Ontology Naming Pattern Sauce for (Human and Computer) Gourmets

Vojtěch Svátek[1], Ondřej Šváb-Zamazal[1], and Valentina Presutti[2]

[1] Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{svatek|ondrej.zamazal}@vse.cz
[2] ISTC-CNR, Via Nomentana 56, 00161 Rome, Italy
presutti@cnr.it

Abstract. Various explicit and implicit naming conventions for entities have emerged in ontological engineering realms during the decade/s of its existence. In the paper we argue that the naming principles are neither trivial nor completely haphazard in practice, present a preliminary categorisation of ontology naming patterns, and discuss the impact of entity naming on both human and computer perception of ontologies.

1 Introduction

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 is one of first attempts to systematically populate this category of design patterns; there has recently been similar effort carried out in the narrower context of bioinformatics [6], and references to entity name content have been made in general literature on ontological modelling such as [3, 8].

Meaningful names are helpful for both *people* and *machines*. Undoubtedly, in particular from the point of view of machine 'consumers', the logical structure of an ontology is mandatory and unambiguous, while entity naming is dependent on subjective choices of designers, and is even optional in the sense that random strings can be used instead of names. Metaphorically, we could thus view the logic as 'meat' and naming as 'sauce'. Even if sauce is not a necessary part of every meal, it often helps *digest* the meat, and could in some cases be more *caloric* (to read: bear more real-world semantics) than the meat. 'Digesting' the logic is easy when entities are presented to the user in small chunks, such as in window-based interfaces of ontology editing environments. However, in this mode, only a small part of the whole knowledge structure can be viewed. On the other hand, various linear and diagrammatic notations allow to display larger clusters of entities but require the user to employ his/her intuition on the role of each entity in the structure. Then natural-language-like naming gains on importance. 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 readability at the level of *meta-model constructions*. Naming patterns could play an analogous role at the level of *model entities*. Naming can also increase the 'nourishing factor' (i.e. information value) of knowledge structures, just because of the same feature that precludes their unambiguous processing: while the inventory of logical constructs (and even logical design patterns) in a language such as OWL is necessarily restricted by the language norm, naming conventions and patterns can exploit any kind of structure that can be expressed within alphanumeric strings. While 'digesting' is only an issue for humans, 'additional calories' can be quite beneficial for software tools that analyse and process ontologies, such as ontology matchers over complex correspondences [5].

Let us rapidly demonstrate the 'digestive' and 'nourishing' potential of adequate naming on an OWL restriction in Manchester syntax:

```
StateOwned Director only (nomination some ministry)
```

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

```
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". We will refer to elements of this example later. The rest of the paper is structured as follows. Section 2 outlines our principles of categorising naming patterns. Section 3 then characterises different categories of patterns, including examples from existing ontologies.[3] We first discuss generic naming conventions, then focus on patterns specific for a particular entity type (classes, instances or properties), and finally on patterns spanning over multiple entities. Finally, Section 4 wraps up the paper and outlines directions for future research.

2 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; (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. 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).

[3] A more thorough description is in the long version of the paper, see `http://nb.vse.cz/~svatek/wop09long.pdf`.

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, we include both patterns that have been tentatively 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 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.

3 Detailed Descriptions of Naming Pattern Categories

3.1 Generic Principles and Conventions

Naming Vocabulary In view of comprehensibility to humans as well as NLP tools, the terms from which an entity name is constructed should be built from *human language vocabulary*; as mentioned in the introduction, the designer should not forget that the ontology will probably be used not only by purely formal reasoners but also by people and even NLP procedures that could leverage on meaningful naming. Furthermore, *abbreviations* (as also suggested in [3]) and *colloquialisms* should be avoided. *Acronyms* are often inevitable; however, the practice of using acronyms as prefixes of whole taxonomic trees, as artificial codes indicating the membership of the entity to this tree, is questionable, as it alienates the naming from the natural language.

The requirement of using human language naturally does not stipulate that only terms from common, generic vocabularies can appear in entity names. Specific domains may have their own terminology that is only familiar to a few dozens of experts and still could (or even should) be included in ontologies. Some of the terms may not exhibit typical features of words in human language; for example, names of genes in a gene ontology would consist of mixed alphabetic/numeric strings. Moreover, terms having a different generic meaning could be used in a specific domain ontology; for example the term 'Mouse' (as one of numerous metaphoric terms that are no longer viewed as colloquialisms) can be used in a domain ontology of computer equipment without the need for (unnatural) specifier such as 'ComputerMouse'. Care should however be taken when using terms so generic that they could interfere with entities in the *same* ontology (e.g. qualifier terms such as 'high' or 'light'); this problem is discussed in Section 3.3.

Case and Delimiter Conventions Such conventions exist even in programming environments. For OWL ontologies, a minimal requirement on capitalisation and delimiters seems to be to keep the same convention for all occurrences of one entity type; in addition, we would recommend, consistently with [3] (and following conventions used in description logics), to capitalise class names and decapitalise property names. As we saw in Example 1, this improves the readability of complex OWL restrictions, which often consist of sequences of alternating class and property names (aside modelling language keywords). For delimiters, OWL best practices do not encourage blanks in names, so underscore (This_Class), hyphen (This-Class) and 'camel case' (ThisClass) are all frequently used. In our opinion, however, underscore and 'camel case' are better alternatives, as the use of hyphen may interfere with compound words (in which the token before the hyphen often has a different role than if the same term were used in appositive), especially if the ontology is analysed by an automated NLP procedure that tries to properly tokenise each entity name.

3.2 Class Naming Patterns

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. We would strongly encourage *singular* for OWL ontologies: first, it is nowadays predominant in existing ontologies; second, some linear RDF notations such as N3[4] 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 merely 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.

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 'StateOwned' as subclass of 'Company' in our initial example. 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 ([3] 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 [8] or linguistic disjunctions mentioned below.

[4] `http://www.w3.org/2000/10/swap/Primer`

3.3 Instance Naming Patterns

If individuals are present in an OWL ontology, their names typically correspond to *standard vocabulary noun phrases*; examples are 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 in logical pattern [4]. 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 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[5] 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. 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'.

3.4 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. 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*

[5] `http://www.ninebynine.org/Software/HaskellRDF/RDF/Harp/test/wine.rdf`

preposition is occasionally used for spatio-temporal relationships. 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 3.7.

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.

3.5 Subclass and Instantiation Naming Patterns

It is quite common that a subclass has the *same head noun* as its parent class.[6] By an earlier study [7] 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. 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.[7]

Retrospectively, violation of head noun correspondence 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.

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. 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',[8] 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.

The *class-instance* relationship does not seem to follow generic naming patterns. An exception is the case of specified values discussed in Section 3.3.

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

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

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

3.6 Subproperty and Inverse Property Naming Patterns

We are not aware of a conspicuous naming pattern for the *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 should help link an object 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.

3.7 Naming Patterns over Restrictions

As we mentioned Section 3.4, 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 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:[9] '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[10] 'hasTopping' or maybe 'PizzaTopping'.

As possible reasons for the different 'psychologically 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.

In principle, we could also identify naming patterns over *local property restrictions*, for example in the form of 'lexical tautologies' such as `MushroomPizza`

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

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

`equivalentTo (Pizza and contains some Mushroom)`. This issue may deserve further study, although the frequency of such constructions is not very high.

4 Conclusions and Future Work

The intended contribution of the paper is a preliminary system of ontology naming patterns, which we illustrated on examples. 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, heuristic nature of any cues one could figure out. We are aware of this risk; the naming 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.

Most imminent future work will consist in *large-scale evaluation* of existing ontologies in terms of naming as well as bare plain logical patterns.[11] Within the empirical analysis stream, we should also study the usage of *other textual labels* rather 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 no.20/08 "Evaluation and matching ontologies via patterns".

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. 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`
4. 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/`.
5. Ritze D., Meilicke C., Šváb-Zamazal O., Stuckenschmidt H.: A pattern-based ontology matching approach for detecting complex correspondences. In: OM Workshop at ISWC'09.
6. Schober D. et al.: Survey-based naming conventions for use in OBO Foundry ontology development. *BMC Bioinformatics*, Vol.10, Issue 1, 2009.
7. Šváb-Zamazal O., Svátek V.: Analysing Ontological Structures through Name Pattern Tracking. In: EKAW-2008, Acitrezza, Italy, 2008.
8. Welty C.: Ontology Engineering with OntoClean. In: SWAP 2007, Bari.

[11] We plan to continuously update these results at `http://nb.vse.cz/~svabo/namingPatternsAnalysis/`.