Helping Naive Users to Reuse Ontology Design Patterns

Elena Montiel-Ponsoda, Guadalupe Aguado de Cea,
Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa

Ontology Engineering Group
Universidad Politécnica de Madrid, Facultad de Informática, Dpto. de Inteligencia Artificial
Campus de Montegancedo s/n, 28660-Boadilla del Monte, Madrid
emontiel@delicias.dia.fi.upm.es { lupe, asun, mcsuarez}@fi.upm.es

Abstract. The decisive launching of the Semantic Web depends on two key factors: the design of sound methodologies for guiding users in the reuse of available knowledge resources that speed up the ontology development, and the suitability of those methodologies for an average user. In this paper we propose a method for the reuse of ontology design patterns aimed at users with little expertise in ontology development, i.e. naive users. The method workflow is explained in the light of some examples of reuse of logical ontology design patterns: the *SubClassOf* relation, *Exhaustive Classes* and *Disjoint Classes* patterns.

Keywords: ontology design patterns reuse, ontology design patterns reuse method, reuse method for naive users

1 Introduction

Reuse has proven to be highly beneficial in many areas, basically from the financial viewpoint, but recently also from the environmental one (see RREUSE[1]). Besides the original reuse of physical resources (from milk or beer bottles to aircraft hulls), reuse of information has dramatically increased in the last decades because of the enormous benefits it brings. Reuse of code or technical components by software companies, or reuse of translated documents by international organizations are just some examples of areas that have benefit from the amount of time and costs that this activity reduces. The motto of its supporters has been chanted in many forums: do not reinvent the wheel. In the Ontology Engineering field this activity has also shown its importance, and it is believed to be the necessary lever for a successful launching of the Semantic Web. Reuse of existent knowledge resources would obviously accelerate the process of ontology development against creation from scratch. However, and depending on the use case needs, solutions do not always rely on the reuse of entire ontologies, but some parts of them. And in some cases, the ontology development would be speeded up by reusing non-ontological resources. In the NeOn project[2], the creation of

[1] http://rreuse.org/t3/
[2] http://www.neon-project.org

methods for guiding the reuse of knowledge resources has a major priority. In deliverable D5.4.1 of this project [11] knowledge resources have been divided into ontological and non-ontological resources. Ontological resources comprise ontologies, ontology modules, ontology statements and ontology design patterns. Non-ontological resources include free texts, textual corpora, web pages, standards, catalogues, web directories, classifications, thesauri, lexicons and folksonomies among others. According to the specific needs in each case, reuse of knowledge resources will be centered in certain ontological and/or non-ontological resources.

In this paper we focus on the reuse of Ontology Design Patterns (henceforth ODPs) as ontological resources in the development of ontologies, and we propose a method for guiding naive users in this activity. ODPs are considered *ontology modeling solutions that after being recurrently used for solving similar design problems can be identified as generalized design solutions for certain ontology modeling issues* (inspired in [5,8]). The success of the ODPs reuse method will depend on its suitability to the user. Thus, instead of assuming expertise on the reuse of design patterns in general, our research focuses on users with little expertise in the development of ontologies.

The reminder of this paper is structured as follows. In Section 2, we include a brief overview of existent methods for design patterns reuse in general, making special emphasis on the claimed limitations of design patterns reuse in the neighboring field of Software Engineering. This overview will serve as motivation for proposing the research objectives in Section 3. A novel method for the reuse of ODPs by naive users will be presented in Section 4. There, we describe the main tasks of the method that mainly relies on Natural Language (NL) techniques for performing the reuse activity, as well as a novel tool that supports the method. Finally, in Section 5, the method workflow is presented in the light of some examples of reuse of Logical ODPs, as they have been identified in D5.1.1 [10] of the NeOn project.

2 State of the Art in Design Patterns Reuse

Design patterns have a long tradition in Software Engineering. With the publication of some of the most important design pattern catalogues [1,4] in the mid 1990s, they reached their height, and the practice of accessing catalogues for reusing design patterns in object-oriented design became a common practice. Patterns were described following certain templates whose sections were decided upon by the authors of the respective catalogues. Such templates included not only graphical representations of the pattern, but also explanations of decisions, motivations, examples or even implementation hints, amongst others. Nowadays, such patterns are integrated in software tools for enabling a quicker access and integration.

However true that might be, object-oriented design pattern catalogues or repositories presuppose prior design knowledge and expertise on the part of the user. This fact and other limitations of the reuse of patterns are being discussed in public

forums by some experts in the Software Engineering domain (see *The Software Patterns Blog*[3]). The main limitations are related to:

1. Lack of general methods or standards for the reuse of the different pattern repositories, since some efforts in that sense are limited to recommendations for local use developed by the authors of the manuals themselves.
2. Differences in the styles followed by templates depending on the manual, in that some of the steps or approaches given by certain authors cannot be extrapolated or reused in searching other design pattern repositories.
3. Efforts demanded by the search activity, which, apart from being time consuming, requires a careful analysis of the patterns on the part of the user.

Regarding the reuse of design patterns in Ontology Engineering, this practice is not so widespread because of two obvious reasons: the early stage in the ODPs research, and the almost inexistence of ODPs repositories. In fact, it is not until the beginning of this century that design patterns are fully introduced in the Ontology Engineering domain by experts in the area as Gangemi and colleagues [5], Rector and Rogers [9], Svatek [12] or the W3C Ontology Engineering and Patterns Task Force[4]. Currently, we find some ODPs on-line repositories, as the one focused on the Biological domain[5], or the preliminary repository of OWL-based Content ODPs[6] at the Laboratory for Applied Ontology wiki page. The latter repository is being extended and enhanced within the NeOn project and will be available at the Ontology Design Patterns.org wiki page[7].

Even so, as in the case of object-oriented design patterns, there exist no methods *per se* for guiding users and facilitating the reuse of ODPs. It is as well assumed that users have some expertise in the reuse of patterns and know that they have to access design pattern repositories and search for the appropriate pattern. There have been, however, some initiatives for helping users in the process of adapting or implementing ODPs by means of wizards, as for example, the ones provided by the CO-ODE project[8] for the Protégé ontology editor[9], as reported in [3]. Finally, we should refer to the inclusion of some guidelines for the *adaptation* of ODPs to real use cases in [8]. In principle, those guidelines are supposed to refer to the manual use of Content ODPs. Authors consider that the adaptation or matching possibilities of Content ODPs to use cases are: *precise* matching, *broader* matching, *narrower* matching, *partial* matching and *redundant* matching. These guidelines will not be considered at this stage because they are aimed at expert users.

[3] http://pattern.ijop.org/
[4] http://www.w3.org/2001/sw/BestPractices/OEP/
[5] http://odps.sourceforge.net/odp/html/index.html
[6] http://wiki.loa-cnr.it/index.php/LoaWiki:CPRepository
[7] http://www.ontologydesignpatterns.org
[8] http://www.co-ode.org/downloads/wizard/
[9] http://protege.stanford.edu/

3 Approaching Ontology Design Pattern Reuse Shortcomings

The main purpose of this research is to define methods for the reuse of ODPs aimed at naive users. Therefore, taking into account the state of the art to this regard, we consider that some effort has to be put into the following actions:

- Creation of standardized templates for the description of ODPs understandable to different types of users
- Creation of functional and generalized methods or guidelines for users with different level of expertise in the ODPs reuse
- Creation of techniques and tools for supporting a semi-automatic or automatic pattern selection in a multilingual environment

The first action pointed out above has been dealt within the NeOn project in various deliverables [8, 10]. Our research thus is centered in the development of methods for the reuse of ODPs, as well as in techniques and tools for supporting those methods.
The first efforts have gone to the creation of a method for the reuse of ODPs aimed at naive users, since we consider that bringing ontologies closer to the average user is a crucial issue for the success of the Semantic Web. Furthermore, we rely on techniques and tools that evolve in parallel to users in the sense that as users gain expertise in the reuse of ODPs, they could skip some steps in the method. Therefore, we can state that our proposal has a didactic nature, as will be outlined in section 5. Last but not least, and taking into account the growing importance of multilinguality for international organizations committed to the introduction of ontologies in their information systems, we have foreseen techniques and tools for users working in a multilingual environment. The first set of languages we have considered are English, Spanish, and German.

4 Novel Method for the Reuse of Ontology Design Patterns by Naive Users: Proposed Guidelines

The takeoff of the Semantic Web relies on extending the use of ontologies among a wider community of users, especially novice users. For this purpose, methods intended for users with little expertise in the development of ontologies, and, in its turn, in the reuse of ODPs, have to be created and supported by user-friendly tools. This implies in most cases the use of Natural Language (NL) techniques. As far as the ODPs reuse activity is concerned, this may rely on the inclusion of NL components that bridge the gap between naive users and ODPs. In this sense, we propose a novel method for the reuse of ODPs that has as a starting point a precise definition in NL of the phenomenon or domain aspect the user wants to model in the ontology, and as a target one, the obtainment of the most suitable ODP. This method can be divided in three main tasks (as illustrated in Figure 1):

1) Task 1. ODPs *Formulation*. The goal of this task is the formulation in full NL of the domain aspect to be modeled: the user has difficulties in modeling a certain domain parcel and expresses that knowledge in NL.

2) Task 2. ODPs *Refinement*. The goal of this task is refining the input from Task 1 by means of the user answering questions. This task is only carried out when the matching between the input and the ODP needs refinement because of ontology enrichment needs or lexical ambiguities. (See section 5 for more details).

3) Task 3. ODPs *Validation*. The goal of this task is to confirm that the resulting ODP meets the user expectations.

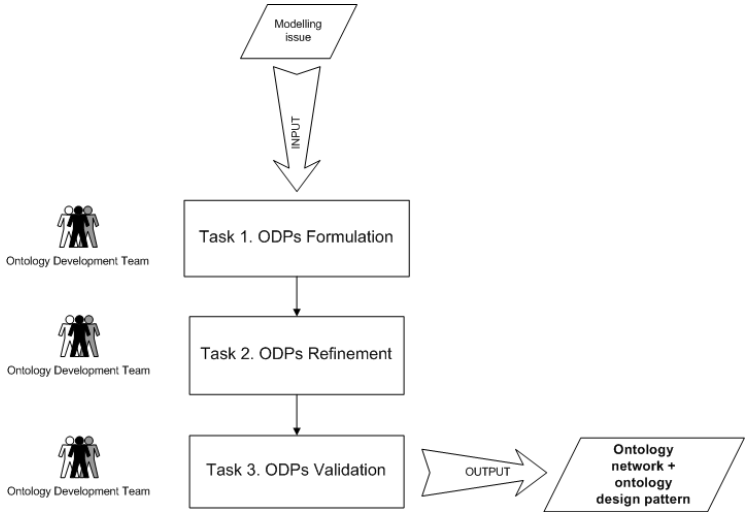

Figure 1. Method for reusing ODPs aimed at naive users

For better understanding, we explain the method in the light of some real examples in section 5. In order to support it, we propose the development of a tool for enabling a semi-automatic selection and integration of ODPs, mainly intended for users with little expertise, but also recommendable for expert users in order to speed up the process of ODP selection. This tool relies on the application of NL techniques for performing the semi-automatic selection. In sections 4.1 and 4.2, we include general information about the techniques and tools that support this novel method.

4.1 Techniques for Supporting ODPs Reuse: Enrichment of ODP templates with NL

In order to support the proposed method for the reuse of ODPs by naive users, NL techniques have been employed for enabling a semi-automatic selection of the ODP that better matches the expression in NL formulated by the user. The first action in this sense has been the enrichment of the templates proposed in D5.1.1 [10] of the NeOn project for describing ODPs with a new slot containing *Lexico-Syntactic patterns*. Lexico-Syntactic Patterns (LSPs henceforth), defined here as *formalized linguistic schemas or constructions derived from expressions in NL that consist of certain linguistic and paralinguistic elements following a specific syntactic order, and*

that permit to extract some conclusions about the meaning they express (definition inspired in [6] and [7]), will be the key element for providing the matching between NL formulations and ODPs.

Lexico-Syntactic Patterns were first introduced in this field by Hearst [6] in the early 1990s. The goal of her research was the automatic acquisition of lexical syntax and semantics from machine readable dictionaries. Hearst said that LSPs were constructions that *occurred frequently and in many text genres, almost always indicated a relation of interest, and were recognized with little or no pre-encoded knowledge*. Since then, there have been many authors that have applied LSPs for the automatic discovery of lexical items related semantically from unstructured texts. Particularly in Ontology Engineering, Hearst LSPs have been used in numerous studies mainly with the objective of performing ontology learning and population from text (see [2]).

In the case of this specific research, we decided to use LSPs for a different purpose. Our aim was to (a) identify and gather all possible expressions in NL that are equivalent to the conceptual schema captured by ODPs identified in [10], (b) make an abstraction of the expressions and (c) formalize them in LSPs. What we obtained in the end was a set of LSPs corresponding to each ODP. In the initial stage of this research, we have identified a preliminary repository of LSPs from NL expressions in English that match the following Logical ODPs: SubClassOf Relation, Multiple Inheritance between Classes, Equivalence Relation between Classes, Object Property, SubPropertyOf Relation, Datatype Property, Existential Restriction, Universal Restriction, UnionOf Relation, Disjoint Classes, Exhaustive Classes, and N-ary Relation. LSPs are language dependant, so we need to follow the same procedure for the rest of languages. (This repository is to be found in NeOn D2.5.1 [8]).

To illustrate this procedure, we have included Table 1 with some examples of sentences in NL expressing a hyperonymy-hyponymy relation that match two different LSPs corresponding to the *SubClassOf* relation (identified in NeOn ODPs repository as LP-SC-01, see [10]). Symbols and restricted words appearing in the selected LSPs have been added below.

Table 1. Examples of NL sentences matching two LSPs for *SubClassOf*

Examples in NL	**LSPs**
▪ *Vertebrates **are** animals.* ▪ *An orphan drug **is** a type of drug.*	NP<subclass> be [CN] NP<superclass>
▪ *Animals **are divided into** two major categories: vertebrates and invertebrates.* ▪ *Vertebrates **include**: mammals, amphibians, reptiles, birds and fish.*	NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>,)* and NP <subclass>
NP= Noun Phrase + semantic role between < > CATV = Verbs of Classification. This group includes verbs as: classify in/into, comprise in, contain in CD = Cardinal Number CN= Class Name. This group includes: class of, group of, type of, member of, etc. PARA = Paralinguistic symbol ( ) = groups two or more elements * = repetition	

[ ] = Optional Elements

4.2 S.O.S.: a System for Ontology design pattern Support

The second action for supporting the proposed method was the creation of a tool that enabled the matching between ODPs and LSPs relying on NL processing tools. This tool has been called *S.O.S., System for Ontology design pattern Support*, and aims at serving as an interface between naive users and ODPs. This tool is under development and will be integrated as a plug-in in the NeOn toolkit[10], and used in combination with its ontology editor. The system workflow can be divided in 4 main steps:

1st step. The user introduces in the system a sentence in NL describing the parcel of knowledge (s)he wants to model. For the first prototype of the S.O.S. tool, English, Spanish and German are the languages foreseen to be supported.

2nd step. The system annotates the sentence with NL processing tools and compares the results of the annotation against the set of LSPs, associated in its turn to ODPs.

3rd step. If the exact matching is found, the system selects the ODP in question. If there is no exact matching or if the exact matching needs refinement because of ontology enrichment needs or lexical ambiguities, the system interacts with the user for refining the input and searching again. The details of this step are explained in section 5.

4th step. Once the matching has been performed, the selected ODP is instantiated or extended with information extracted from the sentence introduced by the user, and it is returned to the user in the form of an integrated ODP with the rest of the ontology being developed.

5 Modeling *SubClassOf* Relations according to proposed Guidelines

In this section we describe in detail the performance of the tasks that make up the proposed guidelines for naive user by making use of S.O.S.

Task 1. ODPs *Formulation*. Users select the language in which they want to model (English, Spanish and German), and introduce a sentence in NL. For this task, they have to exactly and explicitly define the aspect of domain knowledge they want to model. In order to assist users in this task, S.O.S. offers some basic recommendations, as for example:

- Express what you want to model in a declarative way. (E.g.: Amphibians are vertebrates).
- Use exclusively those words that are strictly needed for expressing what you expect to model.

[10] http://www.neon-toolkit.org/

Let us imagine that the user introduced the following sentence expressing a *hyperonym-hyponym* relation: *Vertebrates are classified into different groups: mammals, amphibians, reptiles and birds.*

Task 2. ODPs *Refinement*. Using the NL sentence introduced by the user, the S.O.S. tool has to find the corresponding LSP to the sentence. However, this action is not trivial, even if the result of the matching between the annotated NL sentence and the LSP associate to the ODP is *exact matching* (1 annotated sentence – 1 ODP).The reason for this is that from an ontological perspective it is recommendable to enrich some ontology relations with additional knowledge to ensure a richer conceptualization, and avoid eventual errors and inconsistencies in future ontology-based applications.

Picking up the sentence introduced in Task 1 *Vertebrates are classified into different groups: mammals, amphibians, reptiles and birds*, we will exemplify this case. A sentence like this would unequivocally be matched to the identified LSP for *SubClassOf* relation shown in Table 1 (LP-SC-01 ODP in NeOn ODPs repository):

NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>,)* and NP <subclass>

Correct so far. However, in the design of ontologies this relation could be further specialized with knowledge about disjointness and/or exhaustiveness.

In the case of disjointness and exhaustivness, we have identified some words in language that having a certain position in the sentence can make that sort of knowledge explicit. This assumption needs further evidence and research is currently being conducted. Some examples of words that signalize that sort of knowledge are included in the examples below:

- Vertebrates are classified into mammals, amphibians, reptiles, birds *or* fish.
- Vertebrates are *only* classified into mammals, amphibians, reptiles, birds and fish.

The conjunction *or* in the first sentence is a sign of disjointness. In the second sentence, the adverb *only* indicates that there are no more classes into which vertebrates can be classified into. However, when there is no sign for specifying this kind of knowledge (as in our example), the system needs to make this information explicit, and one way of doing it is by asking the user. Therefore, the S.O.S. will launch a set of questions to the user for finding out if the sentence taken as example could be further specified in disjoint and/or exhaustive.

Regarding disjointness, the question could be:

- *Can a certain vertebrate belong to the group of mammals, amphibians, reptiles and birds at the same time?*

The answer should be no, and the system would further model those subclasses as *Disjoint Classes*, identified by LP-Di-01 ODP in NeOn ODPs repository. Then, it would go on to ask about exhaustiveness. In this case, the question could be:

- *Are there any other types of vertebrates?*

If the answer is yes, the system would offer the user the possibility of introducing the missing subclasses in the input window. In this example, the type *fish* is missing to reach an exhaustive enumeration of *vertebrates*. The user would be made aware of this, and would introduce the new subclass. Then, the system would proceed to model those classes according to the *Exhaustive Classes* relation identified by LP-EC-01 ODP in NeOn ODPs repository. This kind of dependencies between ODPs are planned to be represented by state diagrams as in Figure 2.

Task 3. ODPs *Validation*. The system returns the user a UML diagram modelling the *SubClassOf* and the *Exhaustive Classes* relations fulfilled with information from the NL sentence. This diagram is accompanied by an explanation in NL of the model to instruct the user in the modelling of ontologies. In this way, the user has a new opportunity to check if the returned UML diagram complies with his or her expectations. If (s)he finally accepts the output, it is then integrated into the ontology being developed.

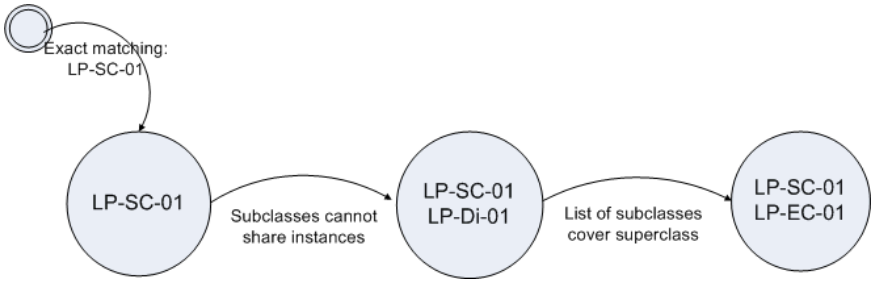

Figure 2. Dependencies between *SubClassOf* relation, *Disjoint Classes* and *Exhaustive Classes* ODPs[11]

6 Conclusions

This paper has concentrated on the reuse of Ontology Design Patterns (ODPs) and the impending need for methods or guidelines aimed at users with different levels of expertise in the ontology development and patterns reuse. We have presented a method and a set of guidelines for enabling naive users reuse of ODPs that mainly rely on Natural Language techniques. In order to support this method, *Lexico-syntactic patterns* that correspond to some Logical ODPs included in NeOn ODPs repository have been identified and formalized in order to constitute the basis of the S.O.S. tool (*System for Ontology design pattern Support*). This tool is in charge of matching expressions in NL about domain knowledge to the corresponding ODPs. The S.O.S. workflow also foresees user's feedback for a semi-automatic selection of the appropriate pattern. We have exemplified the method by means of a sentence in English expressing the *SubClassOf* relation. This has also allowed us to show how to tackle with ontology enrichment needs. In future work we aim at enriching the repository of LSPs with LSPs for more complex patterns.

[11] It is important to note that the Exhaustive Classes relation identified in [10] implies *the union of a set of mutually disjoint subclasses*. Thus, as can be seen in Figure 2, if classes expressed by the user are disjoint and exhaustive, this will be represented by the patterns: *SubClassOf* relation and *Exhaustive Classes* relation. However, there is no pattern in the current NeOn ODPs repository for expressing that classes are exhaustive but not disjoint. In this case, just the *SubClassOf* relation could be matched, and the information about exhaustiveness would be lost.

Acknowledgments. Research for this paper has been supported by the project *Lifecycle support for networked ontologies (NeOn)* (FP6-027595). In addition, it is partially co-funded by an I+D grant from the *Universidad Politécnica de Madrid*. We would also like to thank Inmaculada Álvarez de Mon y Rego for interesting feedback.

References

1. Bushmann, F., Meunier, R., Rohnert, H., Sommerland, P., and Stal, M: Pattern-oriented software architecture. A system of patterns. John Wiley & Sons, Chichester (1996)
2. Cimiano, P.: Ontology Learning and Population from Text. Algorithms, Evaluation and Applications. Springer (2006)
3. Egaña Aranguren, M., Stevens, R., and Antezana, E.: Ontology Design Patterns (ODPs) for bio-ontologies. Talk in Bio-ontologies SIG at ISMB, Vienna (2007)
4. Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley: New York (1995)
5. Gangemi, A., Catenacci, C., and Battaglia, M.: Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In D. M. Pisanelli (ed.), Ontologies in Medicine. IOS Press, Amsterdam (2004)
6. Hearst, M. A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In Proceedings of 14th Inter.Conference on Computational Linguistics, pp. 539-545 (1992)
7. Meyer, I.: Extracting knowledge-rich contexts for terminography. A conceptual and methodological framework. In D. Borigault, Ch. Jacquemin and M.C. L'Homme (eds.) In Recent Advances in Computational Terminology, 14, pp.279–302. John Benjamins (2001)
8. Presutti, V., Gangemi, A., David, S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: NeOn D2.5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NeOn project (2008)
9. Rector, A. and Rogers, J.: Patterns, properties and minimizing commitment: Reconstruction of the GALEN upper ontology in owl. In Proceedings of the EKAW04 Workshop on Core Ontologies in Ontology Engineering. CEUR (2004)
10. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V. and Sabou, M.: NeOn D5.1.1. NeOn Modelling Components. NeOn Project (2007)
11. Suárez-Figueroa, M.C., Dellschaft, K., Montiel-Ponsoda, E., Villazon-Terrazas, B., Yufei, Z., Aguado de Cea, G., García, A., Fernández-López, M., Gómez-Pérez, A., Espinoza, M., Sabou, M.: NeOn D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks. NeOn project (2008)
12. Svatek, V.: Design patterns for semantic web ontologies: Motivation and discussion. In Proceedings of the 7th Conference on Bussiness Information Systems (2004)