Alignment of Ontology Design Patterns: Class As Property Value, Value Partition and Normalisation

Bene Rodriguez-Castro, Mouzhi Ge, and Martin Hepp

Universitaet der Bundeswehr Munich, 85577 Neubiberg, Germany
{benedicto.rodriguez,mouzhi.ge,martin.hepp}@unibw.de

Abstract. Design-pattern driven ontology construction, whether manual or (partially) automated, relies on the availability of curated repositories of Ontology Design Patterns (ODPs) adequately characterized. In order to consistently apply a given ODP, not only it is important to characterize it in full, but also examine its alignment or deviation to other relevant ODPs in relation to it. Otherwise, possible inconsistencies in the application can lead to interoperability issues among the ontology models involved. In that context, this paper revisits a specific version of three different ODPs: Class as a Property Value (CPV), Value Partition (VP) and Normalisation. The review of the CPV identifies two distinct modelling problems being tangled that prompt to decouple the pattern into two variants: a *strict* and a *coarse* CPV pattern. The examination continues with a comparative analysis among the patterns that reveals key alignments and differences at the structural and semantic level. These findings extends the reusability and compositional characteristics of the strict and coarse variants of the CPV ODP in relation to the other two patterns. To illustrate our contribution existing examples in the literature are revisited. They demonstrate the alignments, differences and prototypical OWL idioms identified, which can assist ontology practitioners in mitigating the opportunity for inconsistencies when applying these recurrent ontology building blocks.

Keywords: ontology alignment, ontology modeling, ontology design pattern, normalisation, value partition, class as property value.

1 Introduction

Ontologies remain as one of the key components needed for the realization of the Semantic Web vision. They bring with them a broad range of development activities that can be grouped into what it is referred to as Ontology Engineering.

Within Ontology Engineering, this research primarily focuses on Ontology Design Patterns (ODPs) [5]. ODPs have evolved from the notion of design pattern defined in [5] as "archetypal solutions to design problems in a certain context" and they are justifiably receiving a significant amount of attention by ontologists due to the preceding success achieved by software design patterns in the context of software engineering [4].

R. Meersman et al. (Eds.): OTM 2012, Part II, LNCS 7566, pp. 682–699, 2012.

In order to enable the consistent application of a given ODP, not only it is important to characterize it in full, but also to examine its alignment or deviation to other relevant ODPs related to it. Patterns that are not fully detailed may not be applied consistently, which can lead to interoperability issues among the ontology models involved. To partially assist with these issues, this paper presents a comparative analysis exercise conducted as part of the work that led to the Faceted Classification ODP introduced in [13]. The exercise revisits three patterns: (a) Class as a Property Value (CPV) as featured in [9]; (b) Value Partition (VP) as featured in [12,2,3]; and (c) Normalisation as introduced in [11] and detailed in [2,3]. These patterns are part of two known existing and documented repositories of ODPs introduced by [3] and [10] respectively, supplemented with a corresponding online version[1].

Starting with the CPV ODP, our examination reveals that two subtly different, yet important modelling problems are being addressed at once. This prompts us to decouple the CPV ODP as presented in [9] into two variants. One that might modify the original semantics of the classes to be reused as property values (referred hereto as coarse-CPV), and another that preserves the original meaning of such classes (referred hereto as strict-CPV).

The review continues with the VP and Normalisation ODPs, and after a process of iterative comparisons of the different pairwise combinations of all patterns, it was noticed that the elements that participate in each one of them, can be organized into five distinctive and recurrent functional groups based on: (a) the analogies in the topology of the ontology structure; and (b) the syntactic and semantic analogies in their prototypical implementation. This analysis reveals significant alignments and deviations across the patterns, extending their characterization along several attributes that to the best of our knowledge, have not been documented so far.

For example, our findings indicate the pervasive and compositional nature that a prototypical instantiation of the strict-CPV ODP presents, positioning this as an essential building block in relation to more complex patterns such as the VP and Normalisation ODPs. In fact, it can be observed that an instantiation of the VP ODP is implicitly also an instantiation of the Normalisation ODP, which in turn it is implicitly also an instantiation of the strict-CPV. This collective *nested doll* effect among the patterns suggests how syntactically, a similar set of OWL idioms are employed by three distinct ODPs to address three conceptually different modelling scenarios.

To illustrate our contribution, we used existing examples of these ODPs in the literature that demonstrate the findings put forward throughout this study. These examples are chosen as context to drive the discussion although the information they convey can be extrapolated to any other instantiation of these patterns as well. Ideally, the outcome of this work will raise the awareness of ontology practitioners when applying this recurrent ontological building blocks and reduce the opportunity for unintended inconsistencies.

[1] `http://odps.sourceforge.net/`
`http://www.ontologydesignpatterns.org/`

The rest of this paper is structured as follows: Section 2, Section 3 and Section 4 provide an overview that describe the elements and main features of the CPV, VP and Normalisation ODPs respectively; Section 5 covers the outcome of the comparative analysis of the generic structure and implementation of the three patterns; and finally, Section 6 concludes the paper with some final remarks and future outlook.

2 Revisiting the Class as Property Value ODP

The Class as Property Value (CPV) ODP was originally introduced in [9] and further discussed in [10]. The modelling scenario that motivates the pattern occurs when an existing subsumption class hierarchy is to be reused as a terminology or as a controlled vocabulary to annotate certain elements of other domain concepts in an ontology.

To illustrate the design scenario, the author in [9] uses as an example an existing subsumption class hierarchy in the domain of "Animals" that is intended to be reused as a subject index to annotate the topic of a collection of specific book items. The author presents five different approaches on how to address this modelling problem and analyses their implications for the resulting ontology model in the context of RDF-S and the various OWL profiles (at the time of writing, OWL Lite, DL or Full, as per the OWL 1 specification [1,14]).

The revision of the CPV ODP throughout this paper focuses on the fourth approach of [9], entitled *"Approach 4: Create a special restriction in lieu of using a specific value"*. Fig. 4 of [9] illustrates the ontology model and the elements that participate in the pattern and the corresponding implementation is made available by the author online[2]. Our focus is set on this approach mainly because of two reasons: (1) it complies with the OWL 1 DL profile; and (2) it enables the automatic classification of books based on their subject by a standard OWL DL reasoner.

Compliance with OWL 1 DL is important, because we are bound to the development and use of ontology models and ODPs within this OWL profile. This limitation is due to risks associated to the migration of existing ontologies models into the OWL 2 DL profile [7,8], in connection to backward compatibility issues and lack of tool support outside OWL 1 DL currently in our context. One of the constrains of OWL 1 DL, is that a class per se, must not be the value of a property. If that is the case, which could be seen as the most intuitive and straight-forward idea (presented as part of Approach 1 in [9]), the resulting ontology model would conform to OWL 1 Full instead[3]. In OWL 1 DL, the value of an object property should be an individual, therefore Approach 4 relies

[2] `http://www.w3.org/TR/swbp-classes-as-values/books4.owl`

[3] It is important to note that the new *punning* feature included in OWL 2, does allow the use of a class directly as the value of an object property within the OWL 2 DL profile. However, as stated above, our focus has to be limited to OWL 1 DL, and the implications of punning support at present, have to be left out of the scope of this paper.

on *anonymous* individuals from the :Animal subsumption class hierarchy as the value of dc:subject to annotate the topic of the various instances from the :Book subsumption class hierarchy. This is the modelling technique chosen in Approach 4 to *approximate* the use of a class as a property value and remain within OWL 1 DL.

As stated earlier, the second reason that motivated our focus on the example pattern of Approach 4, is due to its reasoning capabilities. The implementation of the pattern enables a standard DL reasoner to automatically classify books, (in the example :LionsLifeInThePrideBook, :TheAfricanLionBook) based on their subject (:BookAboutLions, :BookAboutAfricanLions respectively). The key aspects of this implementation are discussed as part of Section 5.

2.1 Decoupling the Class as Property Value ODP

Approach 4 of the CPV ODP includes one important disadvantage that is related to the *implicit* modification of the originally intended semantic of the existing class hierarchy subsumed by :Animal that could take place as a result of applying this pattern. This is, in theory, any instance of :Animal represents originally an *actual* animal in the real world but when an instance of :Animal is used as the value of the property dc:subject in the context of the pattern, it stands for an anonymous generic animal interpreted as the *subject* of a book. The modification to the original meaning of the :Animal class hierarchy is partly linked to the apparent mismatch that exist between: (a) the semantics of the *expected* range of the property that uses these anonymous individuals as values (dc:subject), which in this case such range would be precisely a *subject*; and (b) the original semantics of the classes that provide the values (those subsumed by :Animal), which in this case such sematics stand for an actual *animal* (rather than *a subject*).

Noy already acknowledges in [9], the risks associated to this *semantic overload* referred to above but it is this subtle aspect of Approach 4 that makes us believe that the author might be *coupling* inadvertently two distinct modelling problems into one. That is, on one hand there is (a) the problem of using of a class as a property value per se, which requires its own analysis even if the original semantic of the class is not altered, given that in OWL, such problem can be approximated in various ways, each with its own repercussions. And on the other, there is (b) the issue of not only using a class as a property value, but also, the possibility of altering its original intended meaning in the process as a result. This additional complexity found in (b), could be due to the type of example chosen to discuss the pattern, and linked to the intrinsic challenges associated to the ontological representation of the notion of *subject* as discussed in [15].

***strict-CPV* and *coarse-CPV*.** To differentiate these two cases, from here on the modelling problem in (a) above, is referred to as *strict-CPV* ODP given that it deals *only* with approximating the use of a class as a property value preserving the original semantic of the class; while that in (b) is referred to as *coarse-CPV*

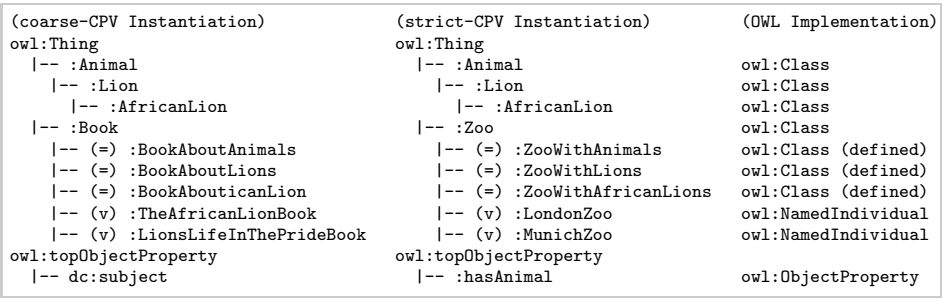

Fig. 1. Elements in the example instantiation of the coarse-, and strict-CPV ODP

given that it deals not only with approximating the use of a class as a property value but *also* with the possibility that the semantic intended originally for the individuals of the class may be altered.

Furthermore, the modelling problem of the strict-CPV ODP can be seen as a particular case of that of a coarse-CPV ODP, where the original semantics of the existing classes that will provide anonymous individuals as property values is preserved. Conversely, the coarse-CPV ODP can be seen as a generalization of the strict-CPV ODP, where the original semantics of the existing classes that will provide anonymous individuals as property values might be modified.

Consequently, Approach 4 of the CPV ODP in [9] is regarded hereafter as an instantiation of the coarse-CPV variant of the pattern.

Example of *strict-* and *coarse-CPV*. To illustrate the decoupling of the original CPV ODP in Approach 4 of [9] into the strict-CPV and the coarse-CPV variants, consider an example in a new domain, so that now instead of book subjects being annotated using an external classification of animals, it is zoological parks of the world being *annotated* using that same external classification of animals, based on the type of animals they exhibit.

Fig. 1 anticipates the elements and the ontological structure that form the new example. The figure illustrates side by side the ontological structure and elements that participate in: (a) the example of Approach 4 of the CPV ODP in [9] and; (b) an analogous example in the new domain of zoological parks. In addition, it notes the prototypical OWL implementation of the elements involved.

Fig. 1 employs a simple and visual notation to convey the key OWL constructs that implement the ontological structure of the patterns. The notation is not aimed at representing every axiom that is part of the ontology model, but simply those relevant in the scope of the pattern at hand. The main motivation for this notation is to facilitate the visual comparison side by side of basic structural and semantic aspects of multiple ODPs in the same figure. The notation is interpreted as follows:

- Nodes indented below owl:Thing denote an owl:Class by default.
- Nodes indented below owl:Thing and marked with the symbol "(v)" denote an owl:NamedIndividual.
- Nodes indented below owl:Thing and marked with the symbol "(=)" denotes a *defined* owl:Class[4].
- The symbol "|--" indented below owl:Thing denotes either: (a) the rdfs:subClassOf relation if the two elements involved are an owl:Class; or (b) the rdf:type relation if the two elements involved are an owl:Class and an owl:NamedIndividual respectively.
- Nodes indented below owl:topObjectProperty denote an owl:ObjectProperty.
- The symbol "|--" indented below owl:topObjectProperty denotes the rdfs:subPropertyOf relation.

The example in this new domain of "Zoo" portrayed by Fig. 1 is built upon applying the following changes with respect to the elements that participate in Approach 4 of [9]: (a) the class :Book is replaced by the class :Zoo; (b) the object property dc:subject is replaced by the property :hasAnimal; (c) the individual books :LionsLifeInThePrideBook, and :TheAfricanLionBook are replaced by the zoo instances :MunichZoo, and :LondonZoo; and lastly (d) the subclasses of :Book, namely :BookAboutAnimals, :BookAboutLions, and :BookAboutAfricanLions, are replaced by the classes :ZooWithAnimals, :ZooWithLions, and :ZooWithAfricanLions respectively.

The example in this new domain of "Zoo" also requires to approximate the use of a class as a property value, similarly to the example in Approach 4 of [9]. The key difference in this case, is that the original semantics of the anonymous individuals from the animal classification hierarchy does not have to be modified as a result of being used as values of the property :hasAnimal. Note that in the new example, the semantic expected for a value of the property :hasAnimal aligns with the original semantic of the class hierarchy subsumed by :Animal. The natural range (rdfs:range) of the property :hasAnimal aligns with what the class :Animal originally represents. The same cannot be said for the property dc:subject in the context of Approach 4. Therefore, the approximation to represent a class as a property value made by the example in the "Zoo" domain corresponds to an application of the strict-CPV ODP variant, while once again, that made by the example in the "Book" domain of Approach 4 corresponds to an instantiation of the coarse-CPV ODP.

Syntactically speaking, the implementation of the coarse-CPV and strict-CPV ODPs, is essentially symmetric. Elements placed at equivalent positions on the ontological structure of both patterns, perform equivalent functions and are implemented following the same set of OWL idioms, However as explained earlier, the semantic implications of each variant can be particularly different. These implementation aspects are covered in more detail throughout Section 5.

[4] In OWL, a *defined* class participates in at least one owl:equivalentClass axiom with respect to another class or class expression, providing at least one set of *necessary and sufficient* conditions for class membership (see §4.10 of [6])

Finally, it is after identifying these two variants of the CPV ODP subtly coupled in [9], that we realized the pervasive nature that a prototypical instantiation of the strict-CPV OPD has in relation to other known ODPs, as will be discussed in the sections that follow.

3 Revisiting the Value Partition ODP

The Value Partition (VP) ODP is introduced in [12] and further revisited in [2,3]. The VP pattern is regarded as a "Good Practice" ODP in the catalog of ODPs in [3]. The pattern is put forward to address the representation of a descriptive feature (also referred to as attribute, modifier or characteristic), of some other entity in the ontology, that is constrained by a set of possible values (known as feature space).

An example in the context of the health condition of a person is used to introduce the pattern in [12]. The concept "health" is the feature to represent and the concepts "good", "medium" and "bad" are the feature space. The author proposes two different versions of the pattern and discusses some advantages and drawbacks of each one, including the OWL expressivity in the resulting ontology models: (a) Pattern 1, where the feature is represented as a class and the feature space as an enumeration of individuals that belong to and exhaust the feature class; and (b) Pattern 2, where the feature is represented as a class and the feature space as a set of pairwise disjoint subclasses that together exhaust the parent (feature) class. This type of class structure is also known as a *partition*.

The revision of the Value Partition ODP throughout this paper focuses on variant 2 of Pattern 2 in [12], entitled "Representation using variant 2: Placing an existential restriction on the individual". Fig. 4 of [12] and Fig. 2 here[5], illustrates the example used by the Pattern 2–Variant 2 mentioned. The example depicts the feature class :Health_Value partitioned by the classes :Poor_health_value, :Medium_health_value and :Good_health_value. These three subclasses represent a value partition of the parent class :Health_Value as per the aforementioned definition: the three are mutually exclusive and their union is equivalent to the parent class :Health_Value. An ontology model is made available online by the author although it implements Variant 1 of Pattern 2 instead[6].

This time, our interest in Pattern 2–Variant 2 is not due to the OWL profile of the resulting ontology model given that all versions of the VP ODP in [12] comply with the OWL 1 DL profile. Our focus is set on this variant of Pattern 2 because: (1) it uses classes instead of individuals to represent the feature space (good, medium, poor) of the feature class (health); (2) it enables the automatic classification of people based on their health status by a standard OWL DL reasoner.

The use of classes to represent the feature space in Pattern 2, allows the class hierarchy subsumed by :Health_Value to be extended further by adding

[5] Fig. 2 introduces the symbol `(P)` to the notation in Fig. 1, to denote that the :Health_Value subsumption class hierarchy is implemented as a *value partition*.

[6] `http://purl.org/net/w3c/odps/vp/pattern2-variant1.owl`

```
(VP Pattern 2 - Version 2)              (OWL Implementation)
owl:Thing
  |-- :Modifier                          owl:Class
    |-- (P) :Health_Value                owl:Class (partition)
      |-- :Good_health_value             owl:Class
      |-- :Medium_health_value           owl:Class
      |-- :Poor_health_value             owl:Class
  |-- :Self_standing_entity              owl:Class
    |-- :Person                          owl:Class
      |-- (=) :Healthy_person            owl:Class (defined)
      |-- (v) :John                      owl:NamedIndividual
owl:topObjectProperty
  |-- :has_health_status                 owl:ObjectProperty and
                                           owl:FunctionalProperty
```

The symbol (P) denotes the class hierarchy subsumed by :Health_Value is a *value partition*.

Fig. 2. Placement of the elements of Pattern 2–Version 2 of the VP ODP in [12]

additional subclasses. In contrast, the use of individuals as in Pattern 1 of [12] does not provide this flexibility. At the same time, the implementation of the feature space via classes in Pattern 2–Variant 2 specifically, prompts the use of anonymous individuals from the :Health_Value sumsumption class hierarchy to indicate a person's health. The use of anonymous individuals and the reasoning capabilities of Pattern 2–Variant 2 in [12], resonate with the similar characteristics outlined in Approach 4 of [9].

4 Revisiting the Normalization ODP

The Normalisation mechanism was firstly introduced in [11] and characterized as an ODP later in [2,3]. Normalisation is regarded as a "Good Practice" ODP in the catalog of ODPs in [3]. The modelling scenario that motivates the pattern occurs when the asserted structure of an ontology model becomes *tangled*, exhibiting a considerable number of poly-hierarchies in which a given class is subsumed by several parent classes. Such model, where the subsumption relations that create these poly-hierarchies are manually asserted, is referred to as a *non-normalised* ontology (i.e. Fig. 6.3 of [3]).

To address this situation, the Normalisation pattern seeks to identify independent *modules* based on *semantic axes* or *principles of division* for the classes with multiple parents, coding the subsumption relations explicitly as restrictions instead of implicitly by hand. Effectively, the goal of the pattern is to allow exactly one unlabelled flavour of *is-a* link, which translates into a single-inheritance structure of the asserted subsumption relations [11]. Such model is referred to as a *normalised* ontology (i.e. Fig. 6.4 of [3]). As a result, the existing poly-hierarchies in the structure of the normalized asserted ontology model are

```
(Instantiation Normalisation ODP)            (OWL Implementation)
owl:Thing
   |-- :Function                              owl:Class
      |-- :Circulation                        owl:Class
      |-- :Defense                            owl:Class
      |-- :StuffAccumulation                  owl:Class
   |-- :Cell                                  owl:Class
      |-- (=) :CirculatingCell                owl:Class (defined)
      |-- (=) :DefensiveCell                  owl:Class (defined)
      |-- (=) :StuffAccumulation              owl:Class (defined)
      |-- :EukaryoticCell                     owl:Class
         |-- :AnimalCell                      owl:Class
            |-- :Neutrophil                   owl:Class
            |-- :SyncitialGiantCell           owl:Class
         |-- :PlantCell                       owl:Class
            |-- :MyrosinCell                  owl:Class
owl:topObjectProperty
   |-- :performs_function                     owl:ObjectProperty
```

Fig. 3. Placement of the elements in the example of the Norm. ODP in [3].

removed and the implementation of the subsumption relations explicitly as restrictions enables a standard DL reasoner to automatically maintain the original poly-hierarchies in the inferred ontology model instead (i.e. Fig. 6.5 of [3]).

To facilitate the comparative analysis to the other patterns, the example of the Normalisation ODP in Appendix A.13 of [3] from the biological domain, is reproduced in Fig. 3 and will be used as context. In the example: (a) the classes with multiple parents causing the poly-hierarchies are :Neutrophil, :SyncitialGiantCell and MyrosinCell; (b) the modules identified as a principle of division (in this case only one) is represented by the class :Function; and (c) the subsumption relations between the classes in (a) and their multiple parents are encoded as restrictions based on their cell function. An OWL implementation of the cited example is made available by the author online[7].

Similarly to the previous patterns, our focus on the Normalisation ODP is motivated by the following factors: (a) compliance with the OWL 1 DL profile; and (b) its reasoning support, by which a standard DL reasoner can be used to maintain the multiple and complex subsumption relations that may exist in the ontology model.

The reasoning support of the Normalisation pattern is partly sustained by the implementation of the manually asserted subsumption relations as restrictions. This implementation prompts the use of anonymous individuals as the value of the property that participate in the restriction. The relevant aspects of this implementation is discussed in Section 5.

[7] `http://purl.org/net/odps.sourceforge.net/Normalisation.owl`

These factors of compliance with OWL 1 DL, the benefits of supporting standard DL reasoning, and using anonymous individuals as property values, appear again in the characteristics of the pattern as they did in the revision of the CPV and VP ODPs. The sections that follow assess the implications of these analogies.

5 Alignment of CPV, VP and Normalization ODPs

The findings of these section are the result of a close examination to several aspects of the four OPDs considered, namely: (1) Approach 4 of the CPV ODP in [9], also referred hereto as coarse-CPV (2) the variant of Approach 4 referred hereto as strict-CPV ODP; (3) Pattern 2–Variant 2 of the VP ODP in [12]; and (4) the Normalisation ODP in [2,3]. The aspects examined include: (a) the elements that participate in each pattern; (b) the underlying ontology structure; and (c) the prototypical implementation of the patterns

Using a holistic view of the four patterns and the various aspects examined, we noticed that all elements that participate in the patterns can be organized into five distinctive groups based on the functionality that they fulfil. They are the result of a process that involved various iterations of comparing the different pairwise combinations of these patterns. These functional groups are: (1) the target domain; (2) the domain elements; (3) the domain *defined* classes; (4) the core property of the pattern, and (5) the range subsumption class hierarchy that will provide the values in the form of anonymous individuals to the property in (4).

Fig. 4 anticipates for three of the four ODPs revisited, the elements that belong to each one of these five functional groups. It provides a side-by-side comparison of the underlying ontology structure of these three patterns. Elements placed at a similar location in the structure perform a similar function in the corresponding pattern and exhibit a similar prototypical OWL implementation (with key differences in some cases, driven by those aspects in which the patterns deviate from each other). The horizontal lines on Fig. 4 delimit the scope of each group and divide the structure of each ontology model by group as per the following top to bottom sequence: range subsumption class hierarchy; target domain; domain *defined* classes; domain elements; and lastly the core property of the pattern. For space reasons, the coarse-CPV ODP pattern is not included in the figure, but based on its structural correlation to the strict-CPV ODP captured in Fig. 1, the information conveyed by Fig. 4 can be easily applied to the former. The sections that follow discuss the various functional groups in detail.

5.1 Patterns Functional Groups

Target Domain. This functional group is formed by a single class that represents the target domain to which the pattern is being applied to, (the overall

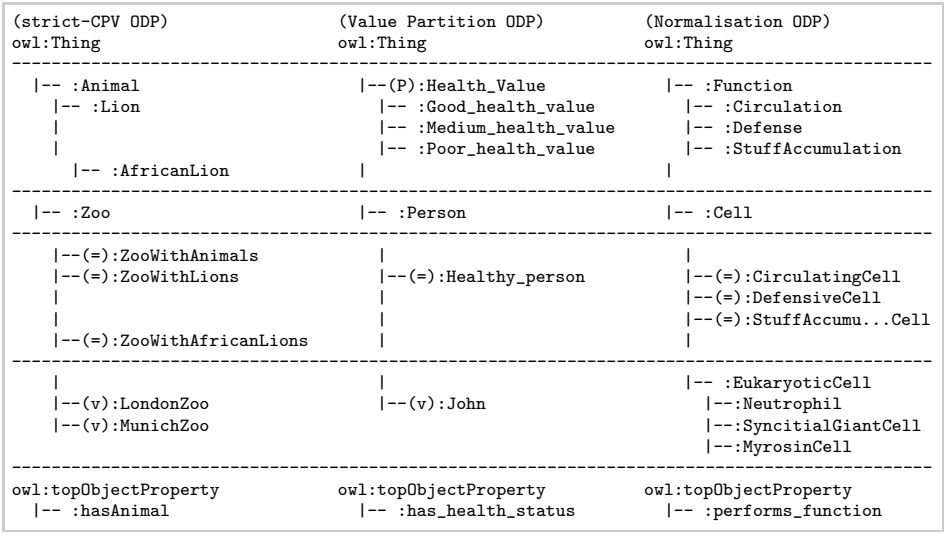

Fig. 4. Alignment of the elements and their functional group in three ODP examples

scope and universe of discourse of the pattern). In terms of the examples in Fig. 4, the target domain class is populated by :Zoo, :Person, and :Cell in the strict-CPV, VP, and Normalisation ODPs respectively. In the case of the coarse-CPV, as Fig. 1 shows, the target domain is populated by the class :Book.

Domain Elements. This group denotes the elements (classes or individuals) of the target domain concept that are annotated using anonymous individuals of a class from the range subsumption class hierarchy group, as values of the pattern core property. They motivate the purpose of the patterns. In terms of the examples in Fig. 1 and Fig. 4, the domain elements are populated by: (a) :TheAfricanLionBook and :LionsLifeInThePrideBook in the coarse-CPV example; (b) :LondonZoo, and :MunichZoo in the strict-CPV example; (c) :John in the VP example; and (d) :Neutrophil, :SyncitialGiantCell and :MyrosinCell in the Normalisation example.

Domain elements can be implemented as a class as in the Normalisation ODP example, or as an individual as in the rest of the patterns, depending on their required semantic in the host ontology model. Listing 1.1 provides the prototypical implementation in N3 *pseudo-code* notation, that can be extrapolated from the implementation of a given domain element, : $DomainElement_i$ in abstract terms, across all four pattern examples:

```
1 : DomainElement_i
2     rdf:type : TargetDomain ,
3         [ rdf:type owl:Restriction ;
4           owl:onProperty : coreProperty ;
5           owl:someValuesFrom : RangeClass_i ] ,
6         [ and rest of existential restrictions
7           on property : coreProperty
8           for every class : RangeClass_i
9           that participates in the description of : DomainElement_i ] .
```

Listing 1.1. Prototypical implementation as an individual of a generic domain element :$DomainElement_i$ in the patterns revisited

Listing 1.1 includes additional abstract terms in the implementation whose names are self-explanatory: (a) : $TargetDomain$ refers to one of :Book, :Zoo, :Person, or :Cell; (b) : $coreProperty$ refers to one of dc:subject, :hasAnimal, :has_health_status, or :performs_function; and (c) : $RangeClass_i$ refers to any class subsumed by one of :Animal, :Health_Value, or :Function.

The use of the restriction owl:someValuesFrom in the implementation of Listing 1.1, is what enables an anonymous individual of the class : $RangeClass_i$, to approximate the use of a class as the value of the property : $coreProperty$.

Note that with the exception of the VP ODP, there might be a *one-to-many* relationship between a given domain element : $DomainElement_i$ and the various classes : $RangeClass_i$ from the range subsumption class hierarchy group that the former is related to via the pattern core property : $coreProperty$. In the case of the VP ODP example, this relationship should be *one-to-one* given that the core property : $coreProperty$ (:has_health_status) is functional (owl:FunctionalProperty) and therefore it should be assigned a single value only.

For example, in the case of the Normalisation ODP, this *one-to-many* relationship is what specifies explicitly the conditions for a domain element to be a member of multiple classes and one of the factors that enables a standard DL reasoner to manage the tangled subsumption hierarchies or poly-hierarchies that the pattern aims to address.

Domain Defined Classes. This group denotes the *defined* classes subsumed by the target domain class. As clarified earlier, a defined class in OWL refers to a class that participates in at least one owl:equivalentClass axiom with respect to another class or class expression, providing at least one set of *necessary and sufficient* conditions for class membership (see §4.10 of [6]). From Fig. 4 together with Fig. 1, it is straightforward to identify the elements that populate this functional group in all four pattern examples. They are marked with the symbol "`(=)`".

The four patterns examples present domain defined classes subsumed by the target domain class with analogous implementation. Listing 1.2 provides the prototypical implementation expressed in N3 *pseudo-code* notation that can be extrapolated from the implementation of a domain defined class across the four patterns examples revisited:

```
1 : DomainDefinedClass_i
2     rdf:type owl:Class ;
3     rdfs:subClassOf : TargetDomain ;
4     owl:equivalentClass
5         [ rdf:type owl:Restriction ;
6           owl:onProperty : coreProperty ;
7           owl:someValuesFrom : RangeClass_i ] .
```

Listing 1.2. Prototypical implementation of a generic domain defined class : $DomainDefinedClass_i$ in the pattern revisited

Listing 1.2 makes use of the same abstract terms as in Listing 1.1. From the names of these abstract terms, it is fairly straightforward to anticipate how they are populated in each pattern. Nonetheless, see notes on Listing 1.1 above for details.

The use of the class axiom owl:equivalentClass in the implementation of a domain defined class : $DomainDefinedClass_i$ in Listing 1.2, together with the use of the restriction owl:someValueFrom in the implementation of a domain element : $DomainElement_i$ in Listing 1.1, enables a standard DL reasoner in all four pattern examples, to infer that : $DomainElement_i$ is also a member of : $DomainDefinedClass_i$, provided the : $RangeClass_i$ is the same on both.

Note that there is a *one-to-one* relationship between a given defined class : $DomainDefinedClass_i$ and the specific class : $RangeClass_i$ that it is related to. However, it is not necessary to implement a domain defined class based on every class that is part of the range subsumption class hierarchy group. For instance, in the VP ODP example, there is no defined class subsumed by :Person based on the concept :Medium_health or :Poor_health. Only the class :Healthy_person is defined in terms of the class :Good_health_value.

Core Property. This functional group is formed by a single object property required by the patterns in order to fulfil their objective. As Fig. 1, and Fig. 4 show, the core property in the examples is populated by dc:subject, :hasAnimal, :has_health_status, and :performs_function in the coarse-CPV, strict-CPV, VP, and Normalisation ODPs respectively.

It is important to note, that in the context of the patterns considered, the VP ODP requires that the core property is functional (owl:FunctionalProperty), so that only one of the mutually exclusive values in the feature space of the partition, can be assigned to the property. Conversely, the coarse-, strict-CPV and Normalisation ODPs does not require any additional characteristic or constrain on the core property.

Range Subsumption Class Hierarchy. This group denotes the set of classes that provide the anonymous individuals that serve as values of the core property to annotate the domain elements of the target domain in the ontology model. It is referred to as *range* because ultimately, the range (in an rdfs:range sense) of the core property is formed by the anonymous individuals that belong to the classes in this subsumption hierarchy and that are used as values of such property.

Fig. 1, and Fig. 4 show the classes that conform the range subsumption hiearchy in the examples. The root or top class of the subsumption hierarchy

is populated by the classes :Animal in the CPV ODP (both strict-, and coarse-variants), :Health_Value in the VP ODP, and :Function in the Normalisation ODP.

This functional group is where the patterns revisited differ the most. These differences can be appreciated by how each pattern implements the classes in this group, going from the VP ODP as the most restrictive implementation, then the Normalisation ODP and lastly, both variants of the CPV ODP.

The VP ODP is the most restrictive because it requires the subsumption class hierarchy in this group, to conform to the definition of value partition as recalled in Section 3. Therefore, in the context of the VP ODP example, the union of all subclasses of :Health_Value exhausts this class and at the same time all of its subclasses are pairwise disjoint.

The Normalisation ODP is next in terms of level of restriction. In this case, the pattern requires a single-inheritance and disjoint structure for all classes that conform this functional group. However, it does not require that the top or root class of the subsumption hierarchy, :Function in the context of the Normalisation ODP example, to be exhausted or covered by its subclasses.

Lastly, both variants of the CPV ODP are the least restrictive patterns given that they do not impose any specific requirement on the classes that form this functional group other than forming a subsumption class hierarchy.

In other words, from the previous statements it follows that: (a) a subsumption class hierarchy that complies with the requirements set out by the VP ODP, also complies inevitably with the requirements of the Normalisation ODP, however the opposite may not apply and; (b) a subsumption class hierarchy that complies with the requirements set out by the Normalisation ODP, also complies inevitably with the requirements of the coarse-, and strict-CPV ODP; while clearly the opposite may not apply.

Anonymous Individuals. The four patterns examples use anonymous individuals of a class from the range subsumption class hierarchy group as values of the core pattern property, to annotate or describe the domain elements from the target domain. In terms of the examples in Fig. 1, and Fig. 4, such functionality can be described for each example as follows: (a) anonymous individuals of the subsumption class hierarchy :Animal are used as values of the core property dc:subject to annotate the domain elements :TheAfricanLionBook and :LionsLifeInThePrideBook; (b) anonymous individuals of the subsumption class hierarchy :Animal are used as values of the core property :hasAnimal to annotate the domain elements :LondonZoo and :MunichZoo; (c) anonymous individuals of the subsumption class hierarchy :Health_Value are used as values of the core property :has_health_status to annotate the domain element :John; and (d) anonymous individuals of the subsumption class hierarchy :Function are used as values of the core property :performs_function to annotate the domain elements :Neutrophil, :SyncitialGiantCell and :MyrosinCell.

Anonymous individuals are another factor of this functional group where the patterns differ especially, and it is due to the motivation for decoupling the CPV ODP into the two variants, coarse-, and strict-CPV addressed in Section 2.1. In

terms of the examples in Fig. 1 and Fig. 4: (a) in the strict-CPV, all anonymous individuals of a given subclass of :Animal represent an animal (an actual lion, an actual African lion, etc.); (b) in the VP ODP, all the anonymous individuals of a given subclass of :Health_Value represent a health value (either good or medium or poor); and (c) in the Normalisation ODP, all the anonymous individuals of a given subclass of :Function represent a (cell) function (circulation, defense or stuff accumulation). However, in the case of (d) the coarse-CPV, some anonymous individuals of a given subclass of :Animal can represent an animal as a subject, as the value of the dc:subject property; while others could in fact represent an actual animal (an actual lion, an actual African lion, etc.).

For these reasons, regarding the semantic of anonymous individuals in the four patterns, the strict-CPV, VP and Normalisation align with each other, while the coarse-CPV deviate from them.

5.2 Reasoning

As recalled in previous sections, the underlying ontology model of the four patterns is within OWL 1 DL. Based on the prototypical implementation of the domain elements in Listing 1.1 and the domain defined classes in Listing 1.2, which are analogous across the four patterns revisited, a standard OWL DL reasoner can automatically infer to which defined subclasses of the target domain the various domain elements belong (or should be classified under).

In terms of the examples in Fig. 4 and Fig. 1, a standard OWL DL reasoner can provide inferences such as: (a) :LondonZoo is a :ZooWithLions and hence, a :ZooWithAnimals; or that :MunichZoo is a :ZooWithAfricanLions and hence, a :ZooWithLions and a :ZooWithAnimals; (b) :John is a :Healthy_person; (c) :Neutrophil is a :CirculatingCell and a :DefensiveCell; or that :SyncitialGiantCell is a :CirculatingCell, a :DefensiveCell and a StuffAccumulatingCell; or that :MyrosinCell is a :StuffAccumulatingCell; and (d) :LionsInThePride is a :BookAboutLions and hence, a :BookAboutAnimals; or that :TheAfricanLionBook is a :BookAboutAfricanLions and hence, a :BookAboutLions and a :BookAboutAnimals;

From a reasoning point of view, all four patterns revisited present a similar behaviour. In that sense, part of the the benefits attributed to the Normalisation ODP are applicable to the others. That is, the subsumption relations that could lead to complex asserted encoded poly-hierarchies are maintained by the reasoner in all four patterns.

5.3 Summary

Table 1 presents a summary of all the functional groups analysed, indicating for every group whether the ODPs are similar to each other or deviate from the rest. If an ODP is different to the rest for a given functional group, a keyword in parenthesis is provided to convey the reason. The rationale for the similarities and differences gathered on the table has been discussed in the previous subsections.

Table 1. Comparative analysis of the coarse-, stric-CPV, VP and Normalisation ODPs

Functional Group	**Similar**	**Different**
Target Domain	All	None
Domain Elements	coarse-CPV, strict-CPV, Normalisation	VP(one-to-one vs. one-to-many)
Domain Defined Classes	All	None
Core Property	coarse-CPV, strict-CPV, Normalisation	VP(owl:FunctionalProperty)
Range Subsumption Class Hierarchy	coarse-CPV, strict-CPV	Normalisation(disjointness), VP(partition)
Anonymous Individuals	strict-CPV, Normalisation, VP	coarse-CPV(altered semantics)

6 Conclusions

This paper has revisited three well known ontology design patterns, namely the CPV, VP and Normalisation ODPs.

Upon a close examination of the example in Approach 4 of [9], it is argued that the modelling problem that motivates the pattern can be further decoupled into two, prompting for the introduction of two variants of the pattern, referred to as: coarse-CPV and strict-CPV. Approach 4 of [9] itself corresponds to an instance of the coarse-CPV due to the potential alteration that could take place in the original semantics of the classes that are intended to be reused as property values, as a result of the application of the pattern. In contrast to Approach 4 of [9], an example of the strict-CPV in the domain of "zoological parks" is put forward. The example shows that the need to use an existing class as a property value does not imply the modification of the original semantics of the class in question.

A revision of two other patterns is carried out, which are: the VP ODP, in particular Pattern 2–Variant 2 of the VP ODP in [12], and the Normalisation ODP as introduced by [11,2,3]. After a review and a comparison of the main characteristics of all patterns, it can be inferred that the elements that participate in each one of them, can be organized into five different functional groups based on the ontological structure of the pattern and the syntactic and semantic function that each element performs.

These functional groups are discussed in detail, analysing the similarities and differences among the patterns on a group by group basis. Existing parallelisms across the patterns are revealed indicating that elements positioned on equivalent locations of the pattern ontology structure, exhibit an equivalent function and prototypical implementation.

The compositional nature of the strict-CPV pattern with respect to the the VP and Normalisation ODPs is presented as well. In that sense, it can be observed that all instantiations of the Pattern 2–Variant 2 of the VP ODP in [12] and the Normalisation ODP in [11,2,3] use implicitly the strict-CPV pattern.

That is, they all approximate the use of an existing class as a property value, relying on an anonymous individual of the class that more importantly, does not require altering the original semantic of the class involved.

Essentially this interrelation among the patterns is reflecting that three different modelling scenarios, (1) the representation of a class as a property value in the strict-CPV ODP; (2) the representation of a descriptive feature in the VP ODP, and (3) the untangling of the poly-hierarchies in a given ontology model of the Normalisation ODP, are being addressed by slight modifications over the same set of OWL idioms, those that follow the strict-CPV ODP. All ODP examples included throughout the paper to guide the discussion, whether from the existing literature or introduced as needed, are used to demonstrate the conclusions outlined.

With this work, we hope to clarify and increase the level of awareness in relation to the use and applicability of these three ODPs by revealing and characterizing key syntactic and semantic aspects that to the best of our knowledge, have not been addressed at a similar level of detail in the ontology development community.

Going forward, the most obvious task is to revisit once again the three patterns under consideration taking into account the *punning* meta-modelling capability available in OWL 2. Punning allows the use of the same URI identifier to represent an owl:Class and an owl:NamedIndividual within the OWL 2 DL profile. The three ODPs in this comparison can leverage this feature, and thus the repercussion at different levels in the resulting ontology models should be reassessed.

Additionally, the plan is to increase the number of ODPs that participate in this comparative analysis. To do so, a survey of the available ODPs in known repositories such as [3,10] can provide further suitable candidates. At a larger scale, the factors that have been discussed from the five functional groups characterized as part of the comparative analysis presented for the four pattern examples, could evolve into a more formal evaluation framework extending the current ODPs evaluation and documentation templates found in [3,10].

Acknowledgements. The outcome of this research was partly influenced by the examination process of the PhD thesis[8] carried out at the University of Southampton (UK) by one of the authors. Thus, we would like to thank for their helpful comments and suggestions, the examination panel formed by John Domingue and Nicholas Gibbins; and the supervisors Hugh Glaser and Leslie Carr.

References

1. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C recommendation, W3C (February 2004), `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`

[8] `http://purl.org/beroca/phd/thesis/final/`

2. Egana-Aranguren, M.: Ontology Design Patterns for the Formalisation of Biological Ontologies. MPhil Dissertation, Bio-Health Informatics Group, School of Computer Science, University of Manchester (2005), http://www.gong.manchester.ac.uk/doc/MPhil_thesis.pdf
3. Egana-Aranguren, M.: Role and Application of Ontology Design Patterns in Bio-ontologies. Ph.D. thesis, School of Computer Science, University of Manchester (2009), http://mikeleganaaranguren.files.wordpress.com/2010/01/thesis.pdf
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (1995)
5. 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), http://dx.doi.org/10.1007/11574620_21
6. Horridge, M., Drummond, N., Jupp, S., Moulton, G., Stevens, R.: A practical guide to building owl ontologies using the protege-owl plugin and co-ode tools edition 1.2. Tech. rep., The University of Manchester (March 2009), http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_2.pdf
7. Krötzsch, M., Patel-Schneider, P.F., Rudolph, S., Hitzler, P., Parsia, B.: OWL 2 Web Ontology Language Primer. W3C recommendation, W3C (October 2009), http://www.w3.org/TR/owl2-primer/
8. Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., Grau, B.C.: OWL 2 web ontology language profiles. W3C recommendation, W3C (October 2009), http://www.w3.org/TR/owl2-profiles/
9. Noy, N.F.: Representing Classes As Property Values on the Semantic Web. Technical Report Note 5, W3C, Semantic Web Best Practices and Deployment Working Group (2005), http://www.w3.org/TR/swbp-classes-as-values/
10. Presutti, V., Gangemi, A., David, S., de Cea, G.A., Suarez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NeOn deliverable D2.5.1, Institute of Cognitive Sciences and Technologies, CNR (2008), http://www.neon-project.org/web-content/images/Publications/neon_2008_d2.5.1.pdf
11. Rector, A.: Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. In: Proceedings of the 2nd International Conference on Knowledge Capture, K-CAP 2003, pp. 121–128. ACM, New York (2003), http://doi.acm.org/10.1145/945645.945664
12. Rector, A.: Representing Specified Values in OWL: "value partitions" and "value sets". Technical Report Note 17, W3C, Semantic Web Best Practices and Deployment Working Group (May 2005), http://www.w3.org/TR/swbp-specified-values/
13. Rodriguez-Castro, B., Glaser, H., Carr, L.: How to *Reuse* a Faceted Classification and Put It on the *Semantic* Web. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 663–678. Springer, Heidelberg (2010), http://eprints.soton.ac.uk/271488/
14. Welty, C., McGuinness, D.L., Smith, M.K.: OWL Web Ontology Language Guide. W3C recommendation, W3C (February 2004), http://www.w3.org/TR/owl-guide/
15. Welty, C.A., Jenkins, J.: Formal ontology for subject. Data & Knowledge Engineering 31(2), 155–181 (1999), http://dx.doi.org/10.1016/S0169-023X(99)90021-6