Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns

Guadalupe Aguado de Cea, Asunción Gómez-Pérez,
Elena Montiel-Ponsoda, and 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
{lupe, asun, mcsuarez}@fi.upm.es,
emontiel@delicias.dia.fi.upm.es

Abstract. Experiments in the reuse of Ontology Design Patterns (ODPs) have revealed that users with different levels of expertise in ontology modelling face difficulties when reusing ODPs. With the aim of tackling this problem we propose a method and a tool for supporting a semi-automatic reuse of ODPs that takes as input formulations in natural language (NL) of the domain aspect to be modelled, and obtains as output a set of ODPs for solving the initial ontological needs. The correspondence between ODPs and NL formulations is done through Lexico-Syntactic Patterns, linguistic constructs that convey the semantic relations present in ODPs, and which constitute the main contribution of this paper. The main benefit of the proposed approach is the use of non-restricted NL formulations in various languages for obtaining ODPs. The use of full NL poses challenges in the disambiguation of linguistic expressions that we expect to solve with user interaction, among other strategies.

Keywords: Ontology Design Patterns Reuse, Lexico-Syntactic Patterns, ontology modelling, restricted vs. unrestricted NL.

1 Introduction

Ontology modelling has become one of the main issues in the current Ontology Engineering research, since ontologies are becoming more and more popular among a wider range of users. Many efforts have been directed to the creation of methodologies for guiding users in the development of ontologies. Some of the well-known methodological approaches are METHONTOLOGY [12], On-To-Knowledge [28], and DILIGENT [20]. They provide guidelines to help users building ontologies from scratch, but lack guidelines for building ontologies by reusing available ontological and non ontological resources. In this regard, one of the crucial issues in ontology development is the need for guidelines or methods in the reuse of Ontology Design Patterns (ODPs hereafter). ODPs have been defined as *archetypal solutions to design problems* [10]. As already proven in Software Engineering, field in which the reuse of object-oriented design patterns has a longer tradition, design pattern reuse is deemed especially suitable for speeding up the development of resources, because not only

A. Gangemi and J. Euzenat (Eds.): EKAW 2008, LNAI 5268, pp. 32–47, 2008.

does it *encourage the reuse of best practices and improves communication among users*, but it also *allows less experienced users to produce a better design* [21]. Design pattern reuse in object-oriented design is an extended practice, well supported by design pattern repositories and manuals as [3,9]. However, most of them presuppose prior design knowledge and expertise. This fact and other limitations are being recently discussed in public forums[1]. The main limitations are related to the lack of general methodologies 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.

A similar situation is to be found in the Ontology Engineering field, in which ODPs have emerged as a way for helping ontology practitioners to model OWL ontologies. The W3C Semantic Web Best Practices and Deployment Working Group[2] proposes patterns for solving design problems for OWL, independently of a particular conceptualization, thus addressing logical problems. In [19] some well known Semantic Web best practices related to W3C activities are presented in a so-called cookbook style, which makes easier the identification and application of best practices. Additionally, [10] presents patterns for solving (in OWL or other logical languages) design problems for the domain classes and properties that populate an ontology, thus addressing content problems. In the NeOn project[3], special attention is also being devoted to ODPs. In D5.1.1 [30], a general template for describing ODPs has been created, and an initial repository of OWL-based ODPs is presented. In the same line, D2.5.1 [22] classifies ODPs in *Structural*, *Correspondence*, *Content*, *Reasoning*, and *Presentation* ODPs, and provides a repository of Content ODPs. This latter repository is being extended, and it is expected to be available on-line in 2008 at the Ontology Design Patterns.org[4] wiki page. However, despite reported initiatives, no effort has been put in the creation of methods *per se*. This need is, if anything, more urgent if we take into account results of experiments on ODPs reuse, which shed some light about the difficulties users encounter when selecting the appropriate ODP during ontology development (cf. section 2).

Also related with the ontology development, there is a general concern about bringing ontology methods and technologies closer to the untrained user in ontology modeling. In this sense, there exist some initiatives based on the creation of controlled languages that make ontology languages based on logics more understandable to non logicians. Such controlled languages allow untrained users to develop ontologies, but without taking into account ODPs (cf section 3).

Therefore, the aim of our research is to alleviate to some extent the problems naive users encounter when modelling ontologies reusing ODPs, by means of a method and the tool that supports it, for semi-automatically reusing ODPs starting from full Natural Language (NL) formulations of domain aspects. The method and the tool rely on a repository of Lexico-Syntactic Patterns (henceforth LSPs) associated to ODPs, and on NL processing tools that annotate the user input with lexical and syntactic information. In this paper, our main contribution consists of an excerpt of the initial repository of LSPs for the English language, as well as the foreseen user interaction for solving problems imposed by the use of full NL.

[1] http://www.pattern.ijop.org
[2] http://www.w3.org/2001/sw/BestPractices/
[3] http://www.neon-project.org
[4] http://www.ontologydesignpatterns.org

The reminder of this paper is structured as follows. Section 2 describes two preliminary experiments on ODPs reuse that show up some of the problems faced by users. In section 3 we discuss related work on controlled languages for helping naive users in ontology modelling. The proposed method and the tool workflow are briefly outlined in section 4, and then, this section is devoted to LSPs identification and includes an excerpt of the preliminary repository of LSPs for English. The method workflow is exemplified in section 5, and we conclude in section 6.

2 Motivation

The development of ontologies using directly ontology languages, such as OWL DL, is not as trivial as reported in [23]. Alan Rector et al. [23] present the most common problems, errors, and misconceptions on understanding OWL DL as well as tips on how to avoid such pitfalls in building OWL DL ontologies. In [23] it is stated that for most people it is very difficult to understand the logical meaning and potential inferred statements of any DL formalism, including OWL DL. Such a paper mentions that one of the most common errors in building ontologies in OWL is to omit the disjointness axioms, when taxonomies are being modeled within ontologies.

Apart from that, it is known that time and resources used for the development of ontologies can be reduced by means of reusing ODPs. Informal evidence such as subjective experience reports of ontology engineers have so far shown the benefits of using patterns in ontology engineering. In fact, logical and content patterns as defined in [30] have for example been used for teaching, in tutorials and summer schools, and it has, according to the tutors, improved the way students solve modelling tasks. However, the reuse of ODPs is not so flawless, as two experiments carried out in the context of the NeOn project show.

These two preliminary experiments involving students at Universidad Politécnica de Madrid (UPM) and at the 5th Knowledge Web Summer School on Ontological Engineering and the Semantic Web (SSSW07) were designed and conducted during 2007. The experiments gave some very important initial results and insights on how subjects really perceive, understand, and use ODPs.

The **first experiment** was carried out with 69 undergraduate students attending the "Artificial Intelligence" course at UPM *Facultad de Informática*. Students were provided with 13 simple modelling problems they had to solve using ODPs, and white papers. They had background in database modelling (e.g., entity relationship), software engineering modelling (e.g., UML), and AI modelling (e.g., frames, semantic networks, DL). The goal was to test if, given a modelling problem expressed in NL, students were able to identify the most appropriate pattern for such a problem. When this was not the case, the objective was to know whether a wrong pattern or antipattern was typically used instead of the most appropriate one.

General conclusions are that 59% of the solutions provided by students corresponded with the expected ODP, that is, the pattern modelling the given problem. 22% of the solutions included a wrong pattern, 11% were empty, and 8% of the answers were wrong but without involving any ODP. Focusing on wrong patterns, we can extract the following conclusions: *subclass-of relation* is mainly confused with *exhaustive classes* and *disjoint classes*; *exhaustive classes* is mainly confused with

subclass-of relation and *disjoint classes*; and *part-whole relation* is mainly confused with *subclass-of relation*. Concrete results related with the aforementioned conclusions are shown in Table 1.

Table 1. Results of the first experiment

Modelling Problem	*Solution*	Number of Students providing the right pattern	Number of Students providing a wrong pattern				
Tasks are management tasks, financial tasks, marketing tasks and control tasks.	*Subclass-of relation (LP-SC-01)*	36	20	Exhaustive classes	Disjoint classes	Individual	Part-Whole
				10	8	1	1
Plans are management plans, financial plans, marketing plans and control plans.	*Subclass-of relation (LP-SC-01)*	39	17	Disjoint classes	Exhaustive classes	Individual	Part-Whole
				8	7	1	1
Control tasks can be only begin task, end task or sequential task; and that a concrete task can only be classified in one of these categories.	*Exhaustive classes (LP-EC-01)*	27	26	Subclass-of relation	Disjoint classes	Individual	Datatype property
				18	6	1	1
Research plan is composed by a theoretical plan and an experimental plan.	*Part-Whole relation (CP-PW-01) or (CP-PW-02)*	35	29	Subclass-of relation	Object property	Exhaustive classes	
				15	9	5	

The **second experiment** was carried out with 50 Master and PhD students attending the SSSW07. Students had different backgrounds in modelling (databases, software, etc.) but no extensive practical experience in ontology modelling. The goal was to test if a subset of logical and content design patterns described in NeOn deliverable D5.1.1 [30] was well explained, and if patterns were easy to understand and to apply in specific modelling problems.

The experiment was divided into two parts. Students had 45 minutes to carry out the first part of the experiment, and 45 minutes to carry out the second part of the experiment. They were provided with 10 modelling problems for the first part, and another 10 for the second. In both cases, students had to solve modelling problems using ODPs. However, in the first part they had no external support, while in the second part they were provided with the NeOn catalogue of ODPs [30].

From the first part of the experiment, in which students solved modelling problems without the ODPs catalogue, we draw the following conclusions:

- 66% of the students did not identify *exhaustive classes*; but provided solutions including a wrong pattern or antipattern, mainly *object property* or *subclass-of*.
- 74% of the students did not correctly identified *n-ary relation*, mistaking such pattern for *object property* in most of the cases.
- A considerable number of students mistook *part-whole* for *subclass-of* or *object property*.

From the second part of the experiment, in which students approached modelling problems using the NeOn ODPs catalogue, we draw the following conclusions:

- 52% of the students confused *subclass-of* with *disjoint classes*.
- 62% of the students confused *exhaustive classes* with *subclass-of*.
- 66% of the students confused *disjoint classes* with *exhaustive classes*.
- 54% of the students confused *n-ary relation* with *object property* or *datatype property*.
- A considerable number of students mistook *part-whole* for *object property*, *subclass-of* or *subproperty-of*.

Taking into account the results of both experiments, we can conclude that guidelines to easily differentiate, first, *subclass-of relation* from *disjoint classes* or *exhaustive classes*, and, second, *part-whole relations* from *subclass-of* and *object property*, should be provided. Based on this, our research aims at tackling these difficulties (as shown in the LSPs repository), without leaving aside other patterns, with the idea of providing an easy identification of the ODPs included in D5.1.1 to users in general, but centred in naive users.

3 Related Work

Regarding the idea of facilitating ontology modelling and bringing ontology technologies closer to the average user, some research has been devoted to the creation of syntaxes or controlled languages[5] to make ontology languages more readable and understandable to naive users. Some examples of controlled languages are: the *Manchester Syntax* [14], the *Attempto Controlled English* (ACE) [15], the *Rabbit* syntax [7], the *Sydney OWL Syntax* [6], or *CLOnE* (Controlled Language for Ontology Editing) [8]. The main idea underlying controlled languages is to allow naive users to express their modelling needs following certain syntactic rules. In return, they obtain an ontology that represents the knowledge associated to the users needs. In the analyzed controlled languages, the obtained ontology is built without using ODPs.

The Manchester Syntax came into existence as a result of teaching experiences on OWL intended for domain experts that made evident that non-logicians had difficulties in understanding the OWL syntax [14]. The result was a syntax mainly based on NL keywords in English (*and*) equivalent to logical expressions (*intersectionOf*). With the aim of avoiding some of the drawbacks of the Manchester Syntax, mainly related to the unnaturalness of the resulting sentences [15], further controlled languages were created as ACE, Rabbit or the Sydney Syntax. These are also based on a subset of English and follow similar approaches, which is why a task force was formed in 2007 to work towards a common *Controlled Natural Language Syntax for OWL 1.1* [32]. In much the same way, CLOnE and its software implementation CLIE, proposed not only a controlled language based on the English grammar, but also an NL interface to help users build ontologies, since they considered common ontology editors to be designed for ontology engineering experts.

[5] A controlled language is an engineered subset of a natural language with explicit constraints on grammar, lexicon, and style. [32].

Undoubtedly, the analyzed approaches allow users to build ontologies using a syntax more closed to NL than to OWL. However, there are still some limitations regarding the efforts users need to make in order to become familiar with these controlled languages before being able to start using them. Furthermore, sentences resulting from the use of controlled languages are quite artificial, since they just manage to disguise the underlying OWL syntax (see examples from [32,8] in Table 2), what makes learning even harder. Finally, they all rely on the English syntax, without taking into account other languages.

Table 2. Examples of sentences resulting from the use of controlled languages

Approach	Resulting sentence
ACE	*Every river-stretch has-part at most 2 confluences.*
Rabbit	*Every Bourne is a kind of Stream.*
Sydney Syntax	*The classes petrol station and gas station are equivalent.*
CLOnE	*Projects have string names.*

On the contrary, our natural language approach for facilitating the ontology modelling to the average user wants to go a step further to allow naive users to express the domain aspect they want to model in unrestricted NL. We aim at being able to move away from ontology modelling paradigms or languages, and concentrate on the domain parcels (s)he needs to formalize. The obvious consequence of committing to *naturalness* is to deal with language ambiguities, which we expect to solve resorting to users feedback (see section 5) or accessing linguistic resources, among other strategies. Additionally, we want to provide a multilingual modelling environment in which not only is the interface multilingual, but also the sentences that express the domain aspect to be modelled can be written in several languages. Actually, we expect to cover English, Spanish, and German in the first prototype of the tool. Finally, our approach is based on consensual modelling solutions, i.e., ODPs for creating the ontology that models user needs.

4 Reuse of Ontology Design Patterns by Naive Users

With the aim of helping users with different backgrounds and expertise in ontology modelling to reuse ODPs, we propose a method for the reuse of ODPs (already outlined in [18,31]) 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 a NeOn UML diagram representing the suitable ODP instantiated with information from the input (see Fig. 1). We assume that the user has a good command of the domain (s)he wants to model, and that the information expressing the modeling aspect in NL corresponds to the knowledge (s)he wants to represent[6]. Note that this method allows users to *freely introduce a sentence in NL*, without any kind of restrictions as imposed by controlled languages, which in some cases may imply

[6] If the user introduces a sentence like *Animals are divided into vertebrates and omnivores*, the system will not analyze whether such a sentence is adequate from the content viewpoint.

further actions, such as interaction with the user for refining the input (as will be explained in section 5). The method can be divided in 3 main tasks:

1) Task 1. ODPs *Formulation*. The goal of this task is to formulate in full NL 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 to refine the input from Task 1. This task is only carried out when there is no direct correspondence to one ODP. The user may have to answer a set of *refining questions*. Refinement may be required because of ontology enrichment needs or lexical ambiguities.
3) Task 3. ODPs *Validation*. The goal of this task is to confirm that the resulting ODP meets user expectations.

The tool that supports this method is called *S.O.S., System for Ontology design pattern Support*, and enables a semi-automatic selection of ODPs, and its integration in the ontology being built. This tool is under development and will be integrated as a plug-in in the NeOn toolkit[7], and used in combination with its ontology editor. The underpinnings of the S.O.S. are constituted by a set of LSPs, since they are the necessary mechanisms to bridge the gap between NL formulations and ODPs.

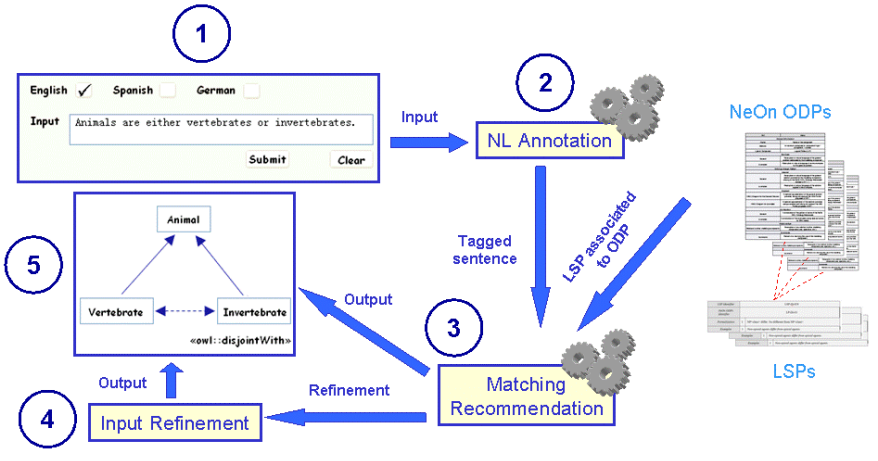

Fig. 1. S.O.S. Workflow

4.1 From Natural Language to Ontology Design Patterns

The central part of this research was the enrichment of the ODPs described within the NeOn project (available in [30]) with an NL field. This NL field consists of a set of LSPs that represent the most usual ways a language has for conveying the conceptual relations formalized in ODPs. LSPs are defined here as *formalized linguistic schemas or constructions derived from regular expressions in NL that consist of certain linguistic and paralinguistic elements, following a specific syntactic order, and that*

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

permit to extract some conclusions about the meaning they express (inspired in [13,17]). Our approach to derive LSPs from NL formulations is based on the assumption that any language has a number of lexical and/or syntactical mechanisms to reliably convey a relation of interest, which in this case is the one represented by the ODP. Contrary to Hearst patterns[8] [13], whose main elements were prepositional phrases, paralinguistic signs or conjunctions, in the first stage of this research we have just considered *verb-oriented* LSPs in the sense of [27, 5], which are mainly composed by tuples of *subject-verb-object*, although we also consider compound sentences, i.e., sentences in which more than one verb is involved. We assume that for expressing how concepts are related in ontologies, we make use of verbs in affirmative or declarative sentences in the simple present tense[9]. In this kind of patterns, verbs are the ones that carry the semantics of the relation.

The idea of applying LSPs to discover semantic relations was introduced in Computation by Hearst [13] in the early 1990s. The goal of her research was the automatic acquisition of lexical syntax and semantics from machine readable dictionaries. Since then, many authors have applied LSPs for the automatic discovery of semantically related lexical items from unstructured texts with different purposes. In Ontology Engineering this has been mainly applied with the aim of automatically or semi-automatically learning classes and/or instances to enrich or populate ontologies [2,4,11,24,26,33]. A second line of research which strongly relies on the use of LSPs is the one that tries to learn taxonomical and non-taxonomical relations between concepts for the construction of ontologies [2,5,16,25,29].

Our approach, however, contributes to the research on LSPs in a new and promising perspective that focuses on the *support to ontology modelling*. In this sense, LSPs are rather a means than an end in itself, because they are the means to identify those NL expressions that instantiate them, with the end of establishing a correspondence to ODPs. Needless to say that most of the research on LSPs has been widely done for the English language, although some scarce attempts have been carried out for French [16], German [33], or Spanish [1] covering different semantic relations. In our case, we aim at identify LSPs for English, Spanish and German corresponding to the same set of ODPs.

To illustrate the process of LSPs identification, let us consider the following sentence in English: *Medications are generally classified into two groups: over-the-counter (OTC) medication and prescription only medicines (POM).*

In this sentence, the verb *classify into* indicates a hypernym-hyponym relation, in which the *superclass* is at the left-hand side of the verb, and the *subclasses* at the right-hand side. In fact, there is a group of sentences constructed in a similar way from which we could draw an LSP embracing all of them (see also Table 3 for symbols and abbreviations in LSPs):

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

[8] Some examples of Hearst patterns are: (1) *NP such as {NP_1, NP_2… (and | or) NP_n}*, (2) *NP {,NP}* {,} or other NP*, and (3) *NP {,} especially {NP,}* { or | and } NP.*

[9] For example: *Animals are divided into two major categories: vertebrates and invertebrates*, would be the usual way of describing domain knowledge.

Table 3. Restricted symbols and abbreviations in LSPs

SYMBOLS & ABBREVIATIONS	DESCRIPTION
AP<…>	Adjectival Phrase. It is defined as a phrase whose head is an adjective accompanied optionally by adverbs or other complements as prepositional phrases. AP is followed by the semantic role played by the concept it represents in the conceptual relation in question in <…>, such as e.g., *property*.
CATV	Verbs of Classification. Set of verbs of classification plus the preposition that normally follows them. Some of the most representative verbs in this group are: *classify in/into, categorize in/into, sub-classify in/into*.
CD	Cardinal Number.
CN	Class Name. Generic names for semantic roles usually accompanied by preposition, such as *class, group, type, member, subclass, category*, *part, set,* etc.
COMP	Verbs of Composition. Set of verbs meaning that something is made up of different parts. Some of the most representative ones are: *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*.
( )	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.
¬	Elements preceded by this symbol should not appear in the pattern.

The sketched process for identifying LSPs is not always so trivial, since ODPs are organized in such a way that they do not always coincide with the way NL expresses conceptual relations. It may happen that an LSP corresponds to more than one ODP. This occurs when, e.g., some NL constructs represented by the same LSP have a different meaning because the verb -which is the one that conveys the semantic relation in our patterns- is polysemous[10]. In order to account for the correspondence between LSPs and ODPs, we decided to classify LSPs in three types:

1. *1 LSP corresponds to 1 ODP*
2. *1 LSP corresponds to N ODPs*
3. *1 LSP corresponds to the combination of N ODPs*

According to this classification, LSPs that are directly identified to one ODP and cannot be associated to a different one belong to the first group. LSPs that can correspond to more than one ODP because the verb they contain is polysemous, or because

[10] Words that *have multiple meanings* are considered polysemous or polysemic [http://www.merriam-webster.com/dictionary/]

different modelling decisions can be taken, fall into the second group. In this case, a process of disambiguation or refining is needed in order to obtain the ODP that corresponds to the input sentence. Finally, the third group includes those LSPs that are realized by two or more ODPs, which means that in order to represent the information expressed in NL, a combination of various ODPs is needed. For the purpose of exemplification, section 5 will be devoted to correspondence case 2, *1 LSP corresponds to N ODPs*.

4.2 Preliminary Repository of LSPs

At this stage of the research, we have identified a set of LSPs from NL expressions in English, Spanish and German. LSPs are considered to be language dependant and not interchangeable among different NLs, despite some overlapping [16]. Currently, the LSPs identification process is in a more advanced stage for English, and in a more initial one for Spanish and German. For space reasons, we have just included here LSPs for English. Then, in order to complement NeOn ODPs with NL information by means of LSPs, we designed a template for a unified description of LSPs that consists of four slots, as shown in Table 4:

- *LSPs Identifier*. This mandatory slot contains an acronym composed of: LSP, plus the acronym of the relation captured by the ODP, plus the ISO-639 code for representing the name of the language for which the LSP is valid.
- *NeOn ODPs Identifier*. This mandatory slot inherits the ODP identifier used in the NeOn ODPs repository within D5.1.1 [30]. Identifiers are composed of the component type (e.g. LP standing for Logical Pattern, or CP for Content Pattern), component (e.g. SC standing for *SubClassOf*), and number of the pattern (01).
- *Formalization*. This mandatory slot includes the various LSPs that express the relation contained in the corresponding ODPs. LSPs have been formalized according to a BNF extension (see Table 3).
- *Examples*. This optional slot shows some examples of sentences in NL that match the LSPs in question.

Table 4. Template for LSPs

LSP Identifier	An acronym composed of LSP + ODP component + ISO code for language
NeOn ODPs Identifier	An acronym composed of component type + component + number
Formalization	LSPs formalized according to BNF extension
Examples	Sentences in NL that exemplify corresponding LSPs

From the LSPs identified up to now, we present here a preliminary excerpt, which includes LSPs for the following NeOn ODPs [30]: *subclass-of relation*, *equivalence relation between classes*, *object property*, *datatype property*, *disjoint classes*, *simple part-whole relation*, and *participation*.

According to the classification presented in section 4.1, we divide the LSPs repository in 3 sets: **(1)** 1 LSP corresponds to 1 ODP; **(2)** 1 LSP corresponds to N ODPs; **(3)** 1 LSP corresponds to the combination of N ODPs.

(1) 1 LSP corresponds to 1 ODP

Table 5. LSPs corresponding to *subclass-of relation* ODP

LSP Identifier	LSP-SC-EN	
NeOn ODPs Identifier	LP-SC-01	
Formalization	1	NP<subclass> be [CN] NP<superclass>
	2	[(NP<subclass>,)* and] NP<subclass> be [CN] NP<superclass>
	3	[(NP<subclass>,)* and] NP<subclass> (group in\into\as) | (fall into) | (belong to) CN NP<superclass>
	4	NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>,)* and NP <subclass>
	5	There are CD CN NP<superclass> PARA [(NP<subclass>,)* and] NP<subclass>
Examples	1	*An orphan drug is a type of drug.*
	2	*Odometry, speedometry and GPS are types of sensors.*
	3	*Thyroid medicines belong to the general group of hormone medicines.*
	4	*Membrane proteins are classified into two major categories, integral proteins and peripheral proteins.*
	5	*There are two types of narcotic analgesics: the opiates and the opioids.*

Table 6. LSPs corresponding to *object property* ODP

LSP Identifier	LSP-OP-EN	
NeOn ODPs Identifier	LP-OP-01	
Formalization	1	NP<class> VB ¬ (be | have | CATV) NP<class>
Examples	1	*Birds build nests.*

Table 7. LSPs corresponding to *datatype property* ODP

LSP Identifier	LSP-DP-EN	
NeOn ODPs Identifier	LP-DP-01	
Formalization	1	Property\lies | characteristic\s | attribute\s of NP<class> be [PARA] [(NP<property>,)* and] NP<property>
	2	NP<class> be [(AP<property>,)*] and AP<property>
Examples	1	*Properties of mammals are hair, sweat glands, milk, and giving live birth.*
	2	*Metals are lustrous, malleable and good conductors of heat and electricity.*

Table 8. LSPs corresponding to *disjoint classes* ODP

LSP Identifier	LSP-Di-EN	
NeOn ODPs Identifier	LP-Di-01	
Formalization	1	NP<class> differ | be different from NP<class>
Examples	1	*Non-opioid agents differ from opioid agents.*

Table 9. LSPs corresponding to *simple part-whole relation* ODP

LSP Identifier	LSP-PW-EN	
NeOn ODPs Identifier	CP-PW-01	
Formalization	1	(NP<part>,)* and NP<part> COMP [CN] NP<whole>
	2	NP<whole> be COMP [CN] (NP<part>,)* and NP<part>
Examples	1	*Proteins form part of the cell membrane.*
	2	*A state machine workflow is made up of a set of states, transitions, and actions.*

Table 10. LSPs corresponding to *participation* ODP

LSP Identifier	LSP-PA-EN	
NeOn ODPs Identifier	CP-PA-01	
Formalization	1	NP<object> participate | take part in [(NP<event>,)* and] NP<event> [in| from | during] [NP<time-interval> to NP<time-interval>]
Examples	1	*Project managers participate in business unit management and marketing.*

(2) 1 LSP corresponds to N ODPs

Table 11. LSPs corresponding to s*ubclass-of relation* and *simple part-whole relation* ODPs

LSP Identifier	LSP-SC-PW-EN	
NeOn ODPs Identifier	LP-SC-01 CP-PW-01	
Formalization	1	NP<class> include | comprise [(NP<class >,)* and] NP<class>
	2	NP<class> be divided | split | separate in|into | [CN] [(NP<class >,)* and] NP<class>
Examples	1	*Arthropods include insects, crustaceans, spiders, scorpions, and centipedes.* (LP-SC-01) *Reproductive structures in female insects include ovaries, bursa copulatrix and uterus.* (CP-PW-01)
	2	*Marine mammals are divided into three orders: Carnivora, Sirenia and Cetacea.* (LP-SC-01). *The cerebrumis divided into two major parts: the right cerebral hemisphere and left cerebral hemisphere.* (CP-PW-01)

Table 12. LSP corresponding to *object property*, *datatype property* and *simple part-whole relation* ODPs

LSP Identifier		LSP-OP-DP-PW-EN
NeOn ODPs Identifier		LP-OP-01 LP-DP-01 CP-PW-01
Formalization	1	NP<class> have NP<class>
Examples	1	*Birds have feathers.* (The three ODPs could correspond to the LSP expressed in this sentence. A modelling decision has to be taken according to the user's needs).

(3) 1 LSP corresponds to the combination of N ODPs

Table 13. LSP including *subclass-of relation* and *disjoint classes* ODPs

LSP Identifier		LSP-SC-Di-EN
NeOn ODPs Identifier		LP-SC-01 + LP-Di-01
Formalization	1	NP<superclass> be | CATV [either] NP<subclass> or NP<subclass>
	2	NP<superclass> be divide inlinto | split | separate inlinto [either] NP<subclass> and NP<subclass>
Examples	1	*Animals are either vertebrates or invertebrates.*
	2	*Sensors are divided into two groups: contact and non-contact sensors.*

5 An Example of ODPs Reuse by Naive Users

In this section, our aim is to exemplify the proposed method, which is supported by the S.O.S. tool.

Task 1.: Let us imagine that the user introduces the following sentence in English in the S.O.S. input window: *Arthropods include insects, crustaceans, spiders, scorpions, and centipedes*. For exemplifying the method, we assume that the user wants to represent a *subclass-of relation*.

Task 2.: The system would identify that the resulting tagged sentence has a correspondence with the LSP identified as LSP-SC-PW-EN (see Table 11) in the LSPs repository presented in section 4.2. Whenever the correspondence is *1 LSP to N ODPs*, a refinement process is needed. As already introduced, this situation results from the ambiguity present in the polysemous verb *include*, since it can correspond to two ODPs, one modelling the *subclass-of relation*, and the other modelling the *simple part-whole relation*. For these cases, 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 insects, crustaceans, spiders, scorpions, and centipedes, types of arthropods?*
b) *Are insects, crustaceans, spiders, scorpions, and centipedes*, *parts of an arthropod?*
The answer to the first question should be *yes*, and to the second, *no*, if the input sentence wants to model a s*ubclass-of relation*, as we suppose in this example. In this way, the system would help users to come to the right decision. Once the correspondence to the *subclass-of relation* ODP has been obtained, it would be recommendable from an ontological viewpoint to enrich this relation with knowledge about disjointness and exhaustiveness. A similar strategy has been also designed with the end of finding out if the classes in a *subclass-of relation* are additionally disjoint and/or exhaustive (cf. [18]), although further strategies that do not require user interaction are being explored.

Task 3.: The system returns a UML diagram modelling, in this case, the *subclass-of*, and additionally *disjoint* or *exhaustive classes* relations depending on users answers [18]. The UML diagram is 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.

Seemingly, a sentence like *Birds have feathers* corresponding to LSP-OP-DP-PW-EN (LSP for *object property*, *datatype property* and *simple part-whole relation*) needs to be disambiguated because three different modelling solutions are possible. Here, however, we are not only dealing with the multiple polysemous senses a verb can have, but also with the different modelling decisions the user can take according to his or her needs, that is, (a) *feather* as a class related to the class *bird*, (b) *feather* as a property of *bird*, or (c) *feather* as *parts of bird*. Conscious of the intricacy, but at the same time the importance of such a modelling problem, we are currently investigating different approaches to deal with this and other difficulties that arise when dealing with full NL.

6 Conclusions and Further Lines of Work

Results of initial experiments on ODPs reuse have strengthen the hypothesis that even users with some background on ontology modelling face difficulties when reusing ODPs for their needs. Based on these results and due to the lack of guidelines that may help users in building ontologies by reusing Ontology Design Patterns (ODPs), we have proposed a novel method for the reuse of ODPs aimed at naive users. Therefore, with the aim of proposing a method that can guide naive users in the reuse of ODPs, we have analyzed controlled languages designed to facilitate untrained users the process of ontology modelling. The main limitations of such approaches, which are not ODPs centred, are related with (a) the efforts users have to make for getting trained in the use of such languages, and (b) the fact that they are constrained to the English language, without taking into account other languages.

Consequently, our approach wants to go a step further in the sense of allowing ontology modelling by means of the reuse of ODPs starting from full NL formulations in several languages. In order to enable this process, we designed the S.O.S. system that used in combination with the NeOn toolkit ontology editor, will permit ODPs

reuse from NL expressions of the domain aspect to be modelled. The S.O.S. system relies on a set of Lexico-Syntactic Patterns for bridging the gap between a NL input and the corresponding set of ODPs, from which we have included an excerpt in this paper. However, the correspondence between NL formulations and ODPs is not always so trivial, even less if we take into account two key factors: (1) language ambiguities, and (2) differences between how natural languages express conceptual relations, and how ODPs do. While some of this complex issues can be solved with user interaction, as we have shown in section 5, our future work will concentrate on designing additional strategies for solving these and other related problems.

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*.

References

1. Álvarez de Mon, I., Aguado de Cea, G.: The phraseology of classification in Spanish: integrating corpus linguistics and ontological approaches for knowledge extraction. Presented in BAAL/IRAAL (2006)
2. Aussenac-Gilles, N., Jacques, M.P.: Designing and Evaluating Patterns for Ontology Enrichment from Texts. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 158–165. Springer, Heidelberg (2006)
3. Bushmann, F., Meunier, R., Rohnert, H., Sommerland, P., Stal, M.: Pattern-oriented software architecture. A system of patterns. John Wiley & Sons, Chichester (1996)
4. Charniak, E., Berland, M.: Finding parts in very large corpora. In: Proc. of the 37th Annual Meeting of the ACL, pp. 57–64 (1999)
5. Cimiano, P., Johanna, W.: Automatic Acquisition of Ranked Qualia Structures from the Web. In: Proc. of the Annual Meeting of the ACL, pp. 888–895 (2007)
6. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL Syntax -towards a Controlled Natural Language Syntax for OWL 1.1. In: Proc. of OWLED (2007)
7. Dolbear, C., Hart, G., Goodwin, J., Zhou, S., Kovacs, K.: The Rabbit language: description, syntax and conversion to OWL. Ordenance Survey Research. Technical Report (2007)
8. Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S.: CLOnE: Controlled Language for Ontology Editing. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 142–155. Springer, Heidelberg (2007)
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, New York (1995)
10. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: Musen, M.A., et al. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005)
11. Girju, R., Badulescu, A., Moldovan, D.: Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. In: Proc. of the HLT-NAACL 2003 (2003)
12. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer, Heidelberg (2003)
13. Hearst, M.A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proc. of 14th International Conference on Computational Linguistics, pp. 539–545 (1992)
14. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.H.: The Manchester OWL Syntax. In: Proc. of OWLED (2006)

15. Kaljurand, K., Fuchs, N.: Verbalizing OWL in Attempto Controlled English. In: Proc. of OWLED (2007)
16. Marshman, E., Morgan, T., Meyer, I.: French patterns for expressing concept relations. Terminology 8(1), 1–29 (2002)
17. Meyer, I.: Extracting knowledge-rich contexts for terminography. In: Borigault, D., Jacquemin, C., L'Homme, M.C. (eds.) Recent Advances in Computational Terminology, pp. 279–302. John Benjamins, Amsterdam (2001)
18. Montiel-Ponsoda, E., Aguado de Cea, G., Gómez-Pérez, A., Suárez-Figueroa, M.C.: Helping Naive Users to Reuse Ontology Design Patterns. In: Proc. of the 1st Workshop on Knowledge Reuse and Reengineering over the Semantic Web at ESWC 2008 (2008)
19. Pan, J.Z., Lancieri, L., Maynard, D., Gandon, F., Cuel, R., Leger, A.: Knowledge Web Deliverable D1.4.2.v2. Success Stories and Best Practices (2007)
20. Pinto, H.S., Tempich, C., Staab, S.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: López de Mantaras, R., Saitta, L. (eds.) Proc. of ECAI 2004, pp. 393–397. IOS Press, Amsterdam (2004)
21. Prechelt, L.: An experiment on the usefulness of design patterns: Detailed description and evaluation. Technical Report. University of Karlsruhe (1997)
22. 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)
23. Rector, A., Drummond, N., Horridge, M., Rogers, M., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)
24. Reinberger, M.L., Spyns, P.: Discovering knowledge in texts for the learning of DOGMA-inspired ontologies. In: Proc. ECAI-Workshop Ontology Learning and Population (2004)
25. Sánchez, D., Moreno, A.: Learning non-taxonomic relationships from web documents for domain ontology construction. Data Knowledge Engineering 64, 600–632 (2008)
26. Snow, R., Jurafsky, D., Andrew, Y.N.: Learning syntactic patterns for automatic hypernym discovery. Advances in Neural Information Processing Systems 17 (2004)
27. Specia, L., Motta, E.: A Hybrid Approach for Relation Extraction Aimed at the Semantic Web. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 564–576. Springer, Heidelberg (2006)
28. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1), 26–34 (2001)
29. Staab, S., Maedche, A.: Ontology Learning for the Semantic Web. IEEE Intelligent Systems, Special Issue on the Semantic Web 16(2), 72–79 (2001)
30. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.: NeOn D5.1.1. NeOn Modelling Components. NeOn Project (2007)
31. Suárez-Figueroa, M.C., Aguado de Cea, G., Buil, C., Dellschaft, K., Fernández-López, M., García, A., Gómez-Pérez, A., Herrero, G., Montiel-Ponsoda, E., Sabou, M., Villazon-Terrazas, B., Yufei, Z.: NeOn D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks. NeOn project (2008)
32. Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., Hart, G.: A Comparison of three Controlled Natural Languages for OWL 1.1. In: OWLED (2008)
33. Xu, F., Kurz, D., Piskorski, J., Schmeier, S.: A Domain Adaptive Approach to Automatic Acquisition of Domain Relevant Terms and their Relations with Bootstrapping. In: Proc. of the 3rd LREC (2002)