Comparing Approaches for Capturing Repetitive Structures in Ontology Design Patterns *

Christian Kindermann, Bijan Parsia, and Uli Sattler

{christian.kindermann,bijan.parsia,uli.sattler}@manchester.ac.uk
University of Manchester, UK

Abstract. Ontology Design Patterns (ODP) are often described as reusable solutions to recurrent modelling problems. Yet, there is no well-established formalism or framework for facilitating the reuse of ODPs in practise. This motivates a comparison of existing approaches and proposals for capturing modelling solutions recommended by ODPs. To gain a first understanding, we identify and characterise repetitive structures, e.g. axioms or logical expressions, as an important aspect of many ODPs. We find that existing frameworks provide features to support simple repetitive structures that commonly occur in modelling solutions of ODPs.

1 Introduction

The idea of Ontology Design Patterns (ODP) has been introduced as a means to facilitate ontology engineering [6,10]. In addition to providing descriptions of best practises, ODPs place an emphasis on notions of reusability in practise. Despite numerous considerations on how ODPs can be reused, there is no widely-used notion of ODP reuse nor is there a well-established formalism or framework for facilitating ontology engineering with ODPs. This motivates a comparison of existing approaches and proposals for capturing modelling solutions recommended by ODPs.

In this paper, we aim to develop a first understanding of the requirements for ODP reuse by focusing on repetitive structures, e.g. axioms or logical expressions, that are part of an ODP's documentation. The contributions are as follows: (i) we identify and characterise repetitive structures occurring in ODPs, (ii) we exmplify how repetitive structures manifest under notions of ODP reuse, (iii) and we compare two existing frameworks with respect to their support for identified repetitive structures.

2 Preliminaries

We assume the reader to be familiar with description logics (DL) [1] and their relation to the OWL Web Ontology Language [2]. Although we will make use of German DL syntax for specifying axioms, we will stick to OWL parlance

otherwise. We will refer to the set of non-logical symbols (class, property, and individual names) of an ontology as its *signature*. Any element of an ontology's signature will be referred to as an *entity*.

3 Repetitive Structures in Ontology Design Patterns

ODPs have been proposed for a wide range of ontology engineering tasks. As a result, a variety of different kinds of ODPs exist [3, 6, 13, 35]. In the scope of this work, we focus on two kinds of ODPs that are most commonly discussed under the names of *Content Ontology Design Pattern* (CODP) and *Logical Ontology Design Patterns* (LODP). CODPs are motivated as *conceptual* modelling solutions that address domain specific modelling problems, whereas LODPs are motivated as *structural* modelling solutions that mitigate expressive limitations in terms of available langauge primitives of knowledge representation fromalisms, e.g. OWL [11]. However, both these two types of ODPs can be characterised by providing "practical building blocks" for ontological engineering [29, 30]. This characterisation is based on the idea of reusing an ODP via a set of predefined axioms that may or may not be modified [7, 29, 36]. In particular, CODPs are supposed to be reused via a concrete set of axioms that features a domain dependant signature. LODPs, on the other hand, are supposed to be reused by instantiating a set of axioms that features variables in lieu of domain dependent signature [10, 30].

In the following, we give examples for the reuse of both CODPs and LODPs (cf. Section 3.1) and how these ways of reusing ODPs involve repetitive structures (cf. Section 3.2). Finally, we characterise these structures and identify publicly available ODPs exhibiting aspects of repetitive structures as part of their design (cf. Section 3.3).

3.1 Examples of ODP Reuse

The descriptions of both CODPs and LODPs often suggest a notion of ODP reuse in terms of some implied metamodel. For example, in case of LODPs there is generally an assumed notion of variables that are to be instantiated. However, neither variables nor a valid ways for instantiating them are explicitly defined or qualified. Such vagueness in the documentation of ODPs gives room for interpretation with respect to an ODP's reuse in practise. We will discuss issues related to this lack of formal rigour in Section 3.2 and Section 5. For the purpose of this section, we present example ODPs that are documented in a well-known, publicly available library of ODPs [30]. An ODP's documentation usually contains a diagram that visualises how different components of the pattern are related to each other. Although these diagrams are commonly given in reference to OWL, in general, there is no one-to-one correspondence to an actual set of OWL axioms. In the following, we give concrete examples for both a LODP and a CODP.

Example 1 (CODP). The CODP `NaryParticipation` [30] provides a modelling solution for representing events in relation to participants, time, and space. It introduces explicit non-logical vocabulary for classes and properties, e.g.,

Event, Situation, isParticipantIn, or participationIncludes. The pattern is supposed to be reused by extending these classes and properties by subclasses and subproperties. The following figure depicts a conceptual diagram for the pattern.

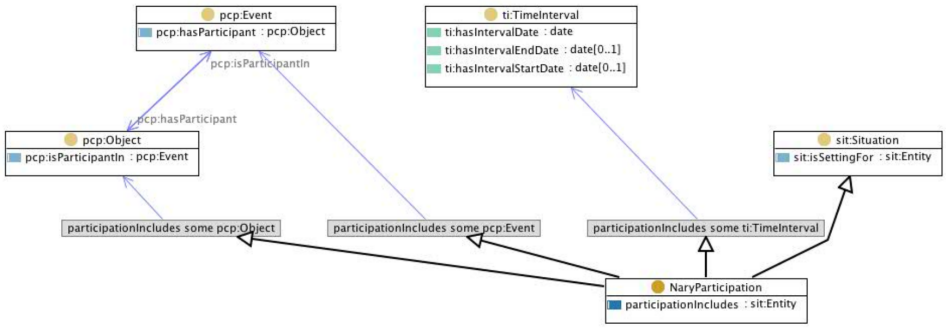

Fig. 1. `NaryParticipation` (taken from "NeOn Catalogue" [30])

For a concrete use case, consider an academic conference, e.g. ISWC, and its participants in terms of authors, reviewers, and organisers. Suppose that authors a_1, a_2, a_3 participated in the 2019 edition of ISWC with a jointly written paper p that has been reviewed by reviewers r_1, r_2, r_3. Then, the following adaptation of the `NaryParticipation` pattern captures this situation.

First, the specified classes of `NaryParticipation` are specialised by creating domain specific subclasses and subproperties (see Figure 2).

$$\begin{aligned}
\mathsf{Conference} &\sqsubseteq \mathsf{Event} & \mathsf{Publication} &\sqsubseteq \exists \mathsf{includesVenue}.\mathsf{Conference}\\
\mathsf{Author} &\sqsubseteq \mathsf{Object} & \mathsf{Publication} &\sqsubseteq \exists \mathsf{includesAuthor}.\mathsf{Author}\\
\mathsf{Organiser} &\sqsubseteq \mathsf{Object} & \mathsf{Publication} &\sqsubseteq \exists \mathsf{includesOrganiser}.\mathsf{Organiser}\\
\mathsf{Reviewer} &\sqsubseteq \mathsf{Object} & \mathsf{Publication} &\sqsubseteq \exists \mathsf{includesReviewer}.\mathsf{Reviewer}\\
\mathsf{Paper} &\sqsubseteq \mathsf{Object} & \mathsf{Publication} &\sqsubseteq \exists \mathsf{includesPaper}.\mathsf{Paper}
\end{aligned}$$

$$\begin{aligned}
\mathsf{Publication} &\sqsubseteq \mathsf{NaryParticipation}\\
\mathsf{Conference} &\sqsubseteq \forall \mathsf{hasParticipant}.(\mathsf{Author} \sqcup \mathsf{Organiser} \sqcup \mathsf{Reviewer})
\end{aligned}$$

$$\begin{aligned}
\mathsf{includesAuthor} &\sqsubseteq \mathsf{participationIncludes}\\
\mathsf{includesPaper} &\sqsubseteq \mathsf{participationIncludes}\\
\mathsf{includesReviewer} &\sqsubseteq \mathsf{participationIncludes}\\
\mathsf{includesOrganiser} &\sqsubseteq \mathsf{participationIncludes}\\
\mathsf{includesTimeInterval} &\sqsubseteq \mathsf{participationIncludes}\\
\mathsf{includesVenue} &\sqsubseteq \mathsf{participationIncludes}
\end{aligned}$$

Fig. 2. TBox for modelling conference publication with `NaryParticipation`

Next, the newly created subclasses are populated as needed, i.e, $\mathsf{Author}(a_i)$, $\mathsf{Reviewer}(\mathsf{r_i})$, etc., and finally all individual parts are interrelated via a shared instance of the Publication class that is subsumed by NaryParticipation (see Figure 3). □

	$\mathsf{Publication}(N_1)$
$\mathsf{Conference}(ISWC)$	$\mathsf{includesConference}(N_1, ISWC)$
$\mathsf{Author}(a_1)$	$\mathsf{includesAuthor}(N_1, a_1)$
$\mathsf{Author}(a_2)$	$\mathsf{includesAuthor}(N_1, a_2)$
$\mathsf{Author}(a_3)$	$\mathsf{includesAuthor}(N_1, a_3)$
$\mathsf{Reviewer}(r_1)$	$\mathsf{includesReviewer}(N_1, r_1)$
$\mathsf{Reviewer}(r_2)$	$\mathsf{includesReviewer}(N_1, r_2)$
$\mathsf{Reviewer}(r_3)$	$\mathsf{includesReviewer}(N_1, r_3)$
$\mathsf{Paper}(p)$	$\mathsf{includesPaper}(N_1, p)$
$\mathsf{TimeInterval}(d_1)$	$\mathsf{includesTimeInterval}(N_1, d_1)$
$\mathsf{hasIntervalStartDate}(d_1, 26.10.2019)$	
$\mathsf{hasIntervalEndDate}(d_1, 30.10.2019)$	

Fig. 3. ABox populating the `NaryParticipation` pattern

Example 2 (LODP). The LODP `NaryRelation` [30] provides a modelling solution for representing n-ary relationships in OWL. Since OWL does not provide a language primitive for n-ary relationships, the pattern describes how n-ary relationships can be captured by a combination of several binary relations and an auxiliary (newly created) class. The auxiliary class is said to *reify* the original n-ary relationship. Figure 4 shows a conceptual diagram for the pattern.

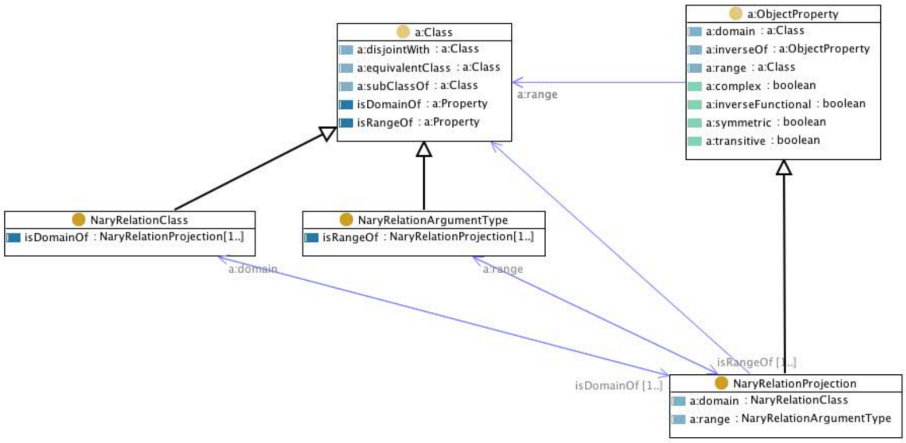

Fig. 4. `NaryRelation` (taken from "NeOn Catalogue" [30])

In this diagram, the NaryRelationClass constitutes the reified class for the n-ary relationship. This class is used to establish a connection between all n components of the original relationship each of which will be represented by a distinct instantiation of the class NaryRelationProjection.

As a concrete use case, consider information about a university level exam. Here, a relation between a student, a course, a grade, and a semester time has to be modelled. This can be achieved by using the `NaryRelation` pattern as follows. First, we reify a class ExamEntry to represent the relationship. Next, we introduce

distinct classes and properties for each component of the original relationship:

$$\begin{aligned} \mathsf{Exam} &\sqsubseteq \exists \mathsf{has.ExamEntry} \\ \mathsf{ExamEntry} &\sqsubseteq \exists \mathsf{ofStudent.Student} \\ \mathsf{ExamEntry} &\sqsubseteq \exists \mathsf{withGrade.Grade} \\ \mathsf{ExamEntry} &\sqsubseteq \exists \mathsf{inSemester.Semester} \\ \mathsf{ExamEntry} &\sqsubseteq \exists \mathsf{inCourse.Course} \end{aligned}$$

Comparing this set of axioms to the axioms for modelling publications in the previous example, it becomes clear that the CODP `NaryParticipation` is based on the LODP `NaryRelation`. Note that the use of existential restrictions in the CODP `NaryParticipation` and in the example for exams is not part of the design proposed by the LODP `NaryRelation`. As Figure 4 indicates, only the domains and ranges of object properties are required to be specified. Our use of existential restrictions is merely for ease of presentation and not a substitute for domain and range axioms. However, there exist variants of the `NaryRelation` pattern that define the use of existential restrictions as mandatory. See for example the following diagram published in the Manchester "ontology design patterns public catalogue".[1] □

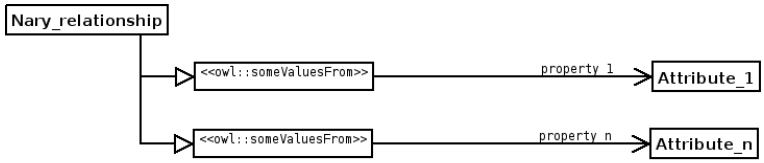

Fig. 5. Variation of `NaryRelation` (taken from "Manchester Catalogue"[2])

3.2 Occurrence of Repetitive Structures in ODPs

The examples in the previous section demonstrate ways of reusing both CODPs and LODPs. For a given CODPs, its classes and properties are extended by subclasses and subproperties, whereas for a given LODP, variable entities are instantiated (e.g. by atomic entities, or compound expressions). For both notions of reuse, extending entities by subsumption on the one hand, and instantiating variables on the other hand, the respective operations may be repeatedly performed to give rise to structurally similar axioms.

For example, the different participants of a conference publication are all modelled in the same manner in Example 1 (CODP). Namely,

$$\begin{aligned} C &\sqsubseteq \mathsf{Object} \\ P &\sqsubseteq \mathsf{participationIncludes} \\ \mathsf{Publication} &\sqsubseteq \exists P.C \end{aligned}$$

[1] http://odps.sourceforge.net/odp/html/index.html
[2] http://odps.sourceforge.net/odp/html/Nary_Relationship.html

where we abstracted over names of domain specific participation objects, e.g. Author or Reviewer, by introducing the variables C (a class variable) and P (a property variable). Similarly, in Example 2 (LODP), all components of an n-ary relationship are modelled by instantiating structurally identical variable components, e.g. existential restrictions or domain and range axioms, repeatedly.

Repetitive structures in ODP driven modelling are not necessarily incidental to ways of reusing patterns in specific use cases. In fact, many ODPs describe repetitive structures as part of their design explicitly. Most CODPs published in [30], that reuse the LODP `NaryRelation`, indicate components of indefinite artiy in their conceptual diagrams by some form of cardinality notation. As an example of this, see the conceptual diagram for the CODP `Situation` shown in Figure 6. Another way of indicating repeated structures is by enumeration. This is frequently done for ODPs published in the Manchester "ontology design patterns public catalogue". An example for this is shown in Figure 5 (see previous section).

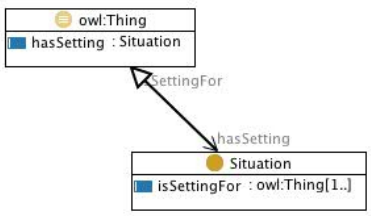

Fig. 6. Situation (taken from "Web Portal Catalogue"[3])

We identify ODPs exhibiting repetitive structures in their design according to the following five criteria:

(a) presence of an indefinite cardinality constraint,
(b) presence of an indefinite enumeration,
(c) presence of a cyclic design component,
(d) presence of an example suggesting indefinite enumeration,
(e) presence of an example suggesting a cyclic component.

Using these criteria, we determine the prevalence of ODPs involving repetitive structures in the following publicly available catalogues:

1. D2.5.1: A Library of Ontology Design Patterns (NeOn Catalogue) [30],
2. ODP Semantic Web Portal (Web Portal Catalogue),[4]
3. Manchester Ontology Design Pattern Public Catalogue (Manchester Catalogue),[5]
4. Modular Ontology Design Library (MODL Catalogue).[6]

Table 1 reports the number of ODPs identified by criteria (a)–(e) above without double-counting ODPs by (d) or (e) if already identified by (a)–(c).

[3] http://ontologydesignpatterns.org/wiki/Submissions:Situation
[4] http://ontologydesignpatterns.org
[5] http://odps.sourceforge.net/odp/html/index.html
[6] https://dase.cs.wright.edu/content/modl-modular-ontology-design-library

Table 1. ODPs exhibiting repetitive structures

Catalogue	Pattern Type	Identified ODPs / Available ODPs	(a)	(b)	(c)	(d)	(e)
NeOn	CODP	23 / 34	5	0	0	16	2
NeOn	LODP	3 / 4	2	0	0	0	1
Web Portal	CODP	31 / 155	14	5	12	0	0
Web Portal	LODP	6 / 18	0	3	2	1	0
Manchester	-	10 / 17	0	9	1	0	0
MODL	-	9 / 18	0	1	8	0	0

3.3 Characterisation of Repetitive Structures

The prevalence of repetitive structures in modelling solutions proposed by ODPs raises the question whether these structures are based on common design principles that suggest similar ways of reuse in practise. Consider the axioms used in Example 1 based on the `NaryParticipation` pattern. Here, n different types of participants $p_1, \ldots, p_n$ at a conference are modelled by using a structure with indefinite arity of the following form:

$$\begin{aligned} \mathsf{Conference} &\sqsubseteq \forall \mathsf{hasParticipant}.\left(\bigsqcup_{1 \leq i \leq n} p_i\right) \\ p_1 &\sqsubseteq \mathsf{Object} \\ &\vdots \\ p_n &\sqsubseteq \mathsf{Object} \end{aligned}$$

In this structure, the variable number of participants $p_1, \ldots, p_n$ are used both within a single logical expression (namely a union of class names), as well as a set of axioms (namely a set of subsumptions $p_i \sqsubseteq \mathsf{Object}$). This observation motivates a threefold distinction of repetitive structures:

1. structures involving some repetition of axioms,
2. structures involving some repetition within logical expressions,
3. structures involving a combination of repetitive structures.

Despite its simplicity, this threefold distinction allows us to gain a first understanding about the nature of repetitive structures that are, by design, part of modelling solutions proposed by ODPs. In Table 2, we list how many ODPs involving repetitive structures (c.f. Section 3.2 categories (a)–(e) and Table 1) involve repetitive axioms, repetitive logical expressions, or some combination of both. It appears that most repetitive structures in ODPs are due to repeated axioms rather than logical expressions of indefinite arity.

Note that these numbers only report on ODPs that suggest repetitive structures explicitly in the documentation of their proposed modelling solution (see Figure 5 for an example). The table does not report on ODPs that give rise to repetitive structures solely by virtue of specific notions of ODPs reuse.

Table 2. Characterisation of ODPs

Catalogue	1. Repetitive Axioms	2. Repetitive Expressions	3. Combination
NeOn	26	0	0
Web Portal	36	0	1
Manchester	5	1	4
MODL	9	0	0

4 Capturing Repetitive Structures in ODP-Based Modelling

Infrastructure for supporting ontology engineering with ODPs is scarce and often not well-maintained [5]. Moreover, it is not well understood what kind of tool support is needed in practise for facilitating ODP-based modelling [16]. However, the prevalence of repetitive structures in ODP's proposed modelling solutions (cf. Table 1 in Section 3.2) raises the question how such structures can be captured. Despite the lack of tool support for ODP reuse, there already exist frameworks for capturing repetitive structures in ontological modelling. In the following, we compare implemented features of two such frameworks with respect to their capability to capture repetitive structures occurring in ODPs.

There currently exist two publicly available frameworks for handling repetitive ontology design tasks, namely the Ontology Pre-Processing Language[7] (OPPL) [8] and Reasonable Ontology Templates[8] (OTTR) [9,33]. Both of these frameworks claim to provide general support for ontological modelling patterns in OWL and suitable support for ODPs in particular. They provide means for directly instantiating variables occurring in a fixed set of axioms. Hence, it is, in principle, possible to reuse LODPs by instantiation. Also, CODPs reuse by the introduction of subclasses and subproperties for some the pattern's entities can be achieved by introducing axioms with variables for said subclasses and subproperties which then need to be instantiated. Thus, there is basic support for some notions of ODP reuse.

In addition to basic support of ODP reuse via simple variable instantiation of a fixed set of axioms, both frameworks provide means for generating a set of axioms based on repeated instantiation of a given set of axioms. For example, given the axiom $?X \sqsubseteq \exists \mathsf{hasAttribute}.?Y$ where $?X$, and $?Y$ are variables, both OPPL and OTTR allow for the generation of the set of axioms, that corresponds to the cross product of variables instantiations of the respective input sets. So, continuing the example with V_1 and V_2 as input sets of class names for $?X$ and $?Y$, then the resulting set of axioms is

$$\{?X \sqsubseteq \exists \mathsf{hasAttribute}.?Y \mid ?X \in V_1, ?Y \in V_2\}.$$

Replicating syntactically similar axioms seems to be a needed feature for the reuse of some LODPs as well as CODPs as most of the repetitive structures

[7] http://oppl2.sourceforge.net/index.html
[8] https://ottr.xyz/

explicitly occurring in ODPs appear to exhibit such repeated axioms (cf. Table 2 in Section 3.3).

Example 3 (Instantiation by Cross Product). Consider the `NaryParticipation` pattern from Example 1 in Section 3.1. Here, all participation objects of an n-ary participation are specialisations of (i.e. are subsumed by) the Object class. Without introducing separate object properties for all objects, the (partial) reuse of the `NaryParticipation` pattern could be formalised by the following set of axioms:

$$?X \sqsubseteq \mathsf{Object}$$
$$?Y \sqsubseteq \mathsf{NaryParticipation}$$
$$?Y \sqsubseteq \exists \mathsf{participationIncludes}.?X.$$

Given $V_1 = \{\mathsf{Author}, \mathsf{Organiser}, \mathsf{Reviewer}, \mathsf{Paper}\}$ and $V_2 = \{\mathsf{Publication}\}$ as sets of class names, the cross product of instantiations $?X$ over V_1 and $?Y$ over V_2 would allow for the generation of repetitive modelling of participation objects. □

In addition to repeated axioms, some ODPs exhibit logical expressions, e.g. class unions, with an indefinite number of components (cf. Table 2 for Catalogue 3). Such variable logical expressions can be captured in both OTTR and OPPL to some degree.

Example 4 (Logical Expressions with indefinite arity). As before, consider the `NaryParticipation` pattern from Example 1 in Section 3.1. Here, the axiom

$$\mathsf{Conference} \sqsubseteq \forall \mathsf{hasParticipant}.(\mathsf{Author} \sqcup \mathsf{Organiser} \sqcup \mathsf{Reviewer})$$

contains a class union of three classes. However, a pattern's design might generalise over the number of different participant types via an axiom of the form

$$\mathsf{Conference} \sqsubseteq \forall \mathsf{hasParticipant}. \left(\bigsqcup_{1 \leq i \leq n} p_i \right),$$

where we introduced n variables for the classes participating in the union that is the filler of the universal quantification. In both OTTR and OPPL, this more general rendering can be expressed if p_i are assumed to be class names. □

While both OPPL and OTTR provide some support for capturing logical expressions of indefinite arity, there are limitations in terms of nesting and composing expressions of indefinite arity as the following example shows.

Example 5 (Compound Logical Expressions with indefinite arity). Both OTTR and OPPL provide support for formulating logical expressions such as an intersection of classes $A_1 \sqcap \ldots \sqcap A_n$, where the number n of classes is not fixed. Furthermore, it is possible to create compound logical expressions of arbitrary arity, as long as

they are not nested. For example, it is possible to formulate an expression that is the union of say three intersections of indefinite arity, i.e.,

$$\underbrace{(A_1 \sqcap \ldots \sqcap A_n)}_{\text{first intersection}} \sqcup \underbrace{(B_1 \sqcap \ldots \sqcap B_m)}_{\text{second intersection}} \sqcup \underbrace{(C_1 \sqcap \ldots \sqcap C_k)}_{\text{third intersection}}.$$

However, it is not possible to generalise this case from a union of three indefinite intersections to an indefinite union of indefinite intersections.[9] □

Lastly, there exist ODPs that propose modelling solutions involving combinations of distinct repetitive structures (cf. Table 2 for the Manchester Catalogue). Such combinations may be structurally interrelated or not. A repetitive structure that involves two structurally independent components may be supported by both OPPL or OTTR if each of the components is supported separately.

Example 6 (Combination of Independent Repetitive Structures). As before, consider the `NaryParticipation` pattern from Example 1 in Section 3.1. Here, the structure

$$\begin{aligned} \mathsf{Conference} \sqsubseteq \forall \mathsf{hasParticipant}. \left(\bigsqcup_{1 \leq i \leq n} p_i \right) \\ p_1 \sqsubseteq \mathsf{Object} \\ \vdots \\ p_n \sqsubseteq \mathsf{Object} \end{aligned}$$

with $p_1 = \mathsf{Author}$, $p_2 = \mathsf{Organiser}$, $p_3 = \mathsf{Reviewer}$ occurs. Since both OPPL and OTTR provide means for generating the repeated axioms $p_i \sqsubseteq \mathsf{Object}$ as well as the axiom containing a union over an indefinite number of class names, the combination of both structures can be captured as well. □

However, not all complex repetitive structures are combinations of structurally independent components as the following example shows.

Example 7 (Interrelated Repetitive Structures). As before, consider the `NaryParticipation` pattern from Example 1 in Section 3.1. Here, the `NaryParticipation` pattern is (partially) reused by extension according to the following general structure:

$$\begin{array}{lll} p_1 \sqsubseteq \mathsf{Object} & r_1 \sqsubseteq \mathsf{participationIncludes} & \mathsf{Publication} \sqsubseteq \exists r_1.p_1 \\ \vdots & \vdots & \vdots \\ p_n \sqsubseteq \mathsf{Object} & r_n \sqsubseteq \mathsf{participationIncludes} & \mathsf{Publication} \sqsubseteq \exists r_n.p_n \end{array}$$

[9] OPPL only allows for a fixed number of 'input variables' that range over entities of an ontology. Hence, only a fixed number of indefinite intersections can be captured. For OTTR, a similar argument can be made.

For each participation object, e.g. Author, a separate object property is introduced, e.g. includesAuthor that is used to establish the relationship to the NaryParticipationObject, here Publication. Therefore, each participation object p_i is related to exactly one object property r_i. This dependency, where each p_i has to be coupled with a corresponding r_i to generate the axioms $\mathsf{Publication} \sqsubseteq \exists r_i.p_i$, constitutes a structure that can be captured in OTTR via a list expansion mode called "zip". However, OPPL does not appear to provide support for such interrelations of repetitive structures. □

5 Discussion

5.1 Choice of Examples

We have guided the comparison of OTTR and OPPL by characteristics of repetitive structures occurring in ODPs and potential ways of their reuse in practise. The choice of our examples was based on two frequently discussed operations for ODP reuse. First, the instantiation of variables in a given set of axioms [12, 17, 30]. And second, extending and replicating a given modelling solution by subclasses and subproperties [5, 29].

However, it needs to be pointed out that there are other ways of ODP reuse and that the very notion of ODP reuse is still an open research question [15, 16]. There is no no standard or widely-used mechanism for reusing an ODP in practise. Hence, the work in this paper is an endeavour to identify features of ODP that may shed light on requirements for mechanisms to facilitate ODP reuse.

5.2 Complex Repetitive Structures

The repetitive structures occurring in most ODPs are of rather simple nature. Axioms are repeated in a very systematic manner and logical expressions with an indefinite number of components are mostly restricted to operators over a set of entity names, e.g. a union of an arbitrary number of class names. However, there are also examples of more complex repetitive structures that exhibit interdependencies between its constituent components.

While a complete coverage of characteristics for more complex patterns is beyond the scope of this work, we give a concrete example `Entity-Property-Quality` pattern (see Figure 7). Reusing this pattern may give rise to three interrelated repetitive structures. First, an entity category is defined in terms of a number of entities $entity_1, \ldots, entity_n$. Second, each entity may have multiple qualities[10] $quality_1, \ldots, quality_m$. And third, each of these qualities is modelled by a union of (pairwise disjoint) quality values $value^q_1, \ldots, value^q_k$ ($value^q_i$ denotes the i^{th} quality value for quality $quality_q$). So overall, an entity $entity_i$ may have a number of quality values $value^p_j, \ldots, value^p_k, \ldots value^m_i, \ldots, value^m_o$ over a number of different qualities $quality_p, \ldots, quality_m$. The interrelationship of these three structures may or may not result in a systematically repetitive ontological model.

[10] Although the diagram for the pattern does not show an enumeration for different qualities, the documentation states the enumeration for qualities verbally: "To model qualities of independent entities (e.g. position, colour, ...)."

It is currently unknown whether (and how) such complex repetitive structures are used in practise and whether they could be captured in OTTR or OPPL.

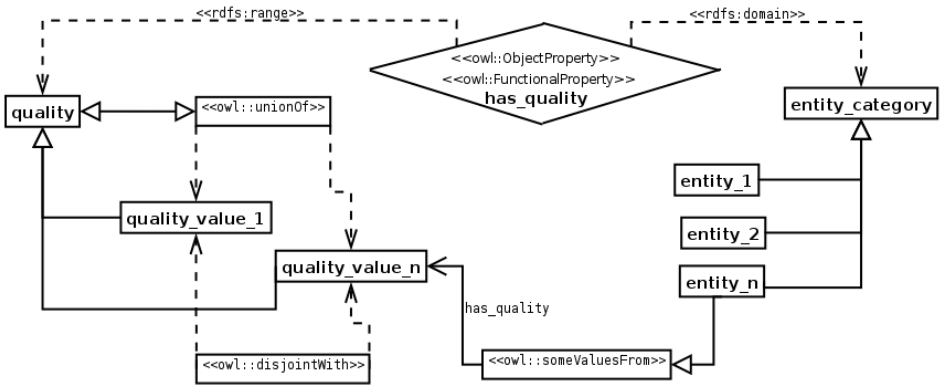

Fig. 7. Entity-Property-Quality (taken from "Manchester Catalogue"[11])

5.3 Related Work

Research on ODPs faces a number of difficult open questions and challenges [16]. Little is known what makes for a good ODPs, how they are to be documented, or how they are best reused in practise.

Existing user studies reveal inconsistent perceptions of the benefits supposedly provided by ODPs. While some studies report that participants perceive ODPs generally as useful [4], others report accounts of scepticism towards their usefulness [14, 17], and still others state demonstrable limitations in practise [23, 32].

These inconclusive results motivate what is often referred to a "bottom-up" approach that tries to gather information for ODP research from existing ontologies. However, empirical studies on the prevalence of publicly accessible ODP repositories indicate limited evidence for ODP reuse in practise [22, 27]. Another bottom-up approach is to motivate new ODPs on the basis of frequent axiom patterns and syntactic or semantic regularities in ontologies [24–26].

Besides user studies, empirical detection and discovery studies for ODPs, there is little research on formal requirements of mechanisms to effectively reuse ODPs in practise. One step in this direction is done by [18], where a mismatch in terms of OWL language profiles between biomedical ontologies and ODPs is identified as a tangible hindrance for many notions of ODP reuse. Another step is done by [20], in which *safe* ways, defined in terms of conservative extensions, of reusing ODP are qualified. Otherwise, there have only been a number of untested proposals of formalisms and frameworks for ODP reuse [6, 7, 10, 19, 21, 28, 29, 31, 34, 36, 37].

6 Conclusion and Future Work

Despite a large number of proposals of different notions for pattern reuse, little is known what kind of features a mechanism for facilitating ODP reuse needs to

[11] http://odps.sourceforge.net/odp/html/Entity_Property_Quality.html

provide. The work in this paper on tool support for capturing repetitive structures contained in modelling solutions of ODPs is a tentative step in this direction. Understanding and qualifying important aspects of ODPs may inform the design of suitable mechanisms for ODP reuse.

In this paper, we have considered repetitive structures of axioms and logical expressions in the context of ODP reuse. It seems that many ODPs exhibit such structures already in their proposed modelling solution or suggest the generation of such structures due to repeated operations of reuse. We found that two existing frameworks seem to provide suitable support for most ODPs with respect to identified repetitive structures. There are examples of complex interrelated structures (cf. Section 5.2) that cannot be appropriately captured by either OPPL nor OTTR but is unclear whether such structures have practical relevance.

Overall, it seems that, in principle, providing sufficient technical support for ODP reuse in practise is possible. The identification of important features occurring in many modelling solutions of ODPs can inform notions of ODP reuse and how they might be supported by tools. While the scope of this work is limited to just two frameworks and their support of repetitive structures in ODPs, there are many more directions for future work. A direct extension of this work might consider aspects of ODPs other than repetitive structures. A more comprehensive comparison between OPPL and OTTR (or other frameworks) to determine how they differ in terms of expressive capabilities would also provide useful information for knowledge engineers in search for a tool of their needs.

Besides such technical aspects of how to provide tool support for ODP reuse, one also has to consider practical aspects in terms of usability. It has already been speculated that OPPL is not widely adopted as a tool for ODP reuse because it may not be easy to use [17]. Therefore, instead of working towards a "feature-complete" mechanism for ODP reuse, one also needs to take into account the practical needs of potential users. Identifying practically relevant ODPs and facilitating their reuse in an intuitive and easy to use manner still appears to be a tough challenge.

References

1. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Tech. rep., W3C, http://www.w3.org/TR/owl-ref/ (February 2004)
3. Blomqvist, E.: Ontology Patterns: Typology and Experiences from Design Pattern Development. In: The Swedish AI Society Workshop May 20-21; 2010; Uppsala University. pp. 55–64. No. 048, Linköping University Electronic Press (2010)
4. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on pattern-based ontology design. In: K-CAP. pp. 41–48. ACM (2009)
5. Blomqvist, E., Hammar, K., Presutti, V.: Engineering ontologies with patterns - the extreme design methodology. In: Ontology Engineering with Ontology Design Patterns, Studies on the Semantic Web, vol. 25, pp. 23–50. IOS Press (2016)

6. Blomqvist, E., Sandkuhl, K.: Patterns in Ontology Engineering–Classification of Ontology Patterns. In: ICEIS 2005: proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005 (2005)
7. Clark, P.: Knowledge Patterns. In: EKAW. Lecture Notes in Computer Science, vol. 5268, pp. 1–3. Springer (2008)
8. Egaña, M., Stevens, R., Antezana, E.: Transforming the axiomisation of ontologies: The ontology pre-processor language. In: OWLED (Spring). CEUR Workshop Proceedings, vol. 496. CEUR-WS.org (2008)
9. Forssell, H., Lupp, D.P., Skjæveland, M.G., Thorstensen, E.: Reasonable macros for ontology construction and maintenance. In: Description Logics. CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017)
10. Gangemi, A.: Ontology Pesign Patterns for Semantic Web Content. In: International Semantic Web Conference. pp. 262–276. Springer (2005)
11. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on ontologies, pp. 221–243. Springer (2009)
12. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on Ontologies, pp. 221–243. International Handbooks on Information Systems, Springer (2009)
13. Guizzardi, G.: Theoretical Foundations and Engineering Tools for Building Ontologies as Reference Conceptual Models. Semantic Web **1**(1-2), 3–10 (2010)
14. Hammar, K.: Ontology design patterns in use: lessons learnt from an ontology engineering case. In: Proceedings of the 3rd International Conference on Ontology Patterns-Volume 929. pp. 13–24. CEUR-WS. org (2012)
15. Hammar, K.: Ontology design pattern property specialisation strategies. In: EKAW. Lecture Notes in Computer Science, vol. 8876, pp. 165–180. Springer (2014)
16. Hammar, K., Blomqvist, E., Carral, D., van Erp, M., Fokkens, A., Gangemi, A., van Hage, W.R., Hitzler, P., Janowicz, K., Karima, N., Krisnadhi, A., Narock, T., Segers, R., Solanki, M., Svátek, V.: Collected Research Questions Concerning Ontology Design Patterns. In: Ontology Engineering with Ontology Design Patterns, Studies on the Semantic Web, vol. 25, pp. 189–198. IOS Press (2016)
17. Hammar, K., Presutti, V.: Template-Based Content ODP Instantiation. In: The 7th Workshop on Ontology and Semantic Web Patterns. IOS Press (2017)
18. Horridge, M., Aranguren, M.E., Mortensen, J., Musen, M.A., Noy, N.F.: Ontology Design Pattern Language Expressivity Requirements. In: WOP. CEUR Workshop Proceedings, vol. 929. CEUR-WS.org (2012)
19. Hou, C.J., Noy, N.F., Musen, M.A.: A Template-Based Approach Toward Acquisition of Logical Sentences. In: Intelligent Information Processing. IFIP Conference Proceedings, vol. 221, pp. 77–89. Kluwer (2002)
20. Iannone, L., Palmisano, I., Rector, A.L., Stevens, R.: Assessing the safety of knowledge patterns in OWL ontologies. In: ESWC (1). Lecture Notes in Computer Science, vol. 6088, pp. 137–151. Springer (2010)
21. Kindermann, C., Lupp, D.P., Sattler, U., Thorstensen, E.: Generating ontologies from templates: A rule-based approach for capturing regularity. In: Description Logics. CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018)
22. Kindermann, C., Parsia, B., Sattler, U.: Detecting influences of ontology design patterns in biomedical ontologies. In: Description Logics. CEUR Workshop Proceedings, vol. 2373. CEUR-WS.org (2019)
23. Lantow, B., Sandkuhl, K., Tarasov, V.: Ontology reuse. In: KEOD. pp. 163–170. SciTePress (2015)
24. Lawrynowicz, A., Potoniec, J., Robaczyk, M., Tudorache, T.: Discovery of emerging design patterns in ontologies using tree mining. Semantic Web **9**(4), 517–544 (2018)

25. Mikroyannidi, E., Manaf, N.A.A., Iannone, L., Stevens, R.: Analysing syntactic regularities in ontologies. In: OWLED. CEUR Workshop Proceedings, vol. 849. CEUR-WS.org (2012)
26. Mikroyannidi, E., Quesada-Martínez, M., Tsarkov, D., Fernández-Breis, J.T., Stevens, R., Palmisano, I.: A quality assurance workflow for ontologies based on semantic regularities. In: EKAW. Lecture Notes in Computer Science, vol. 8876, pp. 288–303. Springer (2014)
27. Mortensen, J., Horridge, M., Musen, M.A., Noy, N.F.: Modest Use of Ontology Design Patterns in a Repository of Biomedical Ontologies. In: WOP. CEUR Workshop Proceedings, vol. 929. CEUR-WS.org (2012)
28. Noppens, O., Liebig, T.: Ontology patterns and beyond - towards a universal pattern language. In: WOP. CEUR Workshop Proceedings, vol. 516. CEUR-WS.org (2009)
29. Presutti, V., Gangemi, A.: Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In: International Conference on Conceptual Modeling. pp. 128–141. Springer (2008)
30. Presutti, V., Gangemi, A., David, S., de Cea, G.A., 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. (2008), (Available at: http://www.neon-project.org/)
31. Reich, J.R.: Onthological Design Patterns for the Integration of Molecular Biological Information. In: German Conference on Bioinformatics. pp. 156–166 (1999)
32. Rodriguez-Castro, B., Ge, M., Hepp, M.: Alignment of ontology design patterns: Class as property value, value partition and normalisation. In: OTM Conferences (2). Lecture Notes in Computer Science, vol. 7566, pp. 682–699. Springer (2012)
33. Skjæveland, M.G., Lupp, D.P., Karlsen, L.H., Forssell, H.: Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In: International Semantic Web Conference (1). Lecture Notes in Computer Science, vol. 11136, pp. 477–494. Springer (2018)
34. Staab, S., Erdmann, M., Maedche, A.: Engineering Ontologies using Semantic Patterns. In: OIS@IJCAI. CEUR Workshop Proceedings, vol. 47. CEUR-WS.org (2001)
35. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.: D 5.1.1 NeOn Modelling Components (March 2007), (Available at: http://www.neon-project.org)
36. Svátek, V.: Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In: In: 7 th Conf. on Business Information Systems (BIS-04) (2004)
37. Vrandecic, D.: Explicit knowledge engineering patterns with macros. In: Proceedings of the Ontology Patterns for the Semantic Web Workshop a the ISWC 2005. Galway, Ireland (2005)