Entity Naming in Semantic Web Ontologies: Design Patterns and Empirical Observations

Vojtěch Svátek and Ondřej Šváb-Zamazal

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

Abstract. We systematically analyse the entity naming options over the structure of the OWL ontology language, both at the level of entity types (classes, properties and individuals) and simple structures such as inverse property or domain/range axioms. We attempt to distinguish what is good and bad practice in entity naming. Finally, we partially compare our assumptions with the reality of OWL ontology design.

Keywords: Ontological Engineering, Natural Language, Design Patterns

1 Introduction

During the decades of knowledge engineering research, there has been recurrent dispute on how the natural language structure influences the structure of formal knowledge bases and vice versa. A large part of the community seems to recognise that the content expressed in formal representation languages should be accessible not only to logical reasoning machines but also to humans and NLP procedures, and thus resemble the natural language as much as possible.[1] On the other hand, even experts in ontological engineering sometimes seem to claim

1. that arbitrary strings could safely be used for entity names (i.e. local URIs)
2. or, at least, that natural-language-based entity names, even if being of some help for people, are totally opaque to machines.

In this paper we claim that naming in ontologies matters and that it can be sensibly designed, or even, with some degree of success, automatically identified in existing ontologies, with the help of *ontology naming patterns* linked to logical patterns of the given language. By the *OntologyDesignPatterns.org* (ODP) portal categorisation, ontology naming patterns (Naming OPs) are "good practices that boost ontology readability and understanding by humans, by supporting homogeneity in naming procedures". The present work attempts to pave the road to systematically populating this category of design patterns. We are not aware of a previous systematic effort to either map the usage of naming patterns in existing ontologies or build a library of such patterns to guide the designers,

[1] See for examples the arguments by Y. Wilks in [8].

aside a handful of ad hoc hints as in [9] or sideways mentions of naming in connection with modelling error descriptions [17]; a notable exception are the initiatives led by the bioinformatics community [11] for their domain ontologies.

As we will demonstrate later, meaningful names are helpful for both *people* and *machines*. Note that some user-facing initiatives in ontological engineering, such as the introduction of Manchester syntax for OWL [1], use natural-language-like features to improve the human readability at the level of *meta-model constructions*. Naming patterns could play an analogous role at the level of *model entities*. Even software tools that analyse and process ontologies can benefit from a resource of information different from (but correlated to) logical constructs. Note that aside pure (deductive) logical reasoning, automated semantic processing of ontology content is needed e.g. for

- Detection (and even suggestion of repair) of possible *conceptual mismatches*
- Automated *alignment* and (modular) *importing* of ontologies
- Model *transformation*, e.g.
  - for better alignment [15]
  - for better tractability by a reasoner.

Regarding the first purpose, note that many *conceptualisation errors* are not manifested by logical inconsistency. Naming analysis alerts on both conceptualisation errors and awkward naming—though it typically cannot discern between the two. Regarding the second and third purposes, heuristic-based processing of ontologies such as automated alignment, importing and transformation typically requires human assistence in *selecting among alternative operations*. To reduce the number of alternatives, or to rank them, even subtle evidence such as entity naming should be exploited.

The paper is structured as follows. Section 2 brings two motivating examples. Section 3 outlines our principles of categorising naming patterns. Sections 4 and 5 characterise different categories of patterns (single-entity and cross-entity ones), including examples from existing ontologies and empirical evidence from analysis of a collection of ontologies. Section 6 is devoted to discussion of conceptual and technical issues related to exploitation of naming patterns. Finally, Section 7 wraps up the paper and outlines directions for future research.

2 Motivating Examples

We will demonstrate the potential of adequate naming on both elementary triples (A-box axioms) and T-box axioms. Consider first the following *elementary triple*:

Example 1. `IBM product ThinkPad700T`.

How would an outsider, ignorant of the identity of the subject and object (which can often be the case for less known companies and products), interpret this as statement? One possibility is, 'IBM has a product called ThinkPad700T'; the other, perhaps equally probable, however is, 'IBM is a product of ThinkPad700T'. Obviously, using a plain noun (such as 'product') as object property name is problematic here. Ambiguity for humans in such a situation can be reduced

- by involving more logical context, for example the neighbour triples:
  - `IBM rdf:type Company`
  - `ThinkPad700T rdf:type ComputerModel`
- or, by including annotation property triples such as `product dc:description ''Relates a company to a product it manufactures.''`

However, displaying such a context to the user is often cumbersome or even unsupported by tools. On the other hand, names as URI fragments are always there by rule and every tool thus has to support them. There can be multiple adequate naming patterns; two possibly most obvious ones could be applied as:

- `IBM produces ThinkPad700T`
- `IBM hasProduct ThinkPad700T`

The semantic direction of the relation is made clear in both cases.

Consider now an example of *T-box axiom* in Manchester syntax:

Example 2. `StateOwned Director only (nomination some ministry)`

With more careful naming the same axiom could look like this:

Example 3. `StateOwnedCompany hasDirector only (nominatedBy some Ministry)`

Presumably, this version much more clearly conveys the message that "all directors of state-owned companies are nominated by some ministry". What has made the difference?

- the head noun ('company') of composed term 'state-owned company' is explicitly present[2]
- plain nouns as object property names ('director', 'nomination') were avoided, cf. Example 1
- capitalisation was made consistent for the same entity type (capitalised class names and decapitalised property names).

Hopefully these two examples demonstrated that ontological entity naming is not a trivial issue and that some patterns can sensibly be formulated for it. In the following three sections we provide a tentative categorisation of such patterns.

3 Naming Pattern Categorisation Criteria

In this first approximation we suggest to categorise naming patterns along four interdependent dimensions:

1. by structural complexity and underlying (modelling) language construct
2. by lexical specificity and linguistic depth
3. by domain specificity

[2] We discuss later that relying on superclass name as supplement to subclass name (`StateOwned subClassOf Company`) is a common type of naming sloppiness.

4. by descriptiveness/prescriptiveness.

In this paper we use the *structural complexity* of the pattern and underlying *language construct* (from the meta-model) as the primary categorisation criterion, as it is rather crisp. In this respect, we distinguish between generic naming conventions, single-entity patterns related to different entity types (classes, object properties, data properties and instances) and cross-entity patterns related to constructs such as class-subclass pairs or pairs of mutually inverse properties. For the moment, we do not systematically cover patterns defined on the top of more than two directly connected entities, although some complex patterns (linked to common logical patterns) have already been formulated ad hoc and empirically tested in our previous research [14]. We also assume the underlying language to be OWL, although naming patterns are obviously, compared to logical patterns, less sensitive to shifting to a different language (say, with different formal semantics but similar outlook, as is the case with frame languages).

Patterns can differ in their *lexical specificity*. Some refer to concrete lexemes, which can be both 'stop words' (such as 'is' or 'of') and 'semantic' words (such as 'part'); on the other hand, some patterns only refer to parts of speech. The *linguistic depth* of patterns may span from surface attributes of strings such as capitalisation or presence of numerals to patterns referring to deeper linguistic notions such as active/passive mode of verbs.

Some naming patterns can certainly be characteristic for problem *domains*, say, engineering or genomics. We do not consider this aspect here.

Finally, in our pattern system we include both patterns that have been, at least tentatively, empirically verified as 'frequent' in *existing ontologies*, i.e. 'descriptive' patterns, and patterns that we see as useful as guidance for developing *new ontologies* (or reengineering old ones) even if they are not widely used nowadays, i.e. 'prescriptive' patterns. We do not make a sharp distinction between the two for the moment, as we believe that naming patterns should on the one hand try to accomodate what is intuitive for most modellers (and thus widely used) and on the other hand promote clarity and readability even at the cost of going against the mainstream.

4 Single-Entity Patterns

Single-entity patterns are the type of pattern most often (if not only) mentioned in the literature for ontology designers. In this paper we focus on patterns specific for a particular entity type, while generic naming conventions have already been well covered by prior literature including the earlier (short) version of this paper [12].

4.1 Class Naming Patterns Patterns

Class Naming Patterns – General The central issue in naming classes is whether the name of a class should imperatively be a noun phrase and whether it should be in singular or plural. The presence of specific lexemes, see below, could also be considered as (rarely applied but possibly strong) heuristics.

While [9] mentions the plural as a valid alternative, we would strongly encourage *singular* for OWL ontologies: first, it is nowadays predominant in existing ontologies; second, some linear RDF notations such as N3[3] expect it in their syntax by using the 'a' (indefinite article) token for instance-class relationship, such as "John a Person" (John is an instance of class Person). On the other hand, there are situations where a syntactical *plural* is fully justified for a class name. Let us consider 'Bananas' as subclass of 'FruitMeal' in a catering ontology: here, multiple physical entities (bananas) play the role of a single object (meal) and do not matter individually. Moreover, there are situations where a noun in plural is simply used as a label that does not have direct semantic linkage to the concept it denotes, as in 'FourSeasons', which is a kind of pizza in the famous Pizza ontology.[4] Therefore we do not encourage ontology editors to strictly enforce the singular.

There does not seem to be any logical reason for using another part of speech than noun for class name. Modellers sometimes omit the noun if it is present at a higher level of the hierarchy, and only use the specifying *adjective*, such as 'Rejected' being the name of a subclass of 'Paper' in a conference management ontology[5], or 'StateOwned' as subclass of 'Company' in the Example 2 in Section 2. We however discourage from such shorthanding. First, for elementary comprehensibility reasons illustrated on Example 2. Second, even if frame-based ontology engineering is tolerant in this respect ([9] for example only discourages from incomplete shorthanding, such as having both 'RedWine' and 'White' as subclasses of 'Wine'), note that in OWL ontologies, due to the underlying description logics, the explicit taxonomy is only secondary to axiomatisation as such. Making a concept anyhow dependent (even in a 'harmless' manner, such as in terms of naming) on its parent concept is thus rather awkward.

On the other hand, entity names consisting of *too many* tokens are also undesirable. It may be the case that they could be transformed to *anonymous classes* as part of axioms, see for example 'FictionalBookbyLatinAmericanAuthor' mentioned by Welty [17] or linguistic disjunctions mentioned below.

Some issues related to automated detection and repair of name shorthanding will be further discussed in Section 5.1 in connection with the class-subclass pattern. We also return to the problem of naming verbosity (caused among other by avoiding shorthanding) in Section 6.

Class Naming Patterns – Specific Lexemes In this paragraph we merely enumerate some tokens that repeatedly occur in various ontologies and thus can be understood as single-entity (mostly, anti-) patterns.

– The logical connective 'or', i.e. linguistic *disjunction*, can presumably appear in class names for two due reasons. First, by committing to the nam-

[3] `http://www.w3.org/2000/10/swap/Primer`
[4] `http://www.co-ode.org/ontologies/pizza/`
[5] See `http://nb.vse.cz/~svabo/oaei2009/` for a collection of such ontologies.

ing used in legacy resources, e.g. biomedical ones such as UMLS[6] ('Body-Part_Organ_OR_OrganComponent', 'GeneOrGenome'). Second, possibly, for parsimony reasons: there would have been two distinct classes that would however not differ at the level of axioms considered, and the designer did not want to write the same axioms twice.

- 'Other' in class name (typically followed by superclass name). Such constructions seem undesirable as they suggest to inappropriately *close the superclass* and may cause maintenance issues; if a new sibling class is added later, it could only be populated by migrating instances that originally belonged to the 'Other' class. This pattern is probably used by designers who assume that subclasses must always generate a partition of the parent class.
- 'Part' in class name signals that the class is likely to be a *role* rather than a natural class. This may be associated with rigidity constraint violation according to the OntoClean methodology [6]; for example, as observed by Welty [17], various classes possibly ranged under 'ComputerPart' can also appear as parts of other technical systems. However, it can also be legal in situations when it denotes a quantitative part (aka amount), for example 'PartOfDNAPopulation'.
- Class names having terms such as 'type' or 'category' in their names are suspects for 'instantiation pitfall' again described by Welty [17]: it is quite likely that the subclasses of the given class should rather be its instances.
- Class names corresponding to *datatypes*, e.g. 'Date' or 'String', should better be replaced by those datatypes proper; they may have arisen by conversion from languages not supporting datatypes.

Instance Naming Patterns The problem of instance naming is closely related to the problem of including instances (i.e. individuals as first-class citizens) as part of ontology. It is generally agreed that instances should not be part of an OWL ontology proper unless they are needed in an axiom: typically the description of an enumerated class or a property value restriction. Such individuals are often *standard vocabulary noun phrases*, such as chemical elements or political countries. In very specific ontologies (or ontologies that are melted with a specific knowledge base) instance names could also be *non-linguistic strings* such as names of genes or product codes.

Individuals are sometimes also used for *specified values*, as depicted in the corresponding logical pattern [10]. Then the enumerated individuals (usually declared as different from each other) define a class; e.g. the set of individuals 'poor_health', 'medium_health' and 'good_health' defines the class 'Health_value'. A subtle issue is whether the name of such an individual can be other than noun phrase. It seems that if an individual is to denote a mere 'value' or 'status' rather than a real-world entity, the part of speech of its name does not matter in principle. However, using plain adjectives such as 'good' or 'high' is tricky. Note that there is a risk of confusing such individuals with the general notions of 'goodness' or 'highness'; this is emphasised by the status of individuals as first-class citizens

[6] `http://umlsks.nlm.nih.gov/`

in OWL. It may then easily happen that an individual originally defining a specified value with respect to a certain class would be *improperly reused with respect to another class*. For example, in a wine ontology[7] the individual Light is part of enumeration of class WineBody; then someone might reuse the same individual as part of enumeration of WineGrape, or even of WineBottle or anything that can be light or heavy (set aside possible confusion with lightness in terms of colour). Clearly, the lightness values of wine body are ontologically different from the lightness values of a wine grape; and even the physical lightness values of a wine grape are ontologically different from the lightness values of a wine bottle, as each of them is associated with a different scope of weight (as measurable quantity). For this reason we recommend to refer to the name of class in the name of the individuals representing specified values. On the other hand, there is a risk of confusing the 'value' or 'status' individuals with *real-world entities*; for example the individual representing the *status* of 'excellent student' should probably not be an instance of class Student. A safe option for naming such individuals thus would be to include both the class name and a term such as 'status' or 'value' in their name, e.g. 'poor_health_value', 'excellent_student_status' or 'light_wineBody_value'.

Finally, another, more parsimonious alternative for modelling specified values would be to resort to *datatypes*; then it would be no problem using just short strings ('poor'; 'excellent'; 'light'), as datatype values, unlike object individuals, are not assumed to map to real-world entities. Thanks to gradually improving support of datatype reasoning in current tools such as Fact++[8] or Pellet[9], this option has nowadays fewer drawbacks than before.

Property Naming Patterns Although object properties and data properties have similar status in OWL, their naming seems to be linked to different patterns.

Comprehensibility concerns suggest that the name of an object property should not normally be a plain noun phrase, for clear discernability from class names as well as from the name of the inverse property; we observed the latter in Example 1 in Section 2, where the plain noun 'product' could be understood both as 'is product of' and 'has product'. Indeed, a majority of object properties either have a *verb* as their head term or end with an attributive *preposition* (such as 'of', 'for'), which indicates that the name should be read as if it started with 'is': for example '(is) friend_of', '(is) component_for'. A *plain preposition* is occasionally used for spatio-temporal relationships. The only type of *noun phrase* 'linguistically adequate' as name for object property seems to be that having the (head noun of) domain class in appositive, for example 'bookAuthor' as name for property having 'Book' as its domain, assuming we want to explicitly distinguish between the authorship of books and, say, articles, at the level of object property (which is not common). Otherwise 'hasAuthor' looks more natural. By a survey of nearly 5000 object properties from 295 ontologies downloaded from

[7] `http://www.ninebynine.org/Software/HaskellRDF/RDF/Harp/test/wine.rdf`
[8] `http://owl.man.ac.uk/factplusplus`
[9] `http://clarkparsia.com/pellet/`

the web, we estimate that their name starts with 'has' in about 15% of cases.[10] Furthermore, linguistic processing of ontologies would possibly benefit from the usage of *content verbs* rather than auxiliary ones where appropriate, as content verbs bring additional lexemes into the game. In this sense, property names like 'manufactures' or 'writtenBy' bring 'extra calories' compared to property names like 'hasProduct' or 'hasAuthor' (assuming that the range of the properties is 'Product' and 'Author', respectively). We elaborate further on object properties in the paragraph on naming patterns over restrictions in Section 5.3.

On the other hand, for *data property* names *nouns* seem appropriate, as they are analogous to database fields. Often the 'primitive data' nature of data properties can be underlined by using head nouns such as 'date', 'code', 'number', 'value', 'id' or the like.

5 Cross-Entity Naming Patterns

In this section we align entity naming with simple logical structures. The coverage of structures rather than individual entities potentially allows to measure the degree of correlation between the logical structures and naming structures; we elaborate in this issue for class-subclass and property restriction patterns.

5.1 Class-Subclass (and Class-Instance) Naming Patterns

It is quite common that a subclass has the *same head noun* as its parent class.[11] By an earlier study [13] we found out that this pattern typically represents between 50–80% of class-subclass pairs such that the subclass name is a multi-token one. This number further increases if we consider *thesaurus correspondence* (synonymy and hyperonymy) rather than literal string equality. In fact, the set-theoretic nature of taxonomic path entails that the correspondence of head nouns along this path should be close to 100% in principle, if mediated by an 'ideal' thesaurus. Sometimes the head noun also disappears and reappears again along the taxonomic path, as a specific concept cannot be expressed by a dedicated term but only by circumlocution; for example in *Player - Flutist - PiccoloPlayer* (note that `Flutist` is a single-token name, i.e. not in conflict with our pattern), in a music ontology.[12] Retrospectively, violation of the head noun correspondence pattern in many cases indicates a problem in the ontology. Common situations are:

– Inadequate use of class-subclass relationship, typically in the place of whole-part or class-instance relationship, i.e. a *conceptualisation error*.
– *Name shorthanding*, typically manifested by use of adjective, such as 'State-Owned' (subclass of 'Company'), as mentioned above.

[10] Let us remark that [9] suggests to use the 'has' prefix or 'of' suffix just for the sake of distinguishing properties (slots) from classes.

[11] The head noun is typically the last token, but not always, in particular due to possible prepositional constructions, as e.g. in 'HeadOfDepartment'.

[12] `http://www.kanzaki.com/ns/music`

While the former probably requires manual debugging of the ontology, the latter could possibly be healed by propagation of the parent name downto the child name, as was previously suggested in non-ontological resource reengineering [7]. Note that such propagation may not be straightforward if the parent itself has a multi-word name. For example, 'MD_Georectified', which is a subclass of 'MD_GridSpatialRepresentation',[13] could be extended to 'MD_GeorectifiedRepresentation', 'MD_GeorectifiedSpatialRepresentation' or 'MD_GeorectifiedGridSpatialRepresentation', and only deep understanding of the domain would allow to choose the right alternative.

A less commonplace aspect of class-subclass naming is associated with the *OntoClean* methodology [6]. We previously mentioned an example of OntoClean-related issue: a class having 'Part' as head noun of its name should only have subclasses with the same feature (e.g. 'HardDiskPart' rather than 'HardDisk'), or rather be transformed to property.

The *class-instance* relationship does not seem to follow generic naming patterns. An exception is the case of specified values discussed in Section 4.1.

5.2 Subproperty and Inverse Property Naming Patterns

We are not aware of a conspicuous naming pattern for the property-subproperty relationship. A tentative suggestion for reengineering methods could perhaps be the following: if there are multiple (object or data) properties with same head noun (depending on a usual auxilliary verb), they could possibly be *generalized* to a superproperty. For example, the properties 'hasFirstName', and 'hasFamilyName' could yield 'hasName' as superproperty.

Inverse property naming patterns helps link a property to its inverse and at the same time discern between the two. They are thus related to the logical design pattern of *bi-directional relations*: if there is no inverse property, there is less of problem at the level of naming but more at the logical level. As canonical inverse property naming patterns we can see the following:

- active and passive form of the same *verb*, such as 'wrote' and 'writtenBy'
- same *noun phrase* packed in auxilliary terms (verbs and/or prepositions), such as 'memberOf' and 'hasMember'.

If the nominal and verbal form are mixed, e.g. 'identifies' and 'hasIdentifier', the accessibility is fine for humans but worse for NLP procedures.

5.3 Naming Patterns over Restrictions

As we mentioned in the subsection on property naming, one alternative for object property name is that including the name of the class in the range and/or domain of this property. This can be seen as a naming pattern over a *global*

[13] Taken from `http://lists.w3.org/Archives/Public/public-webont-comments/2003Oct/att-0026/iso-metadata.owl`.

property restriction. Let us illustrate some options for such patterns on the notorious pizza domain. We suggest that the property from PizzaTopping to Pizza can be labelled as:[14] 'isToppingOfPizza'; 'isToppingOf'; 'toppingOf'; or maybe 'ofPizza'. Intuitively, we probably feel that 'hasPizza' does not sound well. On the other hand, for the inverse property we would rather suggest[15] 'hasTopping' or maybe 'PizzaTopping'. As possible reasons for the different natural (?) choice of naming pattern for the mutually inverse properties we could see the nature of topping as 1) an entity *dependent* on a pizza entity (a topping cannot exist without a pizza), or 2) a *role* entity (as being a pizza topping is merely a role of some food). The first hypothesis would mean that the presence of the name of a class in the name of a property (for which this class is in the domain or range) indicates that entity of this class is dependent on the entity on the other side of the property. The second hypothesis would mean that the presence of the name of a class in the name of a property (for which this class is in the domain or range) indicates that this class is a role. Both hypotheses can also be adjusted according to presence of auxilliary verbs ('is', 'has') and suffixed propositions.

Note that the validity of either hypothesis would open the way to using entity naming as support for assigning *OntoClean labels*; this could be an interesting complement to existing support to OntoClean by environments for manual annotation [5] and text mining from large corpora [16].

We attempted to checked if the possible position of the 'domain' entity as *dependent* is reflected in the logical declaration of the respective property as functional. There was however no empirical evidence for this correlation: the proportion of functional properties was even lower among the object properties satisfying this pattern. On the other hand, an increase of the proportion of functional properties (from 5% to 11%) was observed for object properties starting with 'has', which was originally unexpected. The correlation test was found to be impacted by the low degree of axiomatisation of properties in general (only 5.5% of nearly 5 thousand object properties in our study were declared as functional; in reality this number should be much higher) as well as by the high number of object properties having their name in the form of plain noun (thus governed by other principles than our 'pizza' example).

We also manually tested the second hypothesis, in a refined form: the naming pattern was "is ⟨domain class name⟩ ⟨preposition⟩". From about 20 'domain' classes of such properties, nearly all could be seen as *roles* rather than natural concepts: 'member', 'agent', 'pet', 'capital', 'secretary', 'factor', 'cause', 'consequent', 'explanation', 'location' etc. We however did not yet evaluate the apriori proportion of 'role' concepts among classes in the whole corpus.

In principle, we could also identify naming patterns over *local property restrictions*, for example in the form of 'lexical tautologies' such as `MushroomPizza equivalentTo (Pizza and contains some Mushroom)`. This issue may deserve further study, although the frequency of such constructions is not very high.

[14] Let us for simplicity ignore the naming options with alternative prepositions ('isToppingOn') or without domain/range tokens at all ('laidOn', 'on').

[15] Again ignoring essentially different options such as 'withTopping' or 'laidWith'.

6 Discussion

Although not explicitly separated in the formal structure of this paper, the role of the ontology naming patterns is twofold: first, that of *best practices* to be available to designers of future ontologies; second, that of *empirical patterns* that would assist in the automated analysis of existing ontologies. These two aspects can obviously conflue: a possible scenario is to apply naming pattern analysis on a *freshly developed* (and not yet widely used) ontology, thus alerting the designers of problematic issues. Indeed, adoption of naming patterns in this scenario seems more straightforward than that of other kinds of ontology patterns (say, logical, never mind content ones), as posterior alteration of entity names is much less demanding (and has less dramatic consequences) than changes in the logical structure of an ontology. On the other hand, for an already *published* (and used) ontology, changing URI fragments is problematic,[16] as it may invalidate references in existing semantic documents. To some degree, this controversy could be solved using ontology *versioning* principles; the original entity can be left in the ontology, labelled as deprecated, and declared as equivalent to the new, more adequately labelled entity.

Another problem is that adherence to 'information-rich' naming patterns can lead to *verbosity*. Long labels, especially at lower layers of the concept hieararchy, can even be hard to display in some settings. This problem could presumably be healed by using specific annotation properties referring to 'folded' entity names, while URIs would still retain the whole lexical information. Switching to the values of such properties then would be within the scope of visualisation/editing tools; even nowadays some of the tools allow to switch between the display of URI fragments vs. values of rdf:label.

7 Conclusions and Future Work

The contribution of the paper is a preliminary system of ontology naming patterns, which we illustrated on examples and partly associated with empirical findings. Undoubtedly, consistent and comprehensible entity naming is an important aspect of re/usability of ontologies. The main reason why research on this topic has been quite scarce to date is probably the high risk of subjectivity and subtle nature of any cues one could figure out. We are aware of this risk; the suggestions in this paper are meant to serve as starting point for discussion in the pattern community rather than a mature system of best practices.

The most imminent future work will consist in more systematic *large-scale evaluation* of existing ontologies in terms of naming as well as bare (syntactical) logical patterns. This will among other allow to more sharply delineate the 'descriptive' and 'prescriptive' aspects of the research. Within the empirical analysis stream, we should also study the usage of *other textual labels* rather

[16] It is characteristic that even ontology editing tools such as Protégé (`http://protege.stanford.edu`) understand entity renaming as 'reengineering' rather than simple editing.

than URI fragments (such as rdf:label and rdf:description), and compare their content with that of the URIs. We would also like to set up a specific *metadata schema* for collecting this type of patterns in the *ODP portal*. Finally, in the context of this portal, we would like to apply the naming patterns to evaluate *other types of ontology design patterns*, especially the content ones.

This work has been partially supported by the IGA VSE grant 20/08 and by the CSF grant P202/10/1825 (PatOMat project). The authors are also grateful to the attendees of the ESWC'09 tutorial on "Extreme Design (XD): Pattern-based Ontology Design" for interesting discussion reflected in this paper.

References

1. The Manchester OWL Syntax. Online `http://www.co-ode.org/resources/reference/manchester_syntax/`
2. Annotation System. OWL WG Work-in-Progress document, `http://www.w3.org/2007/OWL/wiki/Annotation_System`.
3. Buitelaar P., Cimiano P., Haase P., Sintek M.: Towards Linguistically Grounded Ontologies. In: ESWC'09, Crete.
4. d'Aquin M., Gridinoc L., Angeletou S., Sabou M., Motta E.: Characterizing Knowledge on the Semantic Web with Watson. In: EON'07 Workshop at ISWC'07.
5. Fernandez-Lopez M., Gomez-Perez A.: The integration of OntoClean in WebODE. In: EON'02.
6. Guarino N, Welty C.: An Overview of OntoClean. In: The Handbook on Ontologies, 151-172. Berlin:Springer-Verlag, 2004.
7. Hepp M., de Bruijn J.: GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In: Proc. ESWC 2007.
8. Nirenburg S., Wilks Y.: Whats in a symbol: Ontology and the surface of language. *Journal of Experimental and Theoretical AI*, 2001.
9. Noy N. F., McGuinness D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Online `http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf`
10. Rector, A. (ed.): Representing Specified Values in OWL: "value partitions" and "value sets". W3C Working Group Note, 17 May 2005, online at `http://www.w3.org/TR/swbp-specified-values/`.
11. Schober D. et al.: Survey-based naming conventions for use in OBO Foundry ontology development. *BMC Bioinformatics*, Vol.10, Issue 1, 2009.
12. Svátek V., Šváb-Zamazal O., Presutti V.: Ontology Naming Pattern Sauce for (Human and Computer) Gourmets. In: Workshop on Ontology Patterns at ISWC'09, Washington DC, 2009.
13. Šváb-Zamazal O., Svátek V.: Analysing Ontological Structures through Name Pattern Tracking. In: EKAW-2008, Acitrezza, Italy, 2008.
14. Šváb-Zamazal O., Svátek V.: Towards ontology matching via pattern-based detection of semantic structures in OWL ontologies. In: Znalosti 2009, Brno.
15. Šváb-Zamazal O., Svátek V., Scharffe F.: Pattern-Based Ontology Transformation Service. In: KEOD-09, Madeira, Portugal.
16. Völker J., Vrandečić D., Sure Y.: Automatic Evaluation of Ontologies (AEON). In: ISWC'05, Galway.
17. Welty C.: Ontology Engineering with OntoClean. In: SWAP 2007, Bari.