Design Patterns for Semantic Web Ontologies: Motivation and Discussion

Vojtěch Svátek
Department of Information and Knowledge Engineering
University of Economics, Prague, W.Churchill Sq.4, 130 67 Praha 3, Czech Republic
svatek@vse.cz

Abstract

The relatively high level of standardisation of semantic web ontology languages is in contrast to mostly ad hoc designed content of ontologies themselves. An overview of existing methods supporting ontology content creation is presented. Methods based on design patterns are then discussed in more detail as they seem most promising particularly for business environment. Examples of elementary problems typical for semantic web ontologies are shown, and their pattern–based solution is outlined.

1. Introduction

The comprehensive proposal of the ontology language OWL, recently completed by the W3C *Web Ontology working group* (http://www.w3.org/2001/sw/WebOnt) is an important step towards ontology re/use on the semantic web. Its is a result of discussion within a large community of researchers, and thus might be considered as, in a sense, optimal. OWL benefits from support by the W3C consortium, which should make it the ontology language of first choice for semantic web developers. Shared *language* is however not a guarantee of shareable *content*. It is clear that (even partial) standardisation of the content of semantic web ontologies (SWOs) is by order of magnitude more difficult than standardisation of their language. Although the new W3C initiative on 'semantic web best practices' is only about to start [Schreiber 2003], there is no doubt about its primordial importance for the semantic web as a whole.

The following three quality criteria (although being, in some form, important for any sort of ontology) seem to be particularly critical for SWOs:

1. *Accuracy*: it should reflect the true state of affairs that holds in reality, with few or no tacit assumptions (which would lead to incorrect use beyond the original context).
2. *Transparency*: in order for a SWO to be shared, its meaning should be comprehensible for other people than just its designers.
3. *Reason–ability*, i.e. usability for (ideally, non–trivial) inference by existing semantic web reasoners.

Unfortunately, many existing SWOs fail in one, if not all aspects. By consequence, the usefulness of semantic web is likely to be questioned:

1. Inaccurate ontologies will produce wrong results as soon as their implicit assumptions are violated.
2. Opaque ontologies (be they accurate) will either not be used outside their native application or will be mapped on an inadequate state of affairs.
3. Ontologies unusable for inference, if prevalent, will question the choice of OWL (as inference–oriented language based on description logics) for the backbone of semantic web.

Clearly, the whole issue of quality cannot be reduced to the aforementioned three, but other criteria seem to be at least partially dependent on them: *consistency* should be more-or-less guaranteed by accuracy, *extendibility* is closely linked to comprehensibility etc.

Recent analysis [Tempich 2003] of the best–known SWO repository at http://www.daml.org (the well-known DAML repository) identified three clusters of ontologies in terms of proportions of constructs they contain. The clusters roughly correspond to three general types of ontologies distinguished e.g. by [van Heijst 1997]:

- *Terminological ontologies* contain many classes but few properties. They have typically been derived from linguistic thesauri or business taxonomies.
- *Information ontologies* contain many datatype properties. They have typically been derived from database (or object) schemata.
- *Knowledge ontologies* contain (relatively) many object properties and defined classes. In the semantic web context, they have often been contrived for reasoning based on description logics (DL) [Baader 2002].

The study [Tempich 2003] only covered syntactical analysis of ontologies, which could be determined automatically, with the goal of generating artificial prototype ontologies for benchmarking the performance of tools such as reasoners or editors. Its by–product

however was a (semantically interpreted) observation on how heterogeneous the writing habits of ontology designers are.

Most terminological and information ontologies had obviously been converted from other ('native') languages. However, their availability in DAML+OIL or OWL syntax (being clearly beneficial by itself) will probably inspire further development of models labelled as SWOs but not taking full advantage of OWL's inference–oriented features. More complex (knowledge) ontologies will be directly adopted and/or used for inspiration much less frequently, since their transparency for humans is significantly lower.

In this paper, we first briefly review a wide scope of approaches, methods and tools aiming at higher–quality content of (existing or newly built) ontologies (Section 2). Then we analyse in more depth the utility of pre–fabricated ('design') patterns for semantic–web ontology development 'in the small' (Section 3). We end up with conclusions and suggestions for future work.

2. Support for ontology content quality

In this section, we align several approaches that differ in their nature as well as in the underlying process (e.g. verification of existing ontologies vs. support for the design of new ones). Some of them could be viewed as orthogonal. We focus on their contribution to the fulfilment of three criteria identified in Section 1.

2.1 Systematic methodologies

Ontology development *methodologies*, most of which have been summarised in [Lopez 2002a], are (similarly to software engineering methodologies) mainly high–level, i.e. independent of a particular language, and concentrate on development processes rather than on the final artefact. They mainly support ontology design *in the large* rather than *in the small*. They indirectly contribute to ontology *accuracy* and *transparency*, but (due to language independence) not that much to *reason–ability*. Their integration with other standardised processes in a business environment may represent significant overhead.

2.2 Upper–level ontologies

Aligning existing or new domain ontologies with carefully–designed *upper–level ontologies* such as SUMO (http://ontology.teknowledge.com) or DOLCE [Gangemi 2002] has been proposed as means for quality improvement. Matching domain concepts and relations to upper–level ones can help the ontology designers realise their true nature, which has positive impact on *accuracy*. Furthermore, mapping on standard models naturally leads to *transparency*. However, it may not be easy for casual ontology designers (especially in business environment) to capture the rationale of complex, philosophically–flavoured models, only a small portion of which appears directly useful for the application.

2.3 Meta–properties

An alternative to aligning, particularly suitable for verification of existing ontologies, is the method suggested in the *OntoClean* project [Guarino 2002]. First, properties (the term 'property' is used in abstract sense in *OntoClean*, i.e. for unary predicates that basically amount to classes!) in an ontology are labelled with meta–properties such as 'rigidity', 'identity' or 'untity'. Predefined constraints on meta–property values are then tested. For example, an anti–rigid property (which does not necessarily hold for the given entity and thus can change over time) cannot hierarchically subsume a rigid property, e.g. 'student' cannot be superclass of 'person'. The value of meta–properties is in offering a formal framework for modelling choices that are done intuitively by experienced developers. They however, in the current form, only apply to taxonomic relations, and (despite existing tool support and integration with a prominent methodology [Lopez 2002b]) may again discourage casual users by the intricacy of underlying philosophical distinctions. Meta–properties significantly contribute to *accuracy*, while their impact on *transparency* is arguable (actually, there is little problem of transparency with hierarchies alone). In the context of SWO, they probably do not offer too much for non–trivial (DL–oriented) reasoning.

2.4 Language–specific user guides

For concrete SWO languages such as OWL, user guides have been designed [Smith 2003]. Their contribution to *accuracy* is in clarification of semantics of individual constructs. For example, their careful reading may prevent the user from mistaking cardinality restrictions for primitive value restrictions on numerical property (e.g. min–cardinality of 'age' for 'adult–person' being set to 18), or from the assumption that existential restriction is always contained in a universal restriction. (An experience of the author is that students in Information Systems, who were in fact more familiar with object models, frequently make both types of errors in the first, intuitive try of ontology design.) The guides however only map from commented ontology fragments to meaning, not vice versa. There is typically no hint on *which* construct and *how* to use it for a certain prototypical state of affairs. The impact on ontology *transparency* is thus relatively low. Finally, the liaison between the used constructs and *reason–ability* is not always obvious from the guides, since the latter typically

depends on a combination of constructs rather than on a single one.

2.5 Collected hints

A slightly different resource is language–independent ontology development guides, the most popular one probably being [Noy 2001]. It contains a 'light–weight' version of *methodology*, accompanied with a chapter containing useful *hints*. Unlike the previous case, the hints take as starting point a modelling decision, e.g. whether to model a certain dichotomy by two disjoint classes or by a binary datatype property. The mapping is therefore from abstract meaning to concrete (yet, partially language–independent) ontology constructions. Such hints may improve not only the *accuracy* but also mutual *transparency* among the developers who stick to them. It is also natural to include a *reason–ability* commentary with a hint.

2.6 Pre–fabricated patterns

A natural follow–up to unordered and informal collection of hints is to formulate a repository of pre–fabricated *patterns*. Coupling these two ways of support together seems to be the most natural (relatively light-weight) solution particularly for business ontology developers, since it mimics the approach used in information system development in general.

Broadly spoken, many research groups already addressed the issue of reusable patterns (for 'system development') in different ways and with different contexts. Let us first briefly characterise the usage of patterns in the *software engineering* community, where it is currently rather widespread. Then we will turn attention to analogous projects in *ontology engineering*.

2.6.1 Design patterns in software engineering

In software engineering, design patterns [Gamma 1995] are a well–known method of reuse, applicable on analysis and design models as well as on implemented code. The representation of a design pattern typically contains the following information:

- Problem description
- Suggested solution
- Implementation guidelines
- Discussion on consequences of using the pattern.

While the problem description and discussion on consequences are typically verbal, the suggested solution often has the form of UML model (class and interaction diagrams, see http://www.omg.org/uml) with abstract roles to be filled in with application–specific concepts. Finally, the implementation guidelines usually contain free–text recommendations related to the specifics of individual languages, as well as samples of source code with abstract roles to be replaced with application–specific names.

2.6.2 Design patterns in ontology engineering

One of the first approaches addressing ontology reuse (although using the term 'knowledge bases' rather than ontologies) was that of [Clark 2000]. They identified the misuse of inheritance (more generally, of the is–a relation) in knowledge reuse, and suggested instead a pattern–based approach backed by category theory. The Prolog implementation of patterns makes it quite general from the point of view of representation language.

Templates (i.e. general patterns) for writing *axioms* in frame–based formalisms (particularly, in Protégé) have later been proposed by [Hou 2002]. The starting point was a collection of Ontolingua [Gruber 2003] ontologies. A set of reusable patterns has been identified and translated to 'fill–in–the–blank' sentences presented to the user. Discovery of the relevant templates was further eased by meta–properties characterising the nature of constraint involved in the axiom.

A slightly different focus is that of *semantic patterns* proposed in [Staab 2001], which address the problem of reuse of elementary constructs (such as 'local range restriction') across different knowledge representation languages.

The notion of 'ontology design patterns' has also been used by [Reich 1999]. There, however, the meaning rather was (software engineering) 'patterns for ontology design': the patterns themselves have *procedural* nature (e.g. 'link dynamically two ontology nodes') and correspond to building blocks for software applications manipulating with *terminological* ontologies. The approach was applied in the bioinformatics domain.

Similarly to collected hints (being a more elaborate version thereof), ontology design patterns could improve *accuracy*, *transparency* as well as (significantly!) *reason–ability*.

2.7 Summary of the overview

Table 1 summarises (in a tentative way) the impact aforementioned methods of ontology content quality support are likely to have on accuracy, transparency and reason–ability of resulting ontologies. All of them have their pros and cons and can be used in a complementary fashion. In the rest of the paper, we however concentrate on the ontology *pattern* paradigm, which seems to fit best to the scenario of *business* ontology development, and discuss the peculiarities of its use in the SWO realms.

Table 1. Methods versus quality criteria

Approach	Acc.	Tran.	Reas.
Methodologies	*	*	
Upper–level ontologies	*	**	
Meta–properties	**		
Language–specific user guides	*		
Collected hints	*	*	*
Pre–fabricated patterns	*	**	**

3. Towards SWO design patterns

Let us formulate some hypotheses about the desired *outlook* of the patterns, and attempt to transfer the *quality criteria* on ontology content (Section 1) from 'live' ontologies to their start–up patterns.

3.1 Presumable features of SWO patterns

An important distinction between software engineering (for simplicity, say, object) models and (not only semantic web) ontologies is the fact the latter do not significantly change from the analysis through design as far as the implementation *phase* of system development. When the ontology structure is set up (possibly, in a graphical environment), it is already accompanied with unambiguous formal code, which is likely to be used, without significant change, in the final artefact (information system). Therefore, it probably makes little sense to distinguish between ontology patterns for different phases. From this follows that the basic form of a SWO pattern will already be a *language–dependent* one. Although the cross–language interoperability issue raised in [Staab 2001] is important, it will be marginal for the (presumably) large community of developers who commit to OWL as W3C recommendation.

On the other hand, ontology design patterns could obviously range from *domain–dependent* ones (such as botanical patterns mentioned in [Clark 2000]) to generic ones (similar to axiom patterns of [Hou 2002]).

Structurally, the *formal part* of a pattern would probably consist of an ontology fragment, including *directly reusable* elements (classes, properties etc.) as well as *demo–elements* that would be replaced by the user's own. The directly reusable elements should typically be borrowed from *upper–level ontologies*. As with any pattern–based approach, *textual explanation* will play an important role.

For more complex patterns, especially those involving local restrictions on properties (probably the 'hardest' feature in OWL etc.), 'fill–in–the–blank' sentences suggested by [Hou 2002] for axiom construction would make sense. Note however that in SWO languages the distinction between axioms and 'frames' is less sharp than in the inherently frame–based environment of Protégé.

3.2 Examples

As discussed in Section 3.1, SWO design patterns could range from basic to complex and from generic to domain–specific. We however assume that *complex* patterns will mostly arise by composition of simpler patterns. Furthermore, by analogy with software engineering, *domain–specific* patterns are likely to be derived from repeated (similar) fragments of real ontologies, as demonstrated in [Hou 2002]. Unfortunately, while Ontolingua has been in use for more than a decade, DL–based SWO language (namely, OIL) only appeared by 2000 and further evolved. The number of ontologies classified as 'DL–oriented' in the DAML repository is not really low according to the study by [Tempich 2003]: 33 out of 95 syntactically correct ontologies belong to this category. However, most of them are either trivial or do not pertain to anything that could be called 'domain' (of business). A majority of them was obviously designed as demos for ontology (or just DL) tools.

Given that, let us outline two modelling problems that are extremely simple and generic but at the same time peculiar to the class of DL–based languages. Both actually deal with *property names*, and should probably be ranged under this heading in a ontology pattern library. For each of them, we informally sketch some 'patterns'; clarifying examples are taken from a hypothetical real–estate ontology.

3.2.1 Use of self–standing properties

The concept of exclusively *binary* relations *independent* of any class is surprising for most newcomers to the SWO world, since in other languages restricted to binary relations (be they based on the frame or object paradigms), slots/attributes/associations are typically linked to classes. The use of the term 'property' makes this fact even more striking. Modellers unfamiliar with this feature might for example define properties such as 'has' and subsequently restrict its domain and range to a pair of classes, e.g. 'apartment' and 'owner'. This will usually work well from the point of view of *inference*; there is however no guarantee that the name of the property and the domain/range axioms will be displayed together in every *modelling* tool to which such an ontology will be imported in the future.

'Naming' patterns could promote the use of self–standing properties that are also to some extent self–explanatory. The most obvious alternative is to define a property 'owns' and its inverse 'owned-by'. However, in a particular context (e.g. if the ownership of an apartment were legally different from owning a movable object), it would be wise to name the property 'owns-apartment' (and the inverse 'apartment-owned-by') and immediately associate a *range* axiom with it. We could similarly fix

the *domain*, or both the domain and range, to a property. Although the existence of such modelling choice is obvious for an experienced modeller, it is probably worthwhile to explain it to a newcomer.

Interestingly, self–standing properties have recently been suggested as enrichment to UML, in connection with the ontology initiative of the Object Management Group [Baklawski 2001]. If this proposal is successful, it might extend the impact of property–oriented patterns from the ontology world even to the object world.

3.2.2 Transformation of n–ary relations

Another problem, shared by a larger classes of languages, stems from the restriction to binary properties, while the 'state–of–affairs' often naturally involves *n*–ary relations (with *n* typically being 3 or 4, scarcely more). For example, a buyer purchases an apartment from a seller using a certain payment method. In order to express this relation e.g. in OWL, we can use one of at least three alternatives:

1. To *reify the whole relation*. Instead of relation 'purchases' we obtain a class 'purchase', and the *roles* (in the sense of e.g. object modelling) would become properties. Interestingly, in such situations role–like properties can often be identified with or subsumed to general relations known from *linguistic* sentence analysis, such as 'actor', 'object' or 'instrument'. The properties relating the buyer and the seller, respectively, to the purchase, will be specialisation of a general 'actor' property, the property relating the apartment to the purchase will be (or specialisation of) 'object', and the property relating the payment method to the purchase will be (or specialisation of) 'instrument'. Examples of this sort abound especially in the conceptual graph [Sowa 2000] community, which is historically biased towards the linguistic view and has to deal with the same 'binarity' constraint as the SWO developers.
2. To *iteratively reify* sub–components of the relation. For example, we can introduce a property 'purchase-connection' linking the buyer and the seller alone, then a property 'object-of-purchase-connection' linking the purchase-connection and the object (apartment), and finally a property 'object-purchased-using' gluing the payment method to the construction built so far.
3. To *dispose* of some components of the relation. For example, if the ontology is to be always (re)used within a single real–estate agency (and not within an open marketplace) and should only deal with physical ownership (not with financial aspects), we can end up with a binary property ('sale') relating apartments to their buyers.

Although the first alternative could be viewed as 'best practice' for SWOs in general, the remaining two options may become worth considering in some contexts.

4. Conclusions

We surveyed the inventory of content–oriented support methods and information resources available for ontology developers, and compared their roles with respect to three quality criteria crucial for semantic web ontologies (SWOs) – accuracy, transparency and reason–ability. Special attention was paid to approaches based on pre–fabricated patterns, since they represent one of few possibilities to map from modelling problems (topical for the ontology designer) to immediately reusable ontology constructs. In the second part of the paper, we hypothesised about the possible outlook of future SWO design patterns, and discussed two simple modelling problems that could reasonably be addressed by generic patterns. Such simple patterns could be viewed as possible building blocks for libraries of design patterns (analogous to those from the software engineering domain), which would be truly beneficial for SWO developers, in particularly in business environment. The new W3C initiative on 'semantic web best practices' will for certain involve systematic effort in this direction.

5. Acknowledgements

The author is thankful to Frank van Harmelen for his comments on a draft of this paper, and to C. Tempich and R. Volz, who made publicly available the detailed documentation on their analysis of the DAML ontology repository. Comments by anonymous reviewers are also acknowledged.

The research is partially supported by grant no. 201/03/1318 of the Grant Agency of the Czech Republic.

6. References

[Baader 2002] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation and Application*, Cambridge University Press, 2002.

[Baclawski 2001] K. Baclawski, M.K.Kokar, P.A. Kogut, L. Hart, J. Smith, W.S. Holmes, J. Letkowski, M.L. Aronson, "Extending UML to Support Ontology Engineering for the Semantic Web". In: Fourth International Conference on UML, Toronto (2001).

[Clark 2000] P. Clark, J. Thompson, B. Porter, "Knowledge Patterns". In KR'2000 (Proc 7th Int Conf), pages 591-600, Eds: A. Cohn, F. Giunchiglia, B. Selman, CA:Kaufmann, 2000.

[Gamma 1995] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley, Boston MA (1995)

[Gangemi 2002] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, "Sweetening ontologies with DOLCE". In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), volume 2473 of Lecture Notes in Computer Science, page 166 ff,, Siguenza, Spain, Oct. 1-4, 2002.

[Gruber 2003] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, 5(2), 199-220, 1993.

[Guarino 2002] N. Guarino, C. Welty. "Evaluating ontological decisions with OntoClean", *Communications of the ACM*, 2(45):61--65, 2002.

[Hou 2002] Chih-Sheng Johnson Hou, N. F. Noy, M. A. Musen, "A Template-Based Approach Toward Acquisition of Logical Sentences". Intelligent Information Processing 2002, World Computer Congress, Montreal, Canada. 2002.

[Lopez 2002a] M. Fernández-López, A. Gómez-Pérez, "Overview and Analysis of methodologies for building ontologies", *Knowledge Engineering Review* (KER). Vol. 17[2]. 2002. 129-156.

[Lopez 2002b] M. Fernández-López, A. Gómez-Pérez, "The Integration of OntoClean in WebODE", In: Evaluation of Ontology-based Tools, Proceedings of the EKAW02 Workshop on Evaluation of Ontology-based Tools, 2002.

[Noy 2001] N.F. Noy, D.L. McGuiness, *Ontology Development 101: A Guide to Creating Your First Ontology*. Available from http://protege.stanford.edu/publications/ontology_development/ontology101.html.

[Reich 1999] J. Reich, "Ontological design patterns for the integration of molecular biological information". In: Proceedings of the German Conference on Bioinformatics GCB'99, pages p.156--166, Hannover, Germany.

[Schreiber 2003] G. Schreiber, *DRAFT: Semantic-Web Best Practices (SWBP) Working Group Charter*. http://www.cs.vu.nl/~guus/public/bp-charter.html

[Smith 2003] M. Smith, C. Welty, D.L. McGuinness, *OWL Web Ontology Language Guide*, W3C Candidate Recommendation, http://www.w3.org/TR/2003/CR-owl-guide-20030818/

[Sowa 2000] J.F. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[Staab 2001] S. Staab, M. Erdmann, A. Maedche, "Engineering ontologies using semantic patterns". In: A. Preece, editor, Proc. of the IJCAI-01 Workshop on E-Business & the Intelligent Web, 2001.

[Tempich 2003] C. Tempich, R. Volz, "Towards a benchmark for Semantic Web reasoners - an analysis of the DAML ontology library". In: Proceedings of Evaluation of Ontology-based Tools (EON2003) at 2nd International Semantic Web Conference (ISWC 2003).

[van Heijst 1997] G. van Heijst, G. Schreiber, B. Wielinga, "Using Explicit Ontologies in KBS development", *International Journal of Human-Computer Studies*, Volume 46, 1997, pp. 183-292.