Ontology Design Patterns: Improving Findability and Composition

Karl Hammar(✉)

Jönköping University, P.O. Box 1026, 551 11 Jönköping, Sweden
karl.hammar@jth.hj.se

Abstract. Ontology Design Patterns (ODPs) are intended to guide non-experts in performing ontology engineering tasks successfully. While being the topic of significant research efforts, the uptake of these ideas outside the academic community is limited. This paper summarises issues preventing broader adoption of Ontology Design Patterns among practitioners, with an emphasis on finding and composing such patterns, and presents early results of work aiming to overcome these issues.

Keywords: Ontology Design Pattern · eXtreme Design · Tools

1 Introduction

Ontology Design Patterns (ODPs) were introduced by Gangemi [8] and Blomqvist and Sandkuhl [4] in 2005 (extending upon ideas by the W3C Semantic Web Best Practices and Deployment Working Group[1]), as a means of facilitating practical ontology development. These patterns are intended to help guide ontology engineering work, by packaging best practice into small reusable blocks of ontology functionality, to be adapted and specialised by users in individual ontology development use cases.

This idea has gained some traction within the academic community, as evidenced by the Workshop on Ontology Patterns series of workshops held on conjunction with the International Semantic Web Conference. However, the adoption of ODPs among practitioners is still quite limited. If such patterns are to be accepted as useful artefacts also in practice, it is essential that they [10]:

- model concepts and phenomena that are relevant to practitioners' needs
- are constructed and documented in a manner which makes them accessible and easy to use by said practitioners in real-world use cases
- are accompanied by appropriate methods and tools that support their use by the intended practitioners.

While the first requirement above can be said to be fulfilled by the ODPs published online (the majority of which result from projects and research involving

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

V. Presutti et al. (Eds.): ESWC Satellite Events 2014, LNCS 8798, pp. 3–13, 2014.
DOI: 10.1007/978-3-319-11955-7_1

both researchers and practitioners), the latter two requirements have largely been overlooked by the academic community. Many patterns are poorly documented, and at the time of writing, none have been sufficiently vetted to graduate from *submitted* to *published* status in the prime pattern repository online[2]. Toolset support is limited to some of the tasks required when employing patterns, while other tasks are entirely unsupported. Furthermore, the most mature pattern usage support tools are implemented as a plugin for an ontology engineering environment which is no longer actively maintained[3].

In the following paper, these ODP adoption challenges are discussed in more detail, and the author's ongoing work on addressing them is reported. The paper focuses exclusively on Content ODPs as defined in the NeOn Project[4], as this is most common type of Ontology Design Patterns with some 100+ patterns published. The paper is structured as follows: Sect. 2 introduces relevant related published research on ODPs, Sect. 3 focuses on the tasks that need be performed when finding, adapting, and applying patterns, Sect. 4 details the challenges preventing the adoption of ODPs by practitioner ontologists, Sect. 5 proposes solutions to these challenges, Sect. 6 presents the initial results of applying some of those solutions, and Sect. 7 concludes and summarises the paper.

2 Related Work

Ontology Design Patterns were introduced as potential solutions to these types of issues at around the same time independently by Gangemi [8] and Blomqvist and Sandkuhl [4]. The former define such patterns by way of a number of characteristics that they display, including examples such as *"[an ODP] is a template to represent, and possibly solve, a modelling problem"* [8, p. 267] and *"[an ODP] can/should be used to describe a 'best practice' of modelling"* [8, p. 268]. The latter describes ODPs as generic descriptions of recurring constructs in ontologies, which can be used to construct components or modules of an ontology. Both approaches emphasise that patterns, in order to be easily reusable, need to include not only textual descriptions of the modelling issue or best practice, but also some formal ontology language encoding of the proposed solution. The documentation portion of the pattern should be structured and contain those fields or slots that are required for finding and using the pattern.

Since their introduction, ODPs have been the subject of some research and work, see for instance the deliverables of the EU FP6 NeOn Project[5] [5,15] and the work presented at instances of the Workshop on Ontology Patterns[6] at the International Semantic Web Conference. There are to the author's best knowledge no studies indicating ontology engineering performance improvements in terms of time required when using patterns, but results so far indicate that

[2] http://ontologydesignpatterns.org/
[3] XD Tools for NeOn Toolkit, http://neon-toolkit.org/wiki/XDTools.
[4] http://ontologydesignpatterns.org/wiki/Category:ContentOP
[5] http://www.neon-project.org/
[6] http://ontologydesignpatterns.org/wiki/WOP:Main

their usage can help lower the number of modelling errors and inconsistencies in ontologies, and that they are perceived as useful and helpful by non-expert users [3,6].

The use and understanding of ODPs have been heavily influenced by the work taking place in the NeOn Project[7], the results of which include a pattern typology [15], and the eXtreme Design collaborative ontology development methods, based on pattern use [5]. eXtreme Design (XD) is defined as *"a family of methods and associated tools, based on the application, exploitation, and definition of Ontology Design Patterns (ODPs) for solving ontology development issues"* [14, p. 83]. The method is influenced by the eXtreme Programming (XP) [2] agile software development method, and like it, emphasises incremental development, test driven development, refactoring, and a divide-and-conquer approach to problem-solving [13]. Additionally, the NeOn project funded the development of the XD Tools, a set of plugin tools for the NeOn Toolkit IDE intended to support the XD method of pattern use.

Ontology Design Patterns have also been studied within the CO-ODE project [1,7], the results of which include a repository of patterns[8] and an Ontology Pre-Processing Language (OPPL)[9].

3 Using Ontology Design Patterns

The eXtreme Design method provides recommendations on how one should structure an Ontology Engineering project of non-trivial size, from tasks and processes of larger granularity (project initialisation, requirements elicitation, etc.) all the way down to the level of which specific tasks need be performed when employing a pattern to solve a modelling problem. Those specific pattern usage tasks (which are also applicable in other pattern-using development methods) are:

1. Finding patterns relevant to the particular modelling issue
2. Adapting those general patterns to the modelling use case
3. Integrating the resulting specialisation with the existing ontology (i.e., the one being built).

3.1 Finding ODPs

In XD, the task of finding an appropriate design pattern for a particular problem is viewed as a matching problem where a local use case (the problem for which the ontology engineer needs guidance) is matched to a general use case (the intended functionality of the pattern) encoded in the appropriate pattern's documentation. In order to perform such matching, the general use case needs be expressed in a way that enables matching to take place. In practice, pattern

[7] http://www.neon-project.org/
[8] http://odps.sourceforge.net/odp/html/index.html
[9] http://oppl2.sourceforge.net/

intent is encoded using Competency Questions [9], and matching is performed by hand, by the ontology engineer him/herself. XD Tools supports rudimentary keyword-based search across the ontologydesignpatterns.org portal, which can provide the ontology engineer with an initial list of candidate patterns for a given query.

3.2 Specialising ODPs

Having located a pattern appropriate for reuse in a specific scenario, the ontology engineer needs to adapt and specialise said pattern for the scenario in question. The specific steps vary from case to case, but a general approach that works in the majority of cases is as follows:

1. Specialise leaf classes of the subclass tree
2. Specialise leaf properties of the subproperty tree
3. Define domains and ranges of specialised properties to correspond with the specialised classes.

The XD Tools provide a wizard interface that supports each these steps. They also provide a certain degree of validation of the generated specialisations, by presenting the user with a list of generated axioms (expressed in natural language) for the user to reject or accept.

3.3 Integrating ODP Instantiations

Once a pattern has been adapted for use in a particular scenario, the resulting solution module needs to be integrated with the ontology under development. This integration involves aligning classes and properties in the pattern module with existing classes and properties in the ontology, using subsumption or equivalency mappings. This integration process may also include refactoring of the existing ontology, in the case that requirements dictate that the resulting ontology be highly harmonised. There is at the time of writing no known tool support for ODP instantiation integration, and this process is therefore performed entirely by hand.

4 ODP Adoption Challenges

As indicated above, there is a thriving research community studying patterns and developing new candidate ODPs. Unfortunately the adoption of Ontology Design Patterns in the broader Semantic Web community, and in particular among practitioners, is limited. The author has, based on experiences from several studies involving users on different levels (from graduate students to domain experts from industry) [10–12], identified a number of issues that give rise to confusion and irritation among users attempting to employ ODPs, and which are likely to slow uptake of these technologies. Those issues are detailed in the subsequent sections.

4.1 Issues on Finding ODPs

As explained, there are two methods for finding appropriate design patterns for a particular modelling challenge - users can do matching by hand (by consulting a pattern repository and reading pattern documentations one by one), or users can employ the pattern search engine included in XD Tools to suggest candidate patterns. In the former case, as soon as the list of available patterns grows to a non-trivial number (such as in the ontologydesignpatterns.org community portal), users find the task challenging to perform correctly, particularly if patterns are not structured in a way that is consistent with their expectations [10].

In the latter case, signal-to-noise ratio of pattern search engine results is often discouragingly low. In initial experiments (detailed in Sect. 6) the author found that with a result list displaying 25 candidate patterns, the correct pattern was included in less than a third of the cases. In order to guarantee that the correct pattern was included, the search engine had to return more than half of the patterns in the portal, essentially negating the point of using a search engine. Also, the existing pattern search engine included in XD Tools does not allow for filtering the results based on user criteria, which makes it easy for a user to mistakenly import and apply a pattern which is inconsistent with ontology requirements, e.g., on reasoning performance or other constraints.

4.2 Issues on Composing ODPs

The process of integrating a specialised pattern solution module into the target ontology is not supported by any published tools, and consequently relies entirely on the user's ontology engineering skill. Users performing such tasks are often confused by the many choices open to them, and the potential consequences of these choices, not limited to:

- Which mapping axioms should be used between the existing classes and properties and those of the solution module, e.g., equivalency or subsumption?
- Where those pattern instantiation module mapping axioms should be placed: in the target ontology, in the instantiated pattern module, or in a separate mapping module?
- The interoperability effects of customising patterns: for instance, what are the risks in case pattern classes are declared to be subsumed by existing top level classes in the target ontology?
- How selections from the above composition choices affect existing ontology characteristics such as reasoning performance, etc.

4.3 Issues on Pattern and Tooling Quality

Users often express dissatisfaction with the varying degree of documentation quality [10]. While some patterns are documented in an exemplary fashion, many lack descriptions of intents and purpose, consequences of use, or example use cases. Experienced ontology engineers can see through this by studying

the accompanying OWL module in order to learn the benefits and drawbacks of a certain pattern, but it is uncommon for non-expert users to do this successfully.

It is not uncommon for patterns to include and build upon other patterns, and these dependencies are not necessarily intuitive or well-explained. On several occasions the author has been questioned by practitioner users as to why, in the ontologydesignpatterns.org repository, the pattern concerning time indexed events makes use of the *Event* class that is defined in the (non time-indexed) *Participation* pattern. The consequence of this dependency structure is of course that any user who models time indexed events using patterns automatically also includes non time-indexed participation representations in their resulting model, which very easily gives rise to modelling mistakes.

In more practical terms, the XD Tools were designed to run as a plugin for the NeOn Toolkit ontology IDE. This IDE unfortunately never gained greater adoption. Additionally, XD Tools and its dependencies require a specific older version of NeOn Toolkit. This means that ontology engineers who want to use newer tools and standards are unable to use XD Tools, but rather have to do their pattern-based ontology engineering without adequate tool support.

5 Improvement Ideas

The author's ongoing research aims to improve upon ODP usage methods and tools, in the process solving some of the issues presented above. To this end, a number of solution suggestions have been developed, and are currently in the process of being tested (some with positive results, see Sect. 6). The following sections present these suggestions and the consequences they would have on both patterns and pattern repositories. Implementation of these suggested improvements within an updated version of the XD Tools targeting the Protégé editor is planned to take place in the coming months.

5.1 Improving ODP Findability

In order to improve recall when searching for suitable ODPs, the author suggests making use of two pieces of knowledge regarding patterns that the current XD Tools pattern search engine does not consider: firstly, that the core intent of the patterns in the index is codified as competency questions, which are structurally similar to such queries that an end-user may pose, and secondly, that patterns are general or abstract solutions to a common problem, and consequently, the specific query that a user inputs needs to be transformed into a more general form in order to match the indexed patterns level of abstraction.

The first piece of knowledge can be exploited by using string distance metrics to determine how similar an input query is to the competency questions associated with a pattern solution. Another approach under study is to employ ontology learning methods to generate graphs from both indexed pattern competency questions and input queries, and then measuring the degree of overlap between concepts referenced in these two graphs.

The second piece of knowledge can be exploited by reusing existing language resources that represent hyponymic relations, such as WordNet. By enriching the indexed patterns with synonyms of disambiguated classes and properties in the pattern, and by enriching the user query using hypernym terms of the query, the degree of overlap between a user query (worded to concern a specific modelling issue) against a pattern competency question (worded to concern a more general phenomenon) can be computed.

5.2 Improving ODP Integration

The challenge of integrating an instantiated pattern module into a target ontology is at its core an ontology alignment challenge. Consequently existing ontology alignment and ontology matching methods are likely to be useful in this context. The behaviour of such systems against very small ontologies such as instantiated pattern modules, is however not well known. The advantage that patterns have over general ontologies in this context is the knowledge that patterns are designed with the very purpose of being adapted and integrated into other ontologies, which is not true in the general ontology alignment use case. Therefore, the pattern creator could a priori consider different ways in which that pattern would best be integrated with an ontology, and construct the pattern in such a way as to make this behaviour known to an alignment system.

The author suggests reusing known good practice from the ontology alignment domain, and combining this with such pattern-specific alignment hints embedded in the individual pattern OWL files. For instance, a pattern class could be tagged with an annotation indicating to a compatible alignment system that this class represents a very high level or foundational concept, and that consequently, it should not be aligned as a subclass; or a pattern class or property could be tagged with annotations indicating labels of suitable sub- or superclasses in the integration step.

Additionally, improved user interfaces would aid non-expert users in applying patterns. Such user interfaces should detail in a graphical or otherwise intuitive manner the consequences of selecting a particular integration strategy, in the case that multiple such strategies are available for consideration.

6 Results

6.1 ODP Search

The author has developed a method of indexing and searching over a set of Ontology Design Patterns based on the ideas presented in Sect. 5. The method combines the existing Lucene-backed Semantic Vectors Search method with a comparison of competency questions based on their relative Levenshtein edit distances, and a comparison of the number of query hypernyms that can be found among the pattern concept synonyms. Each method generates a confidence value between 0 and 1, and these confidence values are added together with equal

weight to generate the final confidence value which is used for candidate pattern ordering. While the approach requires further work, early results are promising, as shown in Table 1.

The dataset used in testing was created by reusing the question sets provided by the *Question Answering over Linked Data* (QALD) evaluation campaign. Each question was matched to one or more ODPs suitable for building an ontology supporting the question. This matching was performed by two senior ontology experts independently, and their respective answer sets merged. The two experts reported very similar pattern selections in the cases where only a single pattern candidate existed in the pattern repository compliant with a competency question (e.g., the *Place*[10] or *Information Realization*[11] patterns), but for such competency questions where multiple candidate patterns existed representing different modelling practices (e.g., the *Agent Role*[12] or *Participant Role*[13] patterns), their selections among these candidate patterns diverged. Consequently, the joint testing dataset was constructed via the union of the two experts' pattern selections (representing the possibility of multiple correct modelling choices), rather than their intersection. Recall was defined as the ratio of such expert-provided ODP candidates that the automated system retrieves for a given input question.

Table 1. Recall improvement for ODP search

	XD-SVS	Composite3
R10	6 %	22 %
R15	8 %	31 %
R20	9 %	37 %
R25	14 %	41 %

As shown in the table, the average recall within the first 10, 15, 20 or 25 results is 3–4 times better using the author's composite method (Composite3) than using the existing XD Tools Semantic Vectors Search (XD-SVS). It should be noted that while Composite3 also increases the precision of the results compared to XD-SVS by a similar degree, that resulting precision is still rather poor, at 5–6 %. The potential pattern user will consequently see a lot of spurious results using either of the approaches. This is understood to be a potential usability problem, and an area for further work.

A factor believed to be limiting the success of this method is the fact that resolving ODP concepts and properties to corresponding concepts and properties in natural language resources (in this case WordNet) is an error-prone process.

[10] http://ontologydesignpatterns.org/wiki/Submissions:Place
[11] http://ontologydesignpatterns.org/wiki/Submissions:Information_realization
[12] http://ontologydesignpatterns.org/wiki/Submissions:AgentRole
[13] http://ontologydesignpatterns.org/wiki/Submissions:ParticipantRole

This is largely due to the ambiguity of language and the fact that concepts in ODPs are generally described using only a single label per supported language. If pattern concepts were more thoroughly documented, using for instance more synonymous labels, class sense disambiguation would likely work better, and ODP search consequently work better also. Additionally, WordNet does contain parts of questionable quality (both in terms of coverage and structure), the improvement of which may lead to increased quality of results for dependent methods such as the one presented here.

6.2 ODP Composition

Based on an empirical study of ODP composition as employed in ontology engineering tasks in the IKS Project[14], a number of heuristics for ODP composition have been extracted, and are presently being developed into a Protégé plugin supporting such composition. Most of these heuristics are very simple and make use of basic string matching techniques across labels or concept URIs (e.g., if there is a greatest common substring of more than trivial length, suggest the concept with the longer label as subconcept of the one with the shorter label, etc.). Yet these simple heuristics cover over 60 % of the composition mappings in the source dataset. While end-users would be able to connect such concepts themselves by hand, suggesting them to the user will still save them considerable work, as they do not themselves have to dig through the subsumption hierarchy to locate the classes and properties.

These heuristics are being coupled with a confidence scoring mechanism based on the *scope of control* of an ontology engineering project. This scope defines which namespaces in the project that are allowed to be modified when performing ODP composition. The scope can be set manually by the ontology engineer, but is by default based on the namespaces of open and editable files in the Protégé environment. Redefining the semantics of concepts outside of the scope of control is not recommended. Consequently, composition subsumption axioms in which concepts outside of the scope of control are defined as the subconcepts, are penalised and ranked lower than axioms which do not give rise to such a situation.

From the same study it was observed that all of the mapping axioms used to composite an ODP specialisation module into a resulting ontology were subsumption mappings (i.e., subClassOf or subPropertyOf). Equivalence mappings were not used at all. This may indicate a cognitive understanding of the domain of discourse as being layered, that is to say, that ODP specialisation modules represent a layer of understanding which is more general than the final ontology, but mer specific than the original ODP. For instance, in the ontologies studied, List (an ODP) is specialised as ContentList (an ODP specialisation), which is then composed via subclassing into RadioChannel, Playlist, and NewsStream. If this observation holds also in a larger set of ontologies and ODPs, it may indicate that the border between ODP specialisation and ODP composition is

[14] http://www.iks-project.eu/

not as clear cut as previously thought, and that tooling for specialisation and composition would need be more tightly integrated.

7 Conclusions

This paper has introduced and discussed some concrete challenges regarding the use of Ontology Design Patterns, with an emphasis on tooling-related challenges that prevent non-expert users from performing Ontology Engineering using such patterns. Those challenges primarily concern; (a) the task of finding patterns, (b) decisions to make when integrating pattern based modules with an existing ontology, and, (c) pattern and tooling quality. The author's work aims to overcome these challenges by developing improved methods and accompanying tools for today's Ontology Engineering IDE:s (i.e., Protégé), better supporting each step of ODP application and use.

The author has developed an ODP search method exploiting both the similarity between pattern competency questions and user queries, and the relative abstraction level of general pattern solutions versus concrete user queries, a method shown to increase recall when searching for candidate ODPs significantly. Future work regarding ODP findability includes improving recall and precision further, and to examine which type of criteria users want to be able to filter results based on.

The author has also developed a set of heuristics for ODP composition suggestions, and a confidence scoring method based on the scope of control of an ODP-based ontology engineering project, presently being implemented into a tool for guiding ODP composition. Future work regarding ODP composition includes tackling the more difficult alignments, possibly via enrichment using lexical resources.

References

1. Aranguren, M.E., Antezana, E., Kuiper, M., Stevens, R.: Ontology design patterns for bio-ontologies: a case study on the cell cycle ontology. BMC Bioinf. **9**(Suppl 5), S1 (2008)
2. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Boston (2004)
3. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on pattern-based ontology design. In: Proceedings of the Fifth International Conference on Knowledge Capture, pp. 41–48. ACM (2009)
4. 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)
5. Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V., Villazon-Terrazas, B.: D2.5.2: pattern based ontology design: methodology and software support. Technical report, NeOn Project (2007)
6. Dzbor, M., Suárez-Figueroa, M.C., Blomqvist, E., Lewen, H., Espinoza, M., Gómez-Pérez, A., Palma, R.: D5.6.2 experimentation and evaluation of the NeOn methodology. Technical report, NeOn Project (2007)

7. Egaña, M., Rector, A.L., Stevens, R., Antezana, E.: Applying ontology design patterns in bio-ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 7–16. Springer, Heidelberg (2008)
8. Gangemi, A.: Ontology design patterns for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005)
9. Grüninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Rolstadås, A. (ed.) Benchmarking – Theory and Practice. IFIP AICT, pp. 22–31. Springer, New York (1995)
10. Hammar, K.: Ontology design patterns in use: lessons learnt from an ontology engineering case. In: Proceedings of the 3rd Workshop on Ontology Patterns (2012)
11. Hammar, K.: Towards an Ontology Design Pattern Quality Model. Linköping Studies in Science and Technology, vol. 1606. Linköping University (2013)
12. Hammar, K., Lin, F., Tarasov, V.: Information reuse and interoperability with ontology patterns and linked data. In: Abramowicz, W., Tolksdorf, R., Węcel, K. (eds.) BIS 2010. LNBIP, vol. 57, pp. 168–179. Springer, Heidelberg (2010)
13. Presutti, V., Blomqvist, E., Daga, E., Gangemi, A.: Pattern-based ontology design. In: Suárez-Figueroa, M.C., et al. (eds.) Ontology Engineering in a Networked World, pp. 35–64. Springer, Heidelberg (2012)
14. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme design with content ontology design patterns. In: Proceedings of the Workshop on Ontology Patterns (WOP 2009), Collocated with ISWC 2009, p. 83 (2009)
15. Presutti, V., Gangemi, A., David, S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: D2.5.1: a library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. Technical report, NeOn Project (2007)