Approaches to Ontology Development by Non Ontology Experts

Guadalupe Aguado de Cea, Elena Montiel-Ponsoda, Mari Carmen Suárez-Figueroa
Ontology Engineering Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid,
Campus de Montegancedo s/n, 28660 Boadilla del Monte, Madrid Spain
{lupe, emontiel, mcsuarez}@fi.upm.es

Abstract

Untrained users in the development of ontologies are challenged by the formal representation languages that underlie the most common ontology editing tools. To reduce that barrier, many efforts have gone in the creation of Controlled Languages (CL) translatable into ontology structures. However, CLs fall short of addressing a more profound problem: the selection of the most appropriate ontology modelling component for a certain modelling problem, regardless of the underlying representation paradigm. With the aim of approaching non ontology expert's difficulties in selecting the most appropriate modelling solution, we propose a Natural Language (NL) guided approach based on a repository of Lexico-Syntactic Patterns associated to consensual modelling solutions, i.e., Ontology Design Patterns. By relying on this repository, untrained users can formulate in NL what they want to model in the ontology, and obtain the corresponding design pattern for the modelling issue.

1. Introduction

In the development of the Semantic Web, ontologies are the knowledge organization systems that have shown most suitable for providing meaning or semantics to the Web. The most quoted definition of ontology in the Artificial Intelligence literature states that an ontology is "an explicit specification of a conceptualization" [12]. To put it in simple words, we can define ontologies as consensual machine-readable models that represent a certain parcel of knowledge by a) identifying the set of concepts that describe that knowledge, b) making explicit the main properties and restrictions of those concepts, and c) establishing relations among them.

In the last years, institutions and enterprises worldwide have expressed their interest in organizing the great amounts of information they manage in ontologies. The main reason for this is that ontologies have proven to represent powerful and sound means of structuring information allowing machines to carry out complex reasoning operations with the information. However, constructing an ontology requires to follow a certain methodology[1], to have a good command of ontology editing tools, and to be proficient in formal representation languages. Because of this, the construction of ontologies has been mainly limited to projects in which domain experts cooperate with ontology engineers in the development of ontologies. The main obstacle for domain experts in building ontologies on their own has been seen in relation with their understanding of the representation paradigms used to encode ontology models.

Nowadays, one of the most followed paradigms for the creation of ontologies is Description Logics (DL) [2], on which the well-known Web Ontology Language (OWL)[2] relies. DL refers to a set of knowledge representation languages based on first-order predicate logics whose understanding demands good background in Logics. Most ontology editors (Protégé[3], TopBraid Composer[4], NeOnToolkit[5], etc.) support DL and are considered quite inaccessible to all but ontology modelling experts [8,9,14,15].

With the aim of overcoming the obstacles imposed by ontology languages, many efforts in the Ontology Engineering community have been directed to the creation of simplified syntaxes including elements of Natural Languages (NL) that try to disguise Logics. In this sense, research has been devoted to the creation of

[1] For a review on ontology development methodologies see [11]
[2] http://www.w3.org/TR/owl-features/
[3] http://protege.stanford.edu/
[4] http://www.topquadrant.com/
[5] http://www.neon-toolkit.org/

Controlled Languages (CLs) to make ontology languages more readable and understandable to non ontology experts. A subset of these CLs will be handled in section 2. There, we will also discuss the suitability and main drawbacks of a CL approach for the development of ontologies. After that, we will comment on the dichotomy CL vs. NL in section 3. Then, our main goal is to present our approach for the development of ontologies aimed also at non ontology experts that introduces two main novelties: 1) it allows users the formulation of sentences in a NL guided approach conveying what they want to model in the ontology, and 2) it enables the reuse of consensual ontology modelling components guaranteeing the modelling correctness of the resulting ontology. This approach is based on a repository of Lexico-Syntactic Patterns and its correspondence to Ontology Design Patterns, described in section 4. Section 5 will exemplify how we expect the proposed NL guided approach to work. We will also comment on some of the difficulties imposed by the use of unrestricted NL and some of the envisioned strategies to overcome limitations. Finally, we will conclude the paper in section 6.

2. Controlled languages for ontology development

Controlled Languages (CLs) in the field of Knowledge Engineering have been seen as attractive ways of making formal representations accessible and understandable to domain experts. Since the early 90s, many efforts have gone in the design of CLs for the development of knowledge-based systems [18]. As in other domains in which CLs have been widely applied (machine translation, generation of technical documents, etc.), these are understood as "subsets of natural languages whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity" [22]. In Ontology Engineering, CLs are supposed to allow users to design, create, and manage information spaces without knowledge of complicated syntax (such as the OWL syntax) or ontology engineering tools [10].

In this paper, we are going to restrict the state-of-the-art on CLs to some that have been designed to facilitate the development of ontologies to non ontology experts. In this regard, we will consider the Manchester OWL Syntax [14], Attempto Controlled English (ACE) [15], the Rabbit syntax [9], the Sydney OWL Syntax [8], and CLOnE (Controlled Language for Ontology Editing) [10].

The whole set of CLs handled here has been designed to express the logical content of ontologies in OWL DL. However, the formulation of statements in DL is not natural for users with no background in Logics. In order to state the properties of ontology classes and the type of relationships that are permitted among classes, a set of facts has to be made explicit, which is otherwise implicit in NL expressions. For example, in the case of herbivores being animals that eat plants, it should be made explicit that the relation "eat" in regard to herbivores can only be established to "plants". This would be formulated in DL in a compacted way by means of mathematical symbols as:

```
Herbivore ∀ eat Plant
```

(∀ meaning "allValuesFrom", i.e., that the object of the predicate "eat" in this specific relation can only be "Plant").

With the aim of making this syntax more readable to non-logicians, the Manchester Syntax came into existence [14]. Symbols in DL were substituted by NL keywords in English, so that logical expressions such as "intersectionOf", "unionOf", "someValuesFrom", or "allValuesFrom" became "and", "or", "some" and "only", respectively. In this way, the sentence above introduced about herbivores, would become "Herbivore eats only Plant". The main drawback still remained the artificiality of the formulations that just managed to somehow disguise the underlying DL syntax. Users still needed to be conscious of the importance of explicitly declaring that "it is only plants that herbivores eat, and nothing else".

Shortly afterwards, some CLs were created adopting the philosophy behind the Manchester Syntax of making the OWL syntax accessible to the average user. These CLs were ACE (or a subset of ACE called OWL ACE [16]), Rabbit and the Sydney OWL Syntax. The motivation behind their creation was the unnaturalness still present in the Manchester Syntax (lack of determiners, heavy use of parentheses [15]) that posed some obstacles to the domain expert in authoring ontologies.

ACE, Rabbit and the Sydney OWL Syntax are all based on well-defined subsets of the English language that translate directly into OWL, and that leave no place for ambiguities. ACE and the Sydney OWL Syntax make use of an intermediate syntax between the controlled language and OWL (Discourse Representation Structure in the case of ACE, and OWL Functional-Style Syntax for the Sydney Syntax) [21]. Rabbit, however, utilizes the GATE[6] Natural

[6] http://gate.ac.uk/

Language Processing (NLP) architecture to convert the controlled language into OWL. In any case, users are required to become familiar with the languages before using them for editing ontologies. Whereas ACE and the Sydney Syntax are intended to people with no training in formal logics as end users, Rabbit identifies as end users domain experts aided by knowledge engineers, thus somehow hinting at the difficulties ontology modelling may still impose despite the CL. In fact, sentences resulting from the use of the three controlled languages still sound unnatural (cf. Table 1). Examples of sentences or even tool support are foreseen for helping users to familiarize with the languages. Regarding these three initiatives, a task force[7] was formed in 2007 to work towards a common Controlled Natural Language Syntax for OWL 1.1 [21], since approaches were found similar in form and purpose.

Table 1. Examples of sentences in ACE, Rabbit and the Sydney Syntax [21]

CL	*Examples*
ACE	Every bourne is a stream. Every river-stretch has-part at most 2 confluences.
Rabbit	Every Bourne is a kind of Stream. Every River Stretch has part at most two confluences.
Sydney Syntax	Every bourne is a stream. Every river stretch has at most 2 confluences as a part.

Finally, we will refer to the CLOnE approach and its software implementation CLIE. CLOnE is also a controlled language based on the English grammar that relies on GATE NLP tools for matching the sentence in controlled language to a syntactic rule that determines the nature of the ontology element to be modelled. Users are supposed to easily learn the language by following examples and guiding rules. Language and editor are intended to users without expertise in ontology modelling, although resulting sentences may also remind of the syntax underlying the ontology, as in the previous examples (see Table 2).

Table 2. Examples of sentences in CLOnE [10]

CL	*Examples*
CLOnE	Universities and agent are types of persons. Projects have string names.

2.1. Main limitations of CLs in ontology modelling

Up to now, the reviewed approaches have as starting point the OWL DL syntax and create a layer above it, supposedly closer to the syntax of a natural language than to formal logics. However, they keep being quite accurate reflections of the ontological structure.

A part from the unnaturalness of sentences, we see some drawbacks in the analysed CLs that should be overcome when aiming at making OWL ontologies accessible to people with no training in formal logics. These problems can be summarized as: 1) CLs do not provide users any help in solving modelling difficulties; 2) they require some effort on the side of the user to learn, read and write statements; 3) they have all been developed as subsets of the English language.

1) Regarding the first problem we have identified, we are of the opinion that users may have more problems finding out which ontology structure or element allows them to represent certain content in the ontology, than selecting the rule that the CL offers them to model it. In order to understand this position, let us consider the following example. Imagine the user wants to model the relation between a "river stretch" and its "confluences", as in some of the examples in Table 1. The user will have to be previously conscious of the fact that a meronymy relation (part-of) is holding between the two concepts. Then, as a second step, (s)he will search for the corresponding CL formulation (e.g., "has-part" in ACE, or "has...as part" in the Sydney Syntax) that allows her or him to model that relation in the ontology. However, selecting the ontological relation that the untrained user needs to solve a certain modelling issue is not a trivial task, as some experiments have revealed [1]. In the mentioned experiments, Computer Science students with some background in modelling had to identify the most appropriate modelling solution given a modelling problem expressed in NL of the type: *A research plan is composed by a theoretical plan and an experimental plan*. Results showed that nearly half of the solutions (41%) were erroneous according to the golden standard. It is worth mentioning, that the meronymy relation (part-of) was mainly confused with the hypernymy-hyponym relation (subclass-of), among other erroneous solutions.

This gives just some hints of the difficulties untrained users face when having to choose for the most appropriate modelling solution. One could argue that this is to a lesser extent related to CLs in themselves. However, we consider that most of the problems domain experts have when developing ontologies are rather related with modelling decisions, than with choosing the CL syntax to express it. We believe that this is a more demanding and complex issue not really considered by the approaches to CLs analysed in section 2, and which should be handled

[7] http://wiki.webont.org/page/OwlCnl

together. In fact, the analysed approaches on CLs provide no guidelines to the user for making that kind of modelling decisions.

2) It must also be noted that learning to use a CL is by no means trivial, let alone it is fairly close to logics. The implications are not only limited to learning some new grammar structures or rules, but to understand what they represent and imply when modelling. And this bring us to previous point 1), since the difficulties in learning new rules is tightly connected with the content they allow to model in the ontology.

Additionally, some experiments have revealed that users prefer the use of full NL when interacting with machines because "they can communicate their information need in a familiar and natural way without having to think of appropriate keywords in order to find what they are looking for" [17]. This result has been obtained in recent usability studies conducted to investigate how useful Natural Language Interfaces (NLI) are to find data in the Semantic Web. From the four NLIs tested by the 48 users involved in the experiment, the one that required full English questions was judged to be the most useful and "best-liked query interface".

3) To the best of our knowledge, CLs aimed at helping users to semantically represent domain content in OWL are only available for the English language. From our point of view, this represents an obstacle to domain experts no proficient in English.

Because of these limitations, we propose a novel approach that will allow domain experts to move away from ontology modelling paradigms and underlying representation languages, and permit them to concentrate on their modelling needs. Our proposal will be explained in section 4..

3. Controlled Language vs. full Natural Language

A key debate that takes place once and again is the dichotomy between "naturalists" vs. "formalist" approaches to CLs [7]. The set of approaches presented in section 2 can be said to follow a "formalist" paradigm, since they comply with the conditions of being "well-defined, predictable, and deterministically translatable into a formal representation". On the other hand, a "naturalist" CL may be closer to the user, but suffer from language ambiguities. It is undeniable that language ambiguities demand sound NLP tools to discern which the correct interpretation of a sentence is in a certain context. However, it is unquestionable as well, that formalist approaches require a great effort on the side of the user in two aspects:

a) the time and effort the user has to put in learning the language, as pointed out in section 2.1

b) the idea that the more "controlled" the language is, the more the user needs to understand the underlying representation language, or in our case, ontology modelling issues

For this and other reasons, we will detail in the following why we have opted for a sort of naturalist approach in which the way of controlling the user input is made by means of some recommendations; otherwise, users can express in NL what they want to model in the ontology. However, the most innovative aspect of our approach is that we identify those linguistic structures that correspond to consensual modelling components. By doing it so, we allow users to express in NL a modelling issue, and translate it into an ontological structure identified as "best practices" by the Ontology Engineering community.

4. Lexico-Syntactic Patterns for ontology development

The basis of this approach is a repository of linguistic structures or Lexico-Syntactic Patterns (LSPs) identified for different NLs (English, Spanish and German) and matched to consensual modelling solutions called Ontology Design Patterns (ODPs), which are being developed within the NeOn project[8]. The identification of linguistic constructs that convey a relation of interest has been applied in Computational Linguistics with several purposes. Hearst [13] was the first in using them for the task of automatically discovering relations from machine readable dictionaries. The object of her research was the hypernym-hyponym relation, but her patterns were extended by other authors for covering relations such as meronymy, causality, functionality, etc., e.g. [3,5]. In Ontology Engineering, linguistic patterns have been mainly applied with two objectives: 1) to learn classes, attributes, or instances for ontology population, or 2) to learn taxonomic and non-taxonomic relations for ontology building. For a compendium on techniques and tools see [6].

Our approach, however, contributes to the research on LSPs in a novel perspective that focuses on the support to ontology modelling. In this sense, LSPs are rather a means than an end in itself, because they serve the identification of NL expressions that linguistically realize them, with the end of establishing a

[8] http://www.neon-project.org

correspondence to a modelling component. The set of modelling components we are considering at this stage of the research have been developed within the NeOn project [20]. Specifically, our LSPs combine lexical items with syntactic dependencies and have verbs as main elements. A first version of the repository has been published in [1]. The aim of this repository is to serve as basis to a tool for enabling domain expert's formulations in *guided NLs* (English, Spanish and German), giving in return an ontological structure or ODP. The construction of this repository is what demands the most effort on the side of the repository designer, but releases end users of having to understand ontology representation formalisms. The current repository is in a more advanced stage for English, and in a more initial one for Spanish and German.

4.1. LSPs-ODPs correspondence

With the aim of illustrating the process of identification of LSPs and the correspondence establishment to ODPs, let us consider the following sentences in English:

1. *Proteins form part of the cell membrane.*
2. *Water is made up of hydrogen and oxygene.*

Both are fully natural sentences in English expressing a meronymy relation. In the first one, the subject of the sentence (proteins) is the "part" element in the relation, whereas in the second one, the objects in the relation (hydrogen and oxygene) are the "parts" or components of "water". We have also identified a set of verbal forms that behave semantically and syntactically in the same way, such as: "to contain", "to hold", or "to consist of", and have grouped them under Verbs of Composition (COMP). Since we come across the same linguistic structure for expressing the relation between "parts" and its "whole" across different domains, we can identify it as an LSP in English for the "part-whole" relation, and establish a correspondence to the ODP for Simple Part-Whole relation (CP-PW-01), according to the classification in [20].

Finally, linguistic constructs are formalized as shown in Table 3, according to a "Backus Naur Form" extension. The set of restricted symbols and abbreviations used in the formalization of the LSPs included here is to be seen in an Appendix at the end of the paper.

Table 3. LSPs formalization for CP-PW-01 [1]

Formalization
(NP<part>,)* and NP<part> COMP [CN] NP<whole>
NP<whole> be COMP [CN] (NP<part>,)* and NP<part>

However, the correspondence between linguistic constructs and ontological constructs is not always so direct and free from ambiguities. Consider the following examples:

3. *Common mass storage devices include disk drives and tape drives.*
4. *Reproductive structures in female insects include ovaries, bursa copulatrix and uterus.*

In this case, the same linguistic structure deployed by the verb *to include* conveys two different meanings: in 3, it expresses a hypernymy-hyponymy relation between a "Common mass storage device" and its subtypes "disk drives and tape drives"; and in 4, it communicates a meronymy relation between "reproductive structures in female insects" and its parts "ovaries, bursa copulatrix, uterus".

Both sentences are formalized in the same LSP (see Table 4). This means to say that the same LSP can correspond to a hypernymy-hyponymy relation or to a meronymy relation, i.e. to the ODP for SubclassOf relation (LP-SC-01) or to the ODP for Simple Part-Whole relation (CP-PW-01). At this point we need to rely on sound NLP tools (GATE) and disambiguation strategies to find the correct modelling solution that we want to include in the ontology. Some of the disambiguation strategies being currently developed will be outlined in section 5.1.

Table 4. LSP formalization for LP-SC-01 / CP-PW-01

Formalization
NP<class> include | comprise | consist of [(NP<class >,)* and] NP<class>

Once we have disambiguated if we are dealing with a hypernymy-hyponymy relation or a meronymy relation, for a correct modelling in DL we need to establish if the classes in the hypernym-hyponym relation are disjoint or exhaustive. In the previous example *Common mass storage devices include disk drives and tape drives*, this would mean that the hyponyms in the relation "disk drives and tape drives" do not share instances (disjointness) and are a complete enumeration of all the types of "common mass storage devices" that exist (exhaustiveness). By determining these features of the hypernym-hyponym relation, we would be reusing consensual modelling solutions that guarantee a correct modelling in the ontology.

5. Approach for the development of *ontologies by using LSPs*

The repository of LSPs matched to ODPs is the core element of the *NL guided approach* we present here, based on the guidelines proposed in [1]. The main purpose of the repository is to enable the use of consensual modelling components to users with little expertise in Ontology Engineering. The whole process needs to be supported by a system that automatically analyses the sentence in NL introduced by the user, and looks in the LSPs-ODPs repository for the linguistic structure that best matches the input. We should keep in mind that by allowing the use of unrestricted NL, we will have to deal with language ambiguities. This means that the same linguistic *structure can equally express different ontological* relations (as already exemplified in section 4.1). Hence, the user input needs to be revised or refined during the process so that only one option is valid. The method is mainly divided in three tasks:

Task 1. Formulation. The goal of this task is to formulate in NL the domain aspect to be modelled. Since this task is to be included in the wider framework of an ontology development methodology (specifically, the *NeOn Methodology for Building Contextualized Ontology Networks* [19]), we can assume that the user has a good command of the knowledge parcel (s)he wants to model in the ontology. For a good completion of this task, the user gets a list of recommendations comparable to some of the rules in CLs or Simplified Technical English approaches that specify how to write[9] (see Table 5). This is the main reasonfor referring to this approach as a *NL guided approach.*

Table 5. Recommendations for Task 1.

Recommendations
1. Express one topic or idea per sentence.
2. Include in each sentence subject, verb and predicate (Try not to use pronouns instead of nouns!).
3. Avoid using neither interrogative nor negative sentences.
4. Avoid coordination of phrases, and use only when necessary.
5. Avoid including redundant or unnecessary information that does not add new content to the idea.
6. Avoid using acronyms.
7. End up each sentence with full stop.
8. In enumerations, use comas to separate elements.

Some of these advices may appear to be unnecessary, since as has been pointed out in [4], studies have shown that when users communicate with machines (specifically, when querying a knowledge base) they formulate queries in a simple manner, and "(queries) do not consist of complex sentence constructs even when users are neither limited by a conventional search interface nor narrowed by a restricted query language". In any case, recommendations in this approach may have the role of simple reminders.

Task 2. Refinement. The goal of this task is to refine the linguistic structure the user has introduced because no correspondence to one ODP has been found. This might be caused by two reasons: 1) The system does not recognize the linguistic structure introduced by the user, because it is not contained in the LSPs-ODPs repository. 2) The LSPs matched by the system corresponds to several ODPs that represent different ontology modelling components, and a disambiguation process is required. This is the case of the pattern presented in Table 4.

When confronted with situation 1), users would have to introduce the input sentence anew. Strategies to avoid the user being discouraged from using the system are being investigated. If situation 2) happens, further information is required to discern among the possible ODPs. Again, different strategies for the performance of this task are being investigated, such as a) interaction with the user; b) search in available ontologies modelling the same domain of knowledge; c) search in lexical resources with some semantics (WordNet[10]); etc. In section 5.1 we provide an example of strategy a) for illustration.

Task 2. only takes place if no direct correspondence to an ODP has been found, otherwise Task 1 is followed by Task 3.

Task 3. Validation. The goal of this task is to confirm that the ODP proposed as modelling solution is correct. The validation is foreseen to be manually carried out by the user.

5.1. Example of use

In this section, our aim is to exemplify the proposed *NL guided approach* for the development of ontologies reusing consensual modelling components or ODPs. For better showing the deployment of the three tasks, we will use the example of a polysemous LSP conveying hypernymy-hyponymy and meronymy relations (see Table 4). The method could help in the following way:

9 http://www.simplifiedenglish.net/en/ste/what_is_simplified_technical_english.asp

10 http://wordnet.princeton.edu/

Task 1. Formulation. Let us assume that the user wants to model the hypernymy-hyponymy relation held between "user-written software" and its subtypes (spreadsheet templates, word processor macros, scientific simulations, graphics, animation scripts). The first task consists in formulating that according to the recommendations introduced in section 5 (see Table 5). Since the types of "user-written software" are going to be enumerated, the user will have to take recommendation number 6 into account, and write something like: *User-written software include spreadsheet templates, word processor macros, scientific simulations, graphics, and animation scripts.*

Task 2. Refinement. The sentence in NL matches the LSPs corresponding to hypernymy-hyponymy and meronymy relations (see Table 4). As already mentioned, this situation is caused by the ambiguity present in the polysemous verb *to include*. An option would be to interact with the user by means of the so-called *refining questions*. In this example, questions would be:

a) *Are spreadsheet templates, word processor macros, scientific simulations, graphics, and animation scripts, **types of** user-written software?*

b) *Are spreadsheet templates, word processor macros, scientific simulations, graphics, and animation scripts, **parts of** user-written software?*

The answer to question a) should be *yes*, and question b), *no*, if the input sentence wants to model a hypernym-hyponym relation, as we suppose in this example.

Once the correspondence to the hypernymhyponym relation ODP (LP-SC-01) has been obtained, this relation should be further enriched with information about disjointness and exhaustiveness. A similar strategy has also been designed, in which the user is asked:

c) *Can certain user-written software **belong to the group of** spreadsheet templates, word processor macros, scientific simulations, graphics, and animation scripts **at the same time**?* d) ***Are there any other types of** user-written software?* If the answer to question c) were *yes*, the system would further model those hyponyms or subclasses as disjoint classes in the ontology. If the answer to question

d) were *yes*, the system would offer the user the possibility of introducing the missing hyponym(s) in the input window. On the contrary, the system would proceed to model those classes as exhaustive.

Task 3. Validation. In return, the system provides a diagram of the ODP instantiated with the information of the input sentence in Task 1. Then, the user has to confirm the suitability of the modelling solution.

This "user-interaction" strategy can be regarded from a didactic point of view as a way of "teaching" untrained users how to build ontologies, because users are made conscious of the sort of information that has to be made explicit when modelling ontologies.

6. Conclusions

Different approaches to CLs have been created to enable untrained users in ontology engineering to understand and formulate formal representations in ontologies. Although they manage to make the underlying logic constructs readable in English, they do not provide any help in the selection process of the ontology modelling components needed for representing domain content in the ontology. In the research presented in this paper, we have tried to overcome some of the modelling difficulties of non ontology experts by providing a *NL guided approach* that allows the reuse of consensual modelling solutions, or ODPs, having as a starting point a description of the modelling problem in NL. We have discussed some of the limitations imposed by the analysed CLs as "formalist" approaches *versus* the more "naturalist" approach that we suggest. However, a NL guided approach has some drawbacks as well, mainly: 1) a great effort has to go on the construction of a repository of linguistic structures (LSPs) that correspond to ontology modelling components (ODPs), and 2) strategies have to be investigated to cope with NL ambiguities. We are currently working in the design of strategies to overcome the mentioned drawbacks, as pointed out in the paper. Further steps in this research will involve the evaluation of the proposed approach.

7. Acknowledgements

The research presented here has been supported by the European project NeOn (FP6-027595), and the National project GeoBuddies (TSI2007-65677C02).

8. References

[1] G. Aguado de Cea, A. Gómez-Pérez, E. Montiel-Ponsoda, and M.C. Suárez-Figueroa, "Natural Language-based Approach for Helping in the Reuse of Ontology Design Patterns", in Proc. of EKAW08, LNCS Springer 5268, 2008, pp. 32-47.

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

[3] M. Berland and E. Charniak, "Finding parts in very large corpora", in Proc. of the 37th Annual Meeting of the ACL, 1999, pp. 57-64.

[4] A. Bernstein, E. Kaufmann, A. Göhring, and C. Kiefer, "Querying ontologies: A controlled english interface for end-users", in Proc. of ISWC2005, 2005, pp.112-126.

[5] P. Cimiano, and W. Johanna, "Automatic Acquisition of Ranked Qualia Structures from the Web", in Proc. of the Annual Meeting of the ACL, 2007, pp. 888-895.

[6] P. Cimiano, *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*, Springer, 2006.

[7] P. Clark, P. Harrison, W.R. Murray, J. Thompson, "Naturalness vs. Predictability: A Key Debate in Controlled Languages", in CNL2009, 2009.

[8] A. Cregan, R. Schwitter, T. Meyer, "Sydney OWL Syntax -towards a Controlled Natural Language Syntax for OWL 1.1", in Proc. of OWLED, 2007.

[9] C. Dolbear, G. Hart, J. Goodwin, S. Zhou, K. Kovacs, "The Rabbit language: description, syntax and conversion to OWL", Ordenance Survey Research, Technical Report, 2007.

[10] A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, S. Handschuh, "CLOnE: Controlled Language for Ontology Editing", in Proc. of ISWC, 2007.

[11] A. Gómez-Pérez, M. Fernández-López, O. Corcho, *Ontological Engineering*, Springer Verlag, 2003.

[12] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing", Pennsylvania: School of information Sciences and Technology (IST). Pennsylvania State University, 1993.

[13] M. A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora", in Proc. of Coling92, 1992, pp. 539-545.

[14] M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. H Wang, "The Manchester OWL Syntax", in Proc. of OWLED, 2006.

[15] K. Kaljurand and N.E. Fuchs, "Verbalizing OWL in Attempto Controlled English", in Proc. of OWLED, 2007.

[16] K. Kaljurand and N.E. Fuchs, "Bidirectional mapping between OWL DL and Attempto Controlled English", in 4th Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro, 2006.

[17] E. Kaufmann, and A. Berstein, "How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users?", in K. Aberer *et al.* (eds.), *ISWC/ASWC 2007*, 2007, pp. 281-294.

[18] S.G. Pulman, "Controlled Language for Knowledge Representation", in Proc. of CLAW96, 1996, pp. 233-242.

[19] M.C. Suárez-Figueroa, *et al.*, "NeOn D5.4.2. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks", NeOn Project, 2009.

[20] M.C. Suárez-Figueroa, S. Brockmans, A. Gangemi, A. Gómez-Pérez, J. Lehmann, H. Lewen, V. Presutti, and M. Sabou, "NeOn D5.1.1. NeOn Modelling Components", NeOn Project, 2007.

[21] R.Schwitter, K. Kaljurand, A. Cregan, C. Dolbear and Hart, "A Comparison of three Controlled Natural Languages for OWL 1.1", in Proc. of OWLED 2008 DC, 2008.

[22] Schwitter, R., "Controlled natural languages. Technical report", Centre for Language Technology, Macquarie University, 2007

Appendix

SYMBOLS & ABBREVIATIONS in LSPs	
CD	Cardinal Number.
CN	Class Name. Generic names for semantic roles usually accompanied by preposition, such as *class, type, category.*
COMP	Verbs of Composition. Set of verbs meaning that something is made up of different parts. Some of the most representative ones are: *contain, hold, consist of, compose of, make up of, form of/by, constitute of/by.*
NP<... >	Noun Phrase. It is defined as a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers, and that functions as the subject or object of a verb. NP is followed by the semantic role played by the concept it represents in the conceptual *relation in question in <...>, e.g., class, subclass.*
PARA	Paralinguistic symbols like *colon*, or more complex structures as *as follows*, etc.
QUAN	Quantifiers such as *all, some, most, many, several, every.*
()	Parentheses group two or more elements.
*	Asterisk indicates repetition.
[]	Elements in brackets are meant to be optional, which means that they can be present either at that stage of the sentence or not, and by default of appearance, the pattern remains unmodified.