David S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M. *NeOn D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies.* NeOn project. http://www.neon-project.org. 2008.
11. Simperl, E. *Reusing ontologies on the Semantic Web: A feasibility study*. Data Knowledge Engineering. Volume 68. Number 10. Pages: 905-925. 2009.
12. Suárez-Figueroa, M.C. *PhD Thesis: NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. Spain. Universidad Politécnica de Madrid. June 2010.
13. Suárez-Figueroa, M.C., Blomqvist, E., D'Aquin, M., Espinoza, M., Gómez-Pérez, A., Lewen, H., Mozetic, I., Palma, R., Poveda, M., Sini, M., Villazón-Terrazas, B., Zablith, F., Dzbor, M. *NeOn D5.4.2: Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. NeOn project. http://www.neon-project.org. February 2009.
14. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.. *NeOn D5.1.1: NeOn Modelling Components*. NeOn project. http://www.neon-project.org. March 2007.
15. Suárez-Figueroa, M.C., Gómez-Pérez, A. *First Attempt towards a Standard Glossary of Ontology Engineering Terminology*. 8th Proceedings of TKE 2008. 18-21 August 2008.
16. Villazón-Terrazas, B. *PhD Thesis: A Method for Reuse and Re-engineering Non-Ontological Resources into Ontologies*. Spain. *Universidad Politécnica de Madrid*. To be appeared.

Position Paper

A Decision-making Format for the Semantic Web

[Position Paper]

Eva Blomqvist
STLab, ISTC-CNR
eva.blomqvist@istc.cnr.it

Marion Ceruti
Jeff Waters
Space and Naval Warfare
Systems Center Pacific
marion.ceruti@navy.mil;
jeff.waters@navy.mil

Don McGarry
MITRE Corporation
dmcgarry@mitre.org

ABSTRACT

This paper describes the work of the W3C Decisions and Decision-making Incubator[1], with the goal to identify requirements for a standard decision format, through a set of use cases, and to develop a first version of a potential standard format for representing decisions, fulfilling the requirements of the use cases and exploiting semantic web standards. Ongoing efforts include the identification and modelling of 'decision patterns' and development of proof-of-concept applications to validate assumptions and patterns.

Keywords

Decision Making, Decision Format, Ontology Pattern

1. INTRODUCTION

The time and effort we spend converting our decisions into work products, such as briefs, proposals, and communication of decisions in meetings, conversations, and emails, could be reduced if we had a standard format for representing and sharing decisions. Our tools could be instrumented to generate our decisions in a format that could be shared and also track the state of decisions within the decision-making process. Instrumentation could support the development of a metric of information flow and help us optimize our decision processes across our organization or enterprise [7]. Visibility of the decisions in their formation and evolution would enable proactive management and assistance from others [8].

1.1 Usage Scenarios

Sharing decisions across a broad and diverse set of users and systems is an important aspect of situational awareness in many domains, for instance, in emergency management[2]. During an emergency, decisions must be shared among emergency managers and first responders from multiple organizations, jurisdictions, and functional capabilities. For example, decisions to route patients must be shared among first responders in the field who are sending the patients, those who are doing the transport, the medical facilities receiving the patients, and the patient's families and relatives.

First responders and emergency managers work under difficult conditions using current mechanisms for information sharing; they need improved solutions. For example, paper-based Incident Command forms provide an initial standardization of emergency information[3]. An Incident Command Structure (ICS) can organize responders into a hierarchical structure of sections (e.g. Operations, Planning, Logistics, Finance) and roles (e.g. Incident Commander, Public Information Officer, Safety Officer)[4]. XML-based standards are being developed to improve sharing of emergency information. The Organization for the Advancement of Structured Information Systems (OASIS) has a family of standards known as the Emergency Data Exchange Language (EDXL)[5]. The Emergency Data Exchange Language Common Alerting Protocol (EDXL-CAP) exemplifies simple, useful, and understandable information-exchange formats. What EDXL-CAP did for alerts, a Common Decision Exchange Protocol (CDEP) could do for decisions [6].

An important next step is to utilize the semantic web standards, including RDF, SPARQL, OWL and GRDDL to integrate information for dynamic queries across datasets, and for inferencing using the underlying ontologies (e.g. indicating that the emergency equipment named X in one jurisdiction is the same as the type named Y in another jurisdiction). Initial steps in this direction are already being taken, e.g., through the OASIS Distribution Element (DE) supporting packaging and addressing of emergency management information for purposes such as routing. The standard has links to externally-managed 'lists' representing concepts such as 'senderRole', 'receiverRole' and 'keywords'. Ontologies should encapsulate, in a machine-understandable manner, such information sharing policies. Implicitly present is the underlying decision-making process, continuing at all levels through an emergency. The decision format advocated in this paper will support the move toward the use of linked data [1], and the recognition of the significance of information sharing policies utilizing semantic standards.

The need for representing, sharing and managing decisions in a machine-understandable format is not exclusive to emergency management. One example of another critical

[1]For more information, or to participate in the Decisions Incubator, please review the charter at http://www.w3.org/2005/Incubator/decision/charter and visit the wiki at http://www.w3.org /2005/Incubator/decision/wiki/Main_Page.

[2]For more information on emergency and incident management, see for example the National Incident Management System, December 2008, published by the U.S. Department of Homeland Security at http://www.fema.gov/pdf/emergency/nims/NIMS_core.pdf.

[3]For examples of incident command forms, see http://training.fema.gov/EMIWeb/IS/ICSResource/ICSResCntr_Forms.htm.

[4]For more information on ICS, see the online training provided by the U.S. Federal Emergency Management Agency, Lesson 3, at http://emilms.fema.gov/IS100A/indexMenu.htm.

[5]For a good overview of EDXL, see http://en.wikipedia.org/wiki/EDXL. The EDXL family of standards is available at the OASIS website: http://www.oasis-open.org/home/index.php.

domain of interest is organizational innovation. Each person is a 'decision-maker' at some level in the organization. The decisions a person makes are critical to the success of an organization, so aspects of decision-making and objective measures of the decision-making process become significant. Decisions involve weighing reasonable options based on metrics in order to take an action. If we granulize the decision-making process by considering each member of our organization as a decision-maker, then we can support the representation and sharing of individual innovative actions. Most organizations attempt to solve this problem through direct or indirect person-to-person communication (e.g., meetings, telecons) or unstructured collaborative tools (email, chat, wiki). XML formats can support notice-type publishing of activities, e.g, RSS or ATOM feeds; however, there remains an opportunity to showcase semantic standards to capture decision-making to improve the querying, inferencing, and integration with underlying ontology support.

The focus of this paper is on the information sharing aspects of a decision, which is fostered by a format which is concise, generic, i.e., domain independent, and tiered. The more concise the format, the more quickly it can be understood and accepted by developers and users alike.

1.2 Project Goals

The work performed by this incubator activity is designed to help organizations improve integration of human decisions into computer systems, to track and manage digitally the decision-making process, to enable improved information-flow metrics, to maintain an archive of the decisions and the decision-making process, to enable semi-automation of certain decision-making processes, to improve information sharing, and ultimately, to support better, rapid, and agile decision making [7]. The potential standard format should provide concise, generic, structured assessments and decisions that allow 'drill down', support pedigree and confidence, enable approvals and vetting, define options considered, including decision criteria with weighting, links to previous decisions and sub-decisions, and support flexible structuring of complex decisions [7]. However, to reach its full potential, the proposed decision format must be compatible with semantic web tools and standards, to provide semantic interoperability and to provide a basis for reasoning that can ease development of advanced applications.

In summary the main goals of the incubator are:

- To discover a set of requirements for a standard decision format, through a set of use cases.
- To develop a draft of a potential standard format for representing decisions, fulfilling the requirements of the use case and exploiting semantic web standards.

2. METHODOLOGY BACKGROUND

Creating a vocabulary for expressing decisions that exploits semantic web standards means, in practice, creating a set of ontology modules that can be linked in a network, to be used independently or together in different combinations. The main tools we use for this practical task is the eXtreme Design ontology engineering methodology and the notion of Ontology Design Patterns (ODPs), supported by the ontology development environment NeOn Toolkit[6].

[6]http://www.neon-toolkit.org

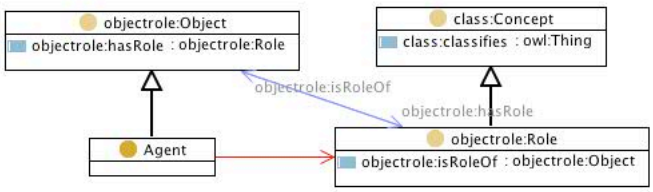

Figure 1: The AgentRole Content ODP's graphical representation in UML.

2.1 Ontology Design Patterns

Under the assumption that classes of problems in ontology design can be solved by applying common solutions (as experienced in software engineering), ODPs can support design reusability. ODPs can be of several types [3], e.g, focusing on logical language constructs, architectural issues, naming, or on the efficient provision of reasoning services. In this paper we focus on Content ODPs (CPs), which are small or cleverly modularized ontologies with explicit documentation of design rationales. CPs can be used as building blocks in ontology design [2]. As an example we describe a CP called AgentRole. It represents the relation between agents, e.g., people, and the roles they play, e.g., manager and meeting chair. Figure 1 shows the UML diagram[7] of the OWL[8] building block representing this CP.

CPs are collected in different catalogues, such as the *ODP portal*[9]. In addition to their diagrammatic representation, CPs are described using catalogue-entry fields (c.f. software pattern templates), such as *name*, *intent*, *covered requirements*, *consequences*, and *building block*, linking to an OWL realization of the pattern. The requirements an ODP covers are expressed using Competency Questions (CQs) [4], i.e., typical natural-language queries.

2.2 eXtreme Design

With the name 'eXtreme Design' (XD) we identify an agile approach to ontology engineering [5]. In this paper we focus on XD for CP reuse in ontology design. In XD a development project is characterized by two sets: (i) the problem space, composed of the actual modeling issues (local problems), e.g., to model steps in a decision making process; (ii) the solution space, made up of reusable modeling solutions, e.g., a piece of an ontology that models sequences of events (a CP). Each CP, as well as the local problem, is related to ontology requirements expressed as CQs or sentences. If a local problem can be described in terms of the CQs of a CP then that CP can be reused for building the solution. XD does not prescribe a specific method for matching the local problem to patterns, and at the moment the only tool support available are search functionalities utilizing the textual descriptions of the patterns.

XD is a test-driven and task-focused approach that results in highly modular ontologies. The main principles of XD include the intensive use of CPs, and extensive collaboration [5]. The iterative workflow of XD contains 12 steps. The project is initiated in the first four steps, which in-

[7]For notation details, see: http://www.topquadrant.com/products/TB_Composer.html

[8]http://www.w3.org/2004/OWL/

[9]http://www.ontologydesignpatterns.org

clude, scoping, and requirements engineering (e.g., deriving the CQs from user stories). In steps five through nine the CQs are divided into into small, coherent sets and ontology modules produced realize those sets of CQs. These steps include unit tests on each module before its release. The three final steps integrate modules into a coherent solution, focusing on collaboration and integration.

3. ONGOING WORK

In this section we describe our ongoing efforts and how we apply the XD methodology to support these efforts. We proceed in a bottom-up fashion, starting from the use cases and deriving requirements for a representation format that can be realized as ontology modules based on ODPs. However, we have also encountered a number of cases where this leads to the development of general ODPs themselves.

3.1 Use Cases

Use cases are in our context general scenarios, horizontal with respect to application domains (i.e., they are represented in multiple domains), where the envisioned decision format can give some substantial benefit. So far, five use cases have been identified (the list is continuously extended). The use cases are intended to be general and not domain specific, in terms of industry domain. Their detailed description, including resulting requirements in the form of CQs can be found in the Incubator wiki[10]. Background and related work for two of the use cases are described more in depth in Sections 3.1.1 and 3.1.2.

- **Measuring Information Flow** - Where a decision process representation can help answering questions such as 'When did a certain process begin and end?', 'How much time was spent on a certain step in the process?', and 'What is the average time for making a certain type of decision?'.
- **Linked Data Supporting Decisions** - Where linked data [1] supports decision making, and a decision representation format could help answer questions such as 'What data support this decision?', 'What were the options and the criteria used for this decision?', and 'How were the options assessed?'
- **Automatic Assessment of Options** - Where a decision format is intended to support semi-automatic decision making by automatic assessment of options through some metric. In this case questions are for instance 'What are the metrics for this decision and to what options do they apply?', 'What are the relative weights of different metrics?', and 'How will the metrics be combined to generate an overall assessment?'
- **Interoperability** - For example, a shared decision representation can support interoperation between different command and control units and between decision makers and people implementing decisions.
- **Situational Awareness** - A representation of decisions and the decision-making process can support systems and/or organizations to be aware of the decision status, to identify situations, such as the situation when important information is missing, and to base new decisions on the collected knowledge in the recorded decisions of the organization.

3.1.1 Measuring Information Flow

Research shows that an analytical solution of information velocity is intractable but metrics that support the understanding of information flow can be useful [8]. An agent-based model for information flow can be used to characterize physical analogs to causal measures [6]. In this use case, interactions and exchanges can be modeled as physical properties. Information, its suppliers, and consumers are then treated as agents. The behavior of the agents and system as a whole can be discussed and infodynamic analogs of thermodynamic and other physical quantities associated with these processes could be explored [8]. The use of conceptual analogs from the physical domain implies the viability of future ontologies to characterize information flow.

3.1.2 Automatic Assessment of Options

Design considerations have been described and exemplified for implementing a decision-acquisition system based on a CDEP [7]. CDEP is an XML- and REST-based protocol for representing generic human decisions in a simple, interoperable format. The characteristics of decisions can be expressed using CDEP and its proposed XML format [7]. The CDEP concepts will be considered, and enhanced, within the currently envisioned decision format, and a conversion XSLT stylesheet will enable interoperability across these formats as needed. The use case ontology would allow for the consideration of multiple data sources, multiple decision options, and the tracking of decision confidence throughout the decision-making process.

3.2 Decision Patterns

The decision patterns include concrete decision format components, as well as generic patterns, hence, both:

- The ontology modules that we propose as a starting point for creating a standard in this field,
- and the more general ODPs that we discover and develop as a result of this effort.

The first module draft that was produced corresponds to the use case of 'Measuring Information Flow' listed above. This ontology module is a specialization of the Transition ODP[11]. In this case we found an ODP already available that we could specialize and create a specific decision-process pattern. In other cases, such as when viewing a decision as a past event, no 'event-pattern' was available in the ODP portal. Therefore, we are creating general ODPs to be specialized in the decision ontology modules. By treating general (rather than domain-specific) use cases of decision-making, we make sure that the developed modules are actually reusable patterns, rather than a solution tailored to one specific application. All decision patterns will be implemented in RDF/OWL. Eight patterns are identified so far, but need to be created. Four examples are described below:

- A 'Statement with variable'-pattern, to describe queries, such as the question underlying a decision.
- 'Filter' and 'Aggregation'-patterns, where a filter would represent criteria applicable to some data, e.g., a set

[10] http://www.w3.org/2005/Incubator/decision/wiki/Final_Report_Use_Cases

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

of options, and an aggregation would represent a way to combine data, e.g., grouping options.

- A 'Normalization'-pattern that models transformations of values into a common scale, for comparing options.
- A 'Weighting'-pattern to express the relative importance of data, e.g., weighting of assessment criteria.

3.3 Proof-of-concept Application

To verify the requirements and the ontology modules, and to demonstrate the usefulness of such a format, a demonstration system is being developed at the Space and Naval Warfare Systems Center Pacific. Initially, the system will focus on enabling decision making using open linked data sets [1]. The user has four modules, or screens. In the Topic screen, the user enters the key question of the decision, keywords, and where the decision result will appear. The keywords will drive a search for relevant open-linked data sets. Next, the user selects a data set from which the entries provide a named set of options. From the Options screen, the user selects the properties to use as metrics. On the Metrics screen, the user selects filtering criteria to reduce the options. The user can additionally assign weights to the metrics. When a similar decision is encountered, users can efficiently select a named set of Options or Metrics to aid reuse of decision components. A semi-automatic learning process will be considered for future releases, proposing named sets of options or metrics found useful to other users, based on similarity of questions and keywords. On the Assessment screen, the filtered options appear in an ordered list based on the weighted metrics. The user selects one or more options as the answer to the decision question. The user is returned to the Topic screen where the answer(s) is/are recorded and visible. Throughout the process, the time spent in the various stages is tracked to assess information flow. Future versions of this system will support manual entry of decisions, a more robust set of filtering criteria, integration of multiple datasets, and mobile applications for efficiency in the field. The decision format discussed here will be used to manage the decision as a whole, and its modular components.

3.4 Experiences

An important outcome, apart from the requirements and a proposed decision representation, will be experiences related to the XD methodology and ODPs. XD has been used in the project both as a framework for the modelling but also as a means for teaching ontology engineering to participants less familiar with semantic technologies. So far we found that the level of detail of the XD methodology is highly beneficial for teaching ontology engineering to novice modelers. It introduces an intuitive way of scoping the problem, through modularization, and it allows the modeler to draw on previous experiences of others through ODPs. We envision that the project will benefit the further development of XD, and XD will be validated through valuable experiences.

4. OUTLOOK

In September 2010, the project reached its half-way point and should be completed by the end of March 2011. By that time the project will have a set of requirements for a potential decision-representation standard, i.e., the use cases (initial set in Section 3.1), and a first draft of such a representation, i.e., the decision patterns (initial ideas in Section 3.2). We intend to submit any patterns developed (both general and specific to decision-making) to the ODP portal. We expect to present a set of proof-of-concept applications (see Section 3.3). These applications will show the potential of our draft patterns. The applications will be used to validate our results against current practices in different domains, e.g., to validate the hypothesis that linked data are suitable to support decision making and that automatic assessment of options is possible in certain use cases. During the project, we will make the problems and possible solutions visible in different communities, e.g., the semantic web community, domain specific interest groups, and standards organizations. We envision that at the end of the project we can propose a standardization effort in the context of W3C. We can pursue several use cases and application ideas as separate research projects.

5. ACKNOWLEDGMENTS

The authors thank the Office of Naval Research for their support of this work. This paper is the work of U.S. Government employees performed in the course of their employment and no copyright subsists therein.

6. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[2] E. Blomqvist. OntoCase-Automatic Ontology Enrichment Based on Ontology Design Patterns. In *ISWC 2009, 8th International Semantic Web Conference*, volume 5823 of *LNCS*, pages 65–80, Washington, DC, November 2009. Springer.

[3] A. Gangemi and V. Presutti. Ontology Design Patterns. In *Handbook on Ontologies, 2nd Ed.*, International Handbooks on Information Systems. Springer, 2009.

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

[5] V. Presutti, E. Daga, A. Gangemi, and E. Blomqvist. eXtreme Design with Content Ontology Design Patterns. In *Proc. of WOP 2009, collocated with ISWC-2009*, volume 516. CEUR Workshop Proceedings, November 2009.

[6] J. Waters and M. Ceruti. Modeling and simulation of information flow: A study of infodynamic quantities. In *Proc. of the 15th International Command and Control Research and Technology Symposium (ICCRTS 2010)*, Santa Monica, CA, June 2010.

[7] J. Waters, M. Ceruti, R. Patel, and J. Eitelberg. Decision-acquisition system based on a common decision-exchange protocol. In *Proc. of the 15th International Command and Control Research and Technology Symposium (ICCRTS 2010)*, Santa Monica, CA, June 2010.

[8] J. Waters, R. Patel, J. Eitelberg, and M. Ceruti. Information velocity metric for the flow of information through an organization: Application to decision support. In *Proc. of the 14th ICCRTS (ICCRTS 2009)*, Washington, DC, June 2009.

Pattern Abstracts

Context Slices: Representing Contexts in OWL
http://ontologydesignpatterns.org/wiki/Submissions:Context_Slices

Chris Welty
IBM Watson Research
Hawthorne, NY 12540, USA
cawelty@gmail.com

ABSTRACT
This ontology pattern can be used to represent and reason about contextualized statements using standard OWL dialects. The simple idea is to bundle the notion of context into certain nodes in the graph, rather than the more typical treatment of contexts as a property of the statements themselves.

Keywords
Semantic Web, OWL, RDF, Contexts.

1. INTRODUCTION
Most information on the web is contextualized somehow, for example information may be believed by a person or organization, it may hold only for some time period, it may have been reported/observed by an individual, etc. There are myriad proposals and logics for context, but none are standards and few have even prototype implementations.

In RDF and other binary relation languages (like object oriented languages and description logics), one typical way to represent that a binary relation holds in some context is to "reify" the relation-holding in the context as an object with a binary relation between the obtainment and each the two relation arguments and a third binary relation between the obtainment and an object representing the context itself. The downside to this approach is the expressive ability of the language to describe the binary relation, especially in the case of description logics, is lost. One can of course use RDF reification, however this is not supported in OWL, either.

The motivation for context slices is to provide a logical pattern for encoding context information in standard RDF graphs that allows some of the expressiveness of OWL to be used in describing the relations that hold in contexts.

This is a generalization of the four dimensional ontology for fluents published in [1].

2. PATTERN DESCRIPTION
The idea of the context slices pattern is, rather than reifying the statement itself, to create a projection of the "relation arguments" in each context for which some binary relation holds between them.

Take for example the statement "Chris believes Sam is CEO of IBM". Say we already have nodes in some graph representing Sam, Chris and IBM. We create, as shown in Figure 1, the context c_1 corresponding to Chris' belief, and two nodes representing Chris' belief about Sam and Chris' belief about IBM (shown as Sam@c_1 and IBM@c_1).

This allows us to represent ceoOf as a binary relation, which seems more natural, and it allows us to use the expressivity of OWL in more ways. We can say of the ceoOf relation that it has an inverse, hasCeo. We can express cardinality, e.g., a company may have only one CEO within a context. We can say that a relation is transitive or symmetric. We can express relation taxonomies in the usual way.

While clearly OWL does not support RDF reification, and so none of this is possible if statement reification is used. As mentioned above a more standard way of representing this kind of information (including time, belief, knowledge, etc.) is to create an OWL class that represents the relation holding, with properties for the arguments. This approach makes it possible to express global but not local range and domain constraints, global but not local cardinality, and symmetry.

Note that the ContextualProjection class should be considered disjoint with any of the classes in an ontology that have projections.

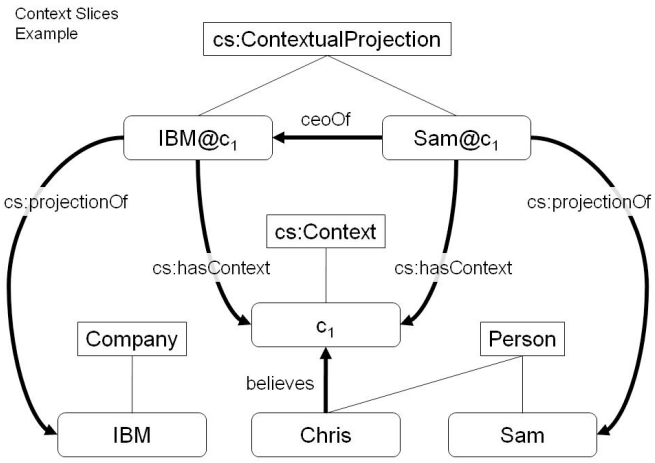

Figure 1: Graphical illustration of an example using the pattern.

3. IMPLEMENTATION
In OWL functional syntax:

```
Ontology(<http://example.org/ContextSlices>
    Annotation(owl:versionInfo "1.0"@en)
    Annotation(rdfs:label "Context slices ontology logical
        pattern"@en)
```

```
Declaration(Class(cs:Context))
DisjointClasses(cs:Context cs:ContextualProjection)
Declaration(Class(cs:ContextualProjection))
SubClassOf(cs:ContextualProjection
    ObjectAllValuesFrom(cs:hasContext cs:Context))
SubClassOf(cs:ContextualProjection
    ObjectExactCardinality(1 cs:hasContext))
SubClassOf(cs:ContextualProjection
    ObjectExactCardinality(1 cs:projectionOf))
DisjointClasses(cs:ContextualProjection cs:Context)
Declaration(ObjectProperty(cs:contextualProperty))
ObjectPropertyDomain(cs:contextualProperty
    cs:ContextualProjection)
ObjectPropertyRange(cs:contextualProperty
    cs:ContextualProjection)
Declaration(ObjectProperty(cs:hasContext))
FunctionalObjectProperty(cs:hasContext)
ObjectPropertyDomain(cs:hasContext
    cs:ContextualProjection)
Declaration(ObjectProperty(cs:projectionOf))
FunctionalObjectProperty(cs:projectionOf)
ObjectPropertyDomain(cs:projectionOf
    cs:ContextualProjection))
```

4. REFERENCES

[1] Welty, Chris and Richard E. Fikes. 2006. A Reusable Ontology for Fluents in OWL. In Bennet and Fellbaum, eds., *Proceedings of the Fourth International Conference on Formal Ontology in Information Systems*. IOS Press. See http://www.booksonline.iospress.nl/Content/View.aspx?piid=2209.

Faceted Classification Scheme ODP

http://ontologydesignpatterns.org/wiki/Submissions:Faceted_Classification_Scheme

Bene Rodriguez-Castro
School of Electronics and Computer Science
University of Southampton
Southampton, UK
b.rodriguez@ecs.soton.ac.uk

Hugh Glaser
School of Electronics and Computer Science
University of Southampton
Southampton, UK
hg@ecs.soton.ac.uk

Les Carr
School of Electronics and Computer Science
University of Southampton
Southampton, UK
lac@ecs.soton.ac.uk

Keywords

Faceted Classification, Normalisation, Multiple Classification Criteria

1. INTRODUCTION

The Faceted Classification Scheme (FCS) ODP is a Reengineering ODP that transforms a non-ontological resource from the field of Library and Information Science, also known as Faceted Classification Scheme, into an ontological resource. The ontological resource corresponds to an OWL DL model that results from a specific application of the Normalisation ODP [4] [2] based on a series of (a) alignments between the two conceptual models; and (b) transformation guidelines.

The FCS ODP targets a specific, very recurrent modeling issue in ontology development, subject to the vulnerability of ad-hoc modeling practices that could potentially lead to unexpected or undesirable results in ontology artifacts. The scenario consists of domain-specific concepts that can be represented according to multiple alternative classification criteria. To the best of our knowledge, guidelines for the conceptualization and representation of domain-specific concepts prone to be described based on multiple (potentially alternative) classification criteria, has not been explicitly considered in the context of ontology modeling for the Semantic Web.

An extended and detailed version of all the sections that follow and the rationale behind the FCS ODP is presented at length in [5].

2. PATTERN DESCRIPTION

A FCS is defined as: "a set of mutually exclusive and jointly exhaustive categories, each made by isolating one perspective on the items (a *facet*), that combine to completely describe all the objects in question, and which users can use, by searching and browsing, to find what they need" [1].

The Norm. ODP is classified as a "Good Practice" pattern in the catalog of ODPs introduced in [2]. It can be applied to any OWL DL ontology that consists of a polyhierarchy where some *semantic axes* can be pointed. Each of those axes will be a *module*.

The key similarity between these two conceptual models, lies in the notion of (a) facet in FCSs; and (b) module (or semantic axis) in the Norm. ODP. Both elements represent one perspective of the domain being modelled, a single characteristic of division, a single criterion of classification in their respective paradigm.

Library Sc.	**Ontology Modeling**	
FCS	**FCS ODP**	**OWL Impl.**
TDC	$:TDC$	owl:Class (primitive)
$Facet_i$	$:Facet_i$	owl:Class (primitive)
	$:hasFacet_i$	owl:ObjectProperty
F_iTerm_j	$:F_iTerm_j$	owl:Class (primitive)
	$:F_iTerm_jTDC$	owl:Class (def.) (≡)
$Item_x$	$:SpecificTDC_x$	owl:Class (primitive)

Table 1: Alignment of a FCS to the Norm. ODP

```
owl:Thing
   |-- :Facet_i
      |-- :F_iTerm_j
   |-- :TargetDomainConcept (or :TDC)
      |-- (≡) :F_iTerm_jTDC
      |-- :SpecificTDC_x

owl:topObjectProperty
   |-- :hasFacet_i
```

(≡) denotes a defined owl:Class.

Figure 1: FCS elements placed into the Norm. ODP

The main principle is to represent each facet as an independent module or semantic axis. Following this principle makes the application of the Norm. ODP *almost* straightforward. Moreover, the resultant ontology includes the representation of the multiple alternative classification criteria that were considered in the original FCS for the target domain concept.

Table 1 summarizes the alignment of the elements in the generic structure of both conceptual models. This alignment enables the conversion of a FCS into an OWL DL ontology by applying the Norm. ODP, where:

- TDC denotes the target domain concept (or domain of discourse) of the FCS.
- $Facet_i$ denotes one of the facets of the FCS.
- F_iTerm_j denotes one of the terms of $Facet_i$.
- $Item_x$ denotes one the items from the domain of discourse to be classified.

Figure 1 depicts the placement of the elements of a generic FCS into the generic structure of the Norm. ODP based on the corresponding mappings from Table 1.

```
Agent: dishwasher, person

Form: gel, gelpac, liquid, powder, tablet

Brand Name: Cascade, Electrasol, Ivory, No Name,
            Palmolive, President's Choice, Sunlight

Scent: green apple, green tea, lavender, lemon,
       mandarin, ocean breeze, [...]

Effect on Agent: aroma therapy (subdivisions:
                 invigorating, relaxing)

Special Property: antibacterial
```

Figure 2: Example of "Dishwashing Detergent" FCS.

3. PATTERN USAGE EXAMPLE

Figure 2 presents the facets and terms of a FCS example in the domain of "Dishwashing Detergent" from [1].

To apply the FCS ODP, the elements in the generic ontology structure (derived from the Norm. ODP) in Fig. 1 are populated with the facets and terms of the "Dishwashing Detergent" FCS example in Fig. 2, according to the alignments specified in Table 1. The overall normalised ontology model obtained as a result is presented in Fig. 3. A version of the complete normalised ontology model for the "Dishwashing Detergent" FCS example in [1] is available online[1] in RDF/XML format.

4. RELATED WORK

The FCS ODP considered previous work that defined mappings between different semantic models and OWL ontologies such as the Resource Space Model (RSM) [6] and the concept of Faceted Lightweight Classification Ontology [3]. A detailed discussion is available in [5].

5. CONCLUSIONS

The FCS ODP has presented an initial set of basic design guidelines to develop an OWL DL ontology model that supports the representation of multiple alternative classification criteria of a specific domain concept. These guidelines provides a *partial* solution to potentially hazardous ad-hoc practices in the development of such ontology models, putting forward a systematic and fit-for-purpose approach.

6. REFERENCES

[1] W. Denton. How to make a faceted classification and put it on the web. Online, November 2003. http://www.miskatonic.org/library/facet-web-howto.html.

[2] M. Egana-Aranguren. *Role and Application of Ontology Design Patterns in Bio-ontologies*. PhD thesis, School of Computer Science, University of Manchester, 2009.

[3] F. Giunchiglia, B. Dutta, and V. Maltese. Faceted lightweight ontologies. In A. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. K. Yu, editors, *Conceptual Modeling: Foundations and Applications*, volume 5600 of *Lecture Notes in Computer Science*, pages 36–51. Springer, 2009.

[1] http://purl.org/net/project/enakting/ontology/detergent_fcs_norm

```
owl:Thing
   |-- :Agent
      |-- :Person
      |-- :Dishwasher
   |-- :Form
      |-- :Gel
      |-- :Gelpac
      |-- (... rest of terms in the facet "Form")
   |-- :BrandName
      |-- :Cascade
      |-- :Electrasol
      |-- (... rest of terms in the facet "Brand Name")
   |-- :Scent
      |-- :GreenApple
      |-- :GreenTea
      |-- (... rest of terms in the facet "Scent")
   |-- :EffectOnAgent
      |-- :AromaTherapy
         |-- :Invigorating
         |-- :Relaxing
   |-- :SpecialProperty
      |-- :Antibacterial
   |-- :DishwashingDetergent (:TDC)
      |-- (≡) :ManualDishDetergent
      |-- (≡) :DishwasherDishDetergent
      |-- (≡) :GelDishDetergent
      |-- (≡) :GelpacDishDetergent
      |-- (≡) (... rest of subclasses for each term
                  in the facet "Form")
      |-- (≡) :CascaseDishDetergent
      |-- (≡) :ElectrasolDishDetergent
      |-- (≡) (... rest of subclasses for each term
                  in the facet "Brand Name")
      |-- (≡) :GreenAppleDishDetergent
      |-- (≡) :GreenTeaDishDetergent
      |-- (≡) (... rest of subclasses for each term
                  in the facet "Scent")
      |-- (≡) :AromaTherapyDishDetergent
         |-- (≡) :InvigoratingDishDetergent
         |-- (≡) :RelaxingDishDetergent
      |-- (≡) :AntibacterialDishDetergent
      |-- :PresidentsPersonLiquidAntibacterial
      |-- :PalmoliveAromaTherapyLavenderYlangYlang
      |-- :SpecificDishDetergent3
      |-- (... rest of specific dish detergent classes
              :SpecificDishDetergent_x to classify)

owl:topObjectProperty
   |-- :hasAgent
   |-- :hasForm
   |-- :hasBrand
   |-- :hasScent
   |-- :hasEffectOnAgent
   |-- :hasSpecialProperty
```

(≡) denotes a defined owl:Class.

Figure 3: Normalised ontology structure of the "Dishwashing Detergent" FCS.

[4] A. L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, pages 121–128, New York, NY, USA, 2003. ACM.

[5] B. Rodriguez-Castro, H. Glaser, and L. Carr. How to reuse a faceted classification and put it on the semantic web. In *The 9th International Semantic Web Conference (ISWC)*, June 2010.

[6] H. Zhuge, Y. Xing, and P. Shi. Resource space model, owl and database: Mapping and integration. *ACM Trans. Internet Technol.*, 8(4):1–31, 2008.

Summarization of an inverse n-ary relation

http://ontologydesignpatterns.org/wiki/Submissions:Summarization_of_an_inverse_n-ary_relation

María Poveda Villalón
Ontology Engineering Group (OEG)
Departamento de Inteligencia Artificial. Facultad de Informática. Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
+34 913363670
mpoveda@delicias.dia.fi.upm.es

Mari Carmen Suárez-Figueroa
Ontology Engineering Group (OEG)
Departamento de Inteligencia Artificial. Facultad de Informática. Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
+34 913363672
mcsuarez@fi.upm.es

ABSTRACT
In this paper, we describe a logical ontology design pattern that summarizes a relationship and its inverse between two distinguished members of an n-ary relationship.

Keywords
Ontology design pattern, N-ary relation, inverse relation.

1. INTRODUCTION
In Semantic Web languages such as RDF and OWL, a property is a binary relation. This binary relation is used to link two individuals or an individual and a value. In some cases, however, the natural and convenient way to represent certain situations is to use relations and to link an individual to more than just one individual or value. These relations are called *n-ary relations*[1].

The n-ary relations become even more complex if we pretend to represent inverse relationships[2] between all the participants in the n-ary relation. However, we might have special interest in the links between two of the participants involved in the relationship, and not have interest in all of them. For this reason, we propose a pattern to speed up both the modelling and the queries of a relationship between two distinguished participants in an n-ary relation, and its inverse relationship.

2. PATTERN DESCRIPTION

2.1 Motivation
It is well known that an n-ary relationship should be used to address any of the following situations [1]:

a) A binary relationship that really needs a further argument. For example, to represent the distance between two places.

b) Two binary relationships that always go together and should be represented as one n-ary relation. For example, to represent the value of an observation (e.g. temperature in a patient) and its trend.

c) A relationship that is really amongst several things. For example, to represent the spatial location of a person in a given point of time.

On the one hand, the motivation of this pattern is to express the inverse relationship of an n-ary relation in which there are two distinguished participants. This means that the relationship exists mainly between two entities and the rest of entities involved in the relationship can be considered as additional arguments. This situation can also mean that there is a single individual standing out as the subject or the "owner" of the relation.

On the other hand, the motivation is to provide a shortcut for queries that involve two distinguished participants in the n-ary relationship.

This pattern is inspired on the third consideration shown in the description of n-ary relations[3] from the W3C Semantic Web Best Practices Group (SWBP Group). The difference in our case is that there are two distinguished participants in the relationship. Therefore, this pattern could be considered as an extension of the third consideration shown by the SWBP Group applied to the use case of additional attributes describing a relation[4].

2.2 Aim
The aim of this pattern is to allow asking for n-ary relationships and their inverse relations between two distinguished participants without a complex query. Such a complex query would involve the class created to support the n-ary relation between the origin and destination classes of the n-ary relationship.

2.3 Solution description
As it can be observed in Figure 1 the class "NAryRelationClass" is the class created to support the n-ary relationship[5] and its further relations or attributes. The relationship "mainRelationship" and its inverse relation have been created to

[1] http://www.w3.org/TR/swbp-n-aryRelations/

[2] http://www.w3.org/TR/swbp-n-aryRelations/#choosingPattern1or2

[3] http://www.w3.org/TR/swbp-n-aryRelations/#choosingPattern1or2

[4] http://www.w3.org/TR/swbp-n-aryRelations/#useCase1

[5] This structure is created like in [1] and http://www.w3.org/TR/swbp-n-aryRelations/#useCase1

short-circuit the relation between the distinguished participants in the n-ary relationship.

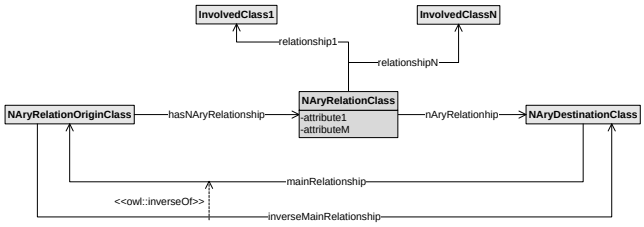

Figure 1. Graphical representation of the "Summarization of an inverse n-ary relation" pattern.

3. Example

3.1 Problem example

We might want to represent that a service provider provides a service at a place in a given period of time with a particular price. The model should also represent that a service is offered by a provider.

In this scenario, we have also observed that the queries executed by our applications often ask for the relationship between providers and their services and rarely ask for the relationships about the services and where they are provided.

Figure 2 depicts the result of applying the "Summarization of an inverse n-ary relation" pattern to represent the abovementioned problem.

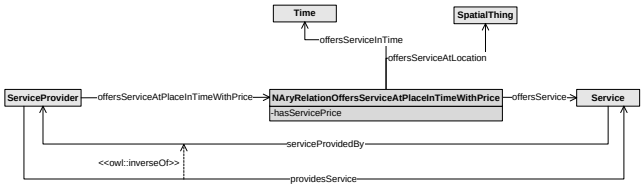

Figure 2. "Summarization of an inverse n-ary relation" pattern applied to service providers.

3.2 Consequences

The main advantage of this pattern is that allows asking for those services that are provided by a service provider and vice-versa without a complex query. This complex query would involve the class created to support the n-ary relation between service providers and services.

4. Related Work

The origin of this pattern is the *Logical Pattern for Modelling N-ary Relation: Introducing a New Class for the Relation* pattern[6] and the third consideration shown in the description of n-ary relations from the W3C SWBP Group. Therefore, this pattern is related to and can be used in combination with the *Logical Pattern for Modelling N-ary Relation: Introducing a New Class for the Relation*.

5. Summary and Outlook

The *Summarization of an inverse n-ary relation* pattern allows us to speed up the queries involving relationships between two distinguished participants in an n-ary relation.

Future lines of work will address the problem of summarizing the relationships and their inverse between a set of distinguished member (at least three) into an n-ary relationship.

In addition, the elaboration of guidelines that explain in detail how to identify the distinguished members in an n-ary relationship would be very useful to extend the pattern description and to facilitate its use.

6. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish project *mIO!* (CENIT-2008-1019).

7. REFERENCES

[1] Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.. *NeOn D5.1.1: NeOn Modelling Components*. NeOn project. http://www.neon-project.org. March 2007

[6] http://www.w3.org/TR/swbp-n-aryRelations/#pattern1

Literal Reification

http://ontologydesignpatterns.org/wiki/Submissions:Literal_Reification

Aldo Gangemi
ISTC-CNR
aldo.gangemi@cnr.it

Silvio Peroni
University of Bologna
speroni@cs.unibo.it

Fabio Vitali
University of Bologna
fabio@cs.unibo.it

ABSTRACT

In this paper we introduce the pattern *literal reification*, a modelling technique to address scenarios, in which we need to bless particular literals, usually when applying data properties, in order to use them as subjects and/or full-fledged objects of semantic assertions.

Keywords

OWL, SWRL, literal reification

1. INTRODUCTION

Recently within the Semantic Web community a new topic has been actively discussed: whether and how to allow literals to be subjects of RDF statements[1]. This discussions failed to provide a unique and clear indication of how to proceed in that regard.

Although one of the suggestions coming out of the discussion was to explicitly include the proposal in a (future) specification of RDF, this topic is not actually new, particularly in ontology modelling. The needs to describe "typical" literals (specially strings) as individuals of a particular class has been addressed by a lot of models in past, such as Common Tag[2] (through the class *Tag*), SIOC[3] (through the classes *Category* and *Tag*), SKOS-XL[4] (through the class *Label*), and LMM[5] (through the class *Expression*). After considering the above-mentioned models and other related and inspiring ones, we have developed a pattern called *literal reification* to address this issue. It allows to express literal values as proper ontological individuals so as to use them as subject/object of any assertion within OWL models.

The rest of the paper follows this structure: in Section 2 and Section 3 we respectively introduce a high level and detailed description of the pattern; in Section 4 we discuss two particular application scenarios that we use to demonstrate all the capabilities of the pattern.

2. GENERAL DESCRIPTION

Extending the pattern *region*[6], the pattern *literal reification* promotes any literal as "first class object" in OWL by reifying it as a proper individual of the class *litre:Literal*. Individuals of this class express literal values through the functional data property *litre:hasLiteralValue* and can be connected to other individuals that share the same literal value by using the property *litre:hasSameLiteralValueAs*. Moreover, a literal may refer to, and may be referred by any OWL individual through *litre:isLiteralOf* and *litre:hasLiteral* respectively.

Note that the pattern defines also a SWRL rule that allows to infer the (not explicitly asserted) literal value of a particular literal individual when it is connected to another literal individual via *litre:hasSameLiteralValueAs*:

```
litre:hasSameLiteralValueAs(x,y) ,
litre:hasLiteralValue(y,v)
  -> litre:hasLiteralValue(x,v)
```

This pattern allows to use each reified literal as subject or object of any assertion, and it is able to address scenarios described, for example, by the following competency questions:

- What is the context in which entities refer to a particular literal value?
- What is the meaning of a particular value considering the context in which it is used?

Plausible scenarios of its application include:

- modelling domains concerning descriptive tags, in which each tag may have more than one meaning depending on the context in which it is used;
- extending quickly the capabilities of a model by adding the possibility to make assertions on values, previously referred through data properties, without modifying it.

3. ELEMENTS

As shown in Fig. 1, the pattern *literal reification* is composed by a class, a data property and three object properties, described as follows:

- Class *litre:Literal*. It describes reified literals, where the literal value they represent is speficied through the property *litre:hasLiteralValue*. Each individual of this class must always have a specified value.
- Data property *litre:hasLiteralValue*. It is used to specify the literal value that an individual of *litre:Literal* represents.

[1] http://www.w3.org/2001/sw/wiki/Literals_as_Subjects.
[2] http://www.commontag.org
[3] http://rdfs.org/sioc/spec
[4] http://www.w3.org/TR/skos-reference/#xl
[5] http://www.ontologydesignpatterns.org/ont/lmm/LMM_L1.owl
[6] http://ontologydesignpatterns.org/wiki/Submissions:Region

- Object property *litre:hasSameLiteralValueAs*. It relates the reified literal to another one that has the same literal value.
- Object property *litre:hasLiteral*. It connects individuals of any class to a reified literal.
- Object property *litre:isLiteralOf*. It connects the reified literal to the individuals that are using it.

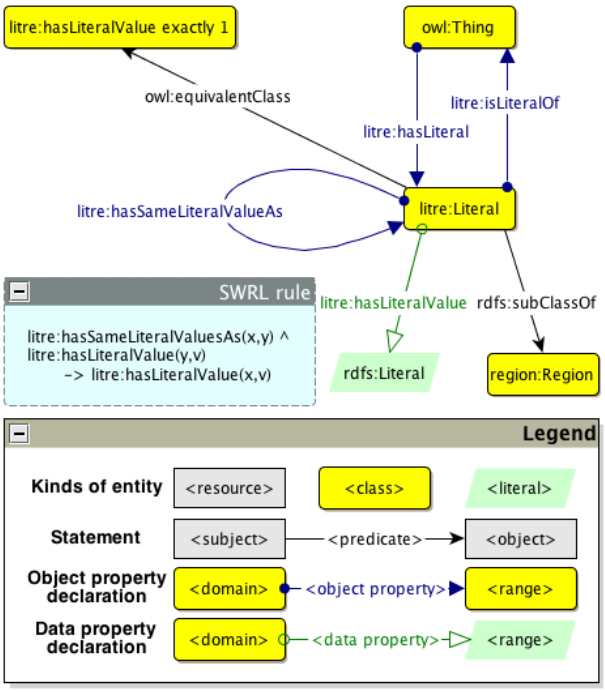

Figure 1: A figure summarizing the pattern.

4. SCENARIOS

4.1 Same tag, different meanings

Used frequently in the Web 2.0, descriptive tags such as the ones used in folksonomies are keywords (e.g., strings) assigned to a particular resource, such as a web document, with the intent to describe it. Just like words in any natural language, tags may have different meanings depending on the context in which they are used.

For instance, the word "Paris" may be either a name of a city or a first name of a person. Here, it is clear that the act of tagging with "Paris" both the Wikipedia pages about the Eiffel Tower and the one about Paris Hilton hides two different intents: in the former case, "Paris" denotes the city in which the tower stands; in the latter case, "Paris" denotes a particular person, i.e., Paris Hilton.

Using the literal reification pattern it is possible to express descriptive tags as first class objects in OWL, by considering them as proper individuals of the class *litre:Literal*. Different individuals may thus represent different meanings even if their literal values are identical[7]:

```
<http://en.wikipedia.org/wiki/Eiffel_Tower>
  a foaf:Document
  ; prism:keyword :parisTag1 .

<http://en.wikipedia.org/wiki/Paris_Hilton>
  a foaf:Document
  ; prism:keyword :parisTag2 .

:parisTag1 a litre:Literal
    , [ a skos:Concept
      ; skos:definition "A name associated
        to a city"@en ]
  ; litre:hasLiteralValue "Paris"
  ; lmm:denotes dbpedia:Paris .

:parisTag2 a litre:Literal
    , [ a skos:Concept
      ; skos:definition "A first name of
        a person"@en ]
  ; litre:hasSameLiteralValueOf :parisTag1
  ; lmm:denotes dbpedia:Paris_Hilton .
```

4.2 Keeping track of name changes

NameHistory3.0 is a (fictional) institution that keeps track of all the names of people, and stores them as an ABox of the FOAF ontology. In particular, each person is stored as an individual of the class *foaf:Person* with a specific first name (data property *foaf:givenName*) and family name (data property *foaf:familyName*).

On 24/09/2010, Bruce Wayne formally applied for changing his first name to Jack. Since NameHistory3.0 has to keep track of everything concerning names of people, on that date "Jack" was added as Mr. Wayne's first name. It was then that NameHistory3.0 noticed that, without any additional information, it is not possible to know which of the two first names are legally valid at any given point in time.

A solution to that scenario, which avoids any modification of the ontology model and consequently of the entire triple store (operation that is obviously time-consuming and error-prone), is to use the literal reification pattern in combination with the new expressivity for *punning* in OWL 2. Through them, it is possible to define a literal individual as also belonging to the class *foaf:givenName* – that is actually defined as a data property, but may be additionally be meta-modelled as a class. We can now associate a particular time interval to each literal, so as to represent when the literal itself, i.e., the given name, is legally valid[8]:

```
:mr_wayne a foaf:Person
  ; foaf:familyName "Wayne"
  ; litre:hasLiteral
    [ a litre:Literal , foaf:givenName
      ; litre:hasLiteralValue "Bruce"
      ; dcterms:valid
        [ a ti:TimeInterval
          ; ti:hasIntervalStartDate
            "1983-01-15"
          ; ti:hasIntervalEndDate
            "2010-09-24"] ]
  ; litre:hasLiteral
    [ a litre:Literal , foaf:givenName
      ; litre:hasLiteralValue "Jack"
      ; dcterms:valid
        [ a ti:TimeInterval
          ; ti:hasIntervalStartDate
            "2010-09-24" ] ] .
```

[7]http://www.essepuntato.it/2010/06/sc1.ttl

[8]http://www.essepuntato.it/2010/06/sc2.ttl

SimpleOrAggregated

http://ontologydesignpatterns.org/wiki/Submissions:SimpleOrAggregated

María Poveda Villalón
Ontology Engineering Group (OEG)
Departamento de Inteligencia Artificial. Facultad de Informática, Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
+34 913363670
mpoveda@delicias.dia.fi.upm.es

Mari Carmen Suárez-Figueroa
Ontology Engineering Group (OEG)
Departamento de Inteligencia Artificial. Facultad de Informática, Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
+34 913363672
mcsuarez@fi.upm.es

ABSTRACT

In this paper, we describe a content ontology design pattern to represent objects that can be simple or aggregated. The aggregation relation refers to several objects gathered in another object acting as a whole; all these objects should belong to the same concept in the model.

Keywords

Ontology design patterns, mereology, aggregation.

1. INTRODUCTION

Mereological relationships are one of the basic structuring primitives of the universe, and many applications require representations of them (catalogues of parts, fault diagnosis, anatomy, geography, etc.) [3].

We usually have the need of representing objects that are made up of other types of object. In these situations, we can use the part-of [1] pattern to represent transitive mereological relationships. Some examples can be "Brain and heart are parts of the human body" or "Substantia nigra is part of brain". In addition, we can use the componency [1] pattern to distinguish between parts and proper parts in a non transitive fashion. An example of this case can be "The turbine is a proper part of the engine; both are parts of a car. Furthermore, the engine and the battery are proper parts of the car".

However, sometimes we need to represent objects that can be made up of objects that belong to the same concept. In these cases it is also need to distinguish objects into simple or aggregated ones. For this reason, we have created the *SimpleOrAggregated pattern* to represent aggregation relationships, both transitive and non transitive, between objects that belong to the same concept in the model. An example of this situation can be "aggregated service provider is formed by simple or aggregated service providers".

2. PATTERN DESCRIPTION

2.1 Intent

The goal of this pattern is to represent objects that can be simple or aggregated (that is, several objects gathered in another object acting as a whole).

The main difference between the aggregation relation and other mereological relationships (such as part-of or componency) is that the aggregated object and its aggregated members should belong to the same concept.

2.2 Solution Description

As it can be observed in Figure 1 the class "ObjectByCardinality" has been created to classify simple and aggregated objects into its subclasses "SimpleObject" and "AggregatedObject", respectively. These subclasses are disjoint among them.

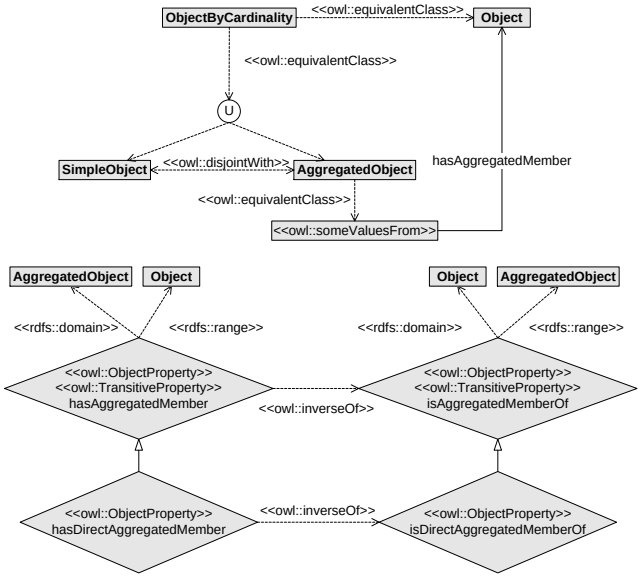

Figure 1. Graphical representation of the SimpleOrAggregated pattern.

The aggregation relationship between objects means that objects of a class can be composed by other objects of the same class. This relationship is represented by the transtive property "hasAggregatedMember" and its inverse property "isAggregatedMemberOf". These properties have as subproperties the non transitive properties "hasDirectAggregatedMember" and its inverse "isDirectAggregatedMemberOf", respectively. By means of this structure of properties, we provide a mechanism (a) to represent transitive aggregation relationships (that is, if A has B as aggregated member and B has C as aggregated member then A has C as aggregated member) and (b) to link each aggregated

member just to the next level (that is, A has B as direct aggregated member).

Finally, the class "AggregatedObject" has been defined as equivalent to those things that have some values for the property "hasAggregatedMember". This modelling allows the automatic classification of aggregated objects in this class when a reasoner is applied.

2.3 Consequences

This content pattern allows designers to represent both simple individuals of a given concept (that is, an individual that is made up of itself) and aggregated individuals of a given concept (that is, an individual that is made up of several individuals of the same concept). In summary, this pattern allows to represent both simple objects and aggregated objects and their members.

In addition, this pattern can be used to detect the following contradictory situation by means of applying a reasoner: 'to instantiate the relationship "hasAggregatedMember" for an Object that belongs to "SimpleObject"'. This situation represents a consistency error and it is detected when a resoner is applied due to the following modelling decisions included in the pattern: (a) "AggregatedObject" class represents the "hasAggregatedMember" domain and (b) "AggregatedObject" is disjoint with "SimpleObject".

3. PATTERN USAGE EXAMPLE

This pattern has been applied to different domains such as service providers and context sources during the mIO! ontology network[1] development.

As an example, we show in Figure 2 the application of the SimpleOrAggegrated pattern to represent that a service provider can be classified as simple or aggregated. Each service provider can be also classified with respect to the type of service it provides (e.g. cultural, entertainment, food, health, etc.).

4. Related work

The origin of this pattern is the modelling of service providers and context sources into the mIO! ontology network [2] within the Spanish project mIO![2]. The pattern has been also applied to computing and storage resources modelling in the Metascheduler ontology[3] in the context of the Spanish project *España Virtual*[4].

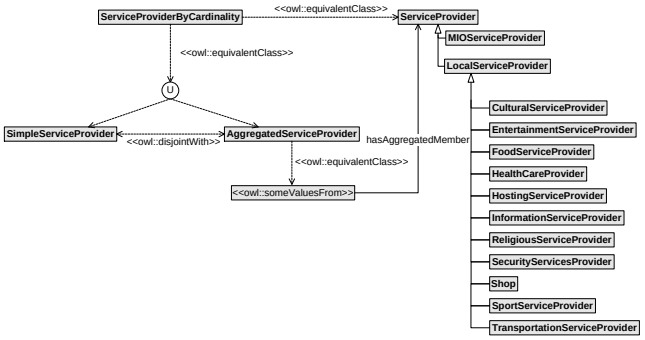

Figure 2. SimpleOrAggregated pattern applied to service providers.

5. Summary and Outlook

The *SimpleOrAggregated pattern* provides a mechanism to classify objects as simple or aggregated objects depending on whether they are an aggregation of some objects. This classification is compatible with another possible classification of objects.

6. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish project *mIO!* (CENIT-2008-1019).

7. REFERENCES

[1] Presutti, V., Gangemi, A., David S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M. *NeOn D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies.* NeOn project. http://www.neon-project.org. 2008.

[2] Poveda, M., Suárez-Figueroa, M.C., García-Castro, R., Gómez-Pérez, A. *A Context Ontology for Mobile Environments*. Proceedings of CIAO 2010. Lisbon, Portugal. 11 October 2010.

[3] Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.. *NeOn D5.1.1: NeOn Modelling Components*. NeOn project. http://www.neon-project.org. March 2007.

[1] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/82-mio-ontologies

[2] http://www.cenitmio.es/

[3] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/85-metascheduler-ontologies

[4] http://www.españavirtual.org/