International Journal on Semantic Web and Information Systems, 6(4), 27-63, October-December 2010 27

A Pattern-Based Method for Re-Engineering Non-Ontological Resources into Ontologies

Boris Villazón-Terrazas, Universidad Politécnica de Madrid, Spain

Mari Carmen Suárez-Figueroa, Universidad Politécnica de Madrid, Spain

Asunción Gómez-Pérez, Universidad Politécnica de Madrid, Spain

ABSTRACT

To speed up the ontology development process, ontology developers are reusing all available ontological and non-ontological resources, such as classification schemes, thesauri, lexicons, and so forth, that have already reached some consensus. Non-ontological resources are highly heterogeneous in their data model and storage system (or implementation). The reuse of these non-ontological resources involves their re-engineering into ontologies. This paper presents a method for re-engineering non-ontological resources into ontologies. The method is based on so-called re-engineering patterns, which define a procedure that transforms the non-ontological resource components into ontology representational primitives using WordNet for making explicit the relations among the non-ontological resource terms. The paper also provides the description of NOR$_2$O, a software library that implements the transformations suggested by the patterns. Finally, it depicts an evaluation of the method, patterns, and software library proposed.

Keywords: *Non-Ontological Resources, Ontologies, Ontology Development, Patterns, Re-Engineering*

INTRODUCTION

Research on Ontology Engineering methodologies has provided methods and techniques for developing ontologies from scratch. Well-recognized methodological approaches such as METHONTOLOGY (Gómez-Pérez, Fernández-López & Corcho, 2003), On-To-Knowledge (Staab, Schnurr, Studer & Sure, 2001), and DILIGENT (Pinto, Tempich & Staab, 2004) give guidelines that help researchers to develop ontologies. However, researchers face an important limitation: no guidelines are provided for building ontologies by re-engineering existing knowledge resources widely used by a particular community.

During the last decade, specific methods, techniques and tools were proposed for building ontologies from existing knowledge resources. First, ontology learning methods and tools have been proposed to extract relevant concepts and relations from structured, semi-structured, and non-structured resources (Gómez-Pérez & Man-

DOI: 10.4018/jswis.2010100102

zano-Macho, 2004; Maedche & Staab, 2001) in order to form a single ontology. One important constraint to these methods and tools is that they propose *ad-hoc* solutions to transforming such resources, mainly texts, into ontologies. Hepp et al. (Hepp, 2006; Hepp & Brujin, 2007; Hepp, 2007) stated that employing methods and techniques when ontologizing non-ontological resources to the level of ontologies is key for the success of semantic technology for two main reasons: (1) if the use of semantic technologies for real-world data integration challenges is required, it is possible to refer to the original conceptual elements, and (2) for many domains, the existing category systems, XML schemas, and normative entity identifiers are the most efficient resources for engineering ontologies.

The ontologization of non-ontological resources has led to the design of several specific methods, techniques and tools (Hepp & Brujin, 2007; Hyvönen, Viljanen, Tuominen & Seppälä, 2008; Gangemi, Guarino, Masolo & Oltramari, 2003; García & Celma, 2005). These are mainly specific to a particular resource type, or to a particular resource implementation. Thus, everytime ontology engineers face a new resource type or implementation, they develop *ad-hoc* solutions for transforming such resource into a single ontology.

In parallel, and within the context of the NeOn project[1], a novel scenario-based methodology for building ontology networks[2] has been proposed: the NeOn Methodology (Suárez-Figueroa, 2010; Gómez-Pérez & Suárez-Figueroa, 2009). One of these novel scenarios is *Building Ontology Networks by Reusing and Re-engineering Non-Ontological Resources*. As opposed to custom-building silos of single ontologies from scratch, this new scenario emphasizes the re-engineering of knowledge resources for building ontologies that are connected with other ontologies in the ontology network.

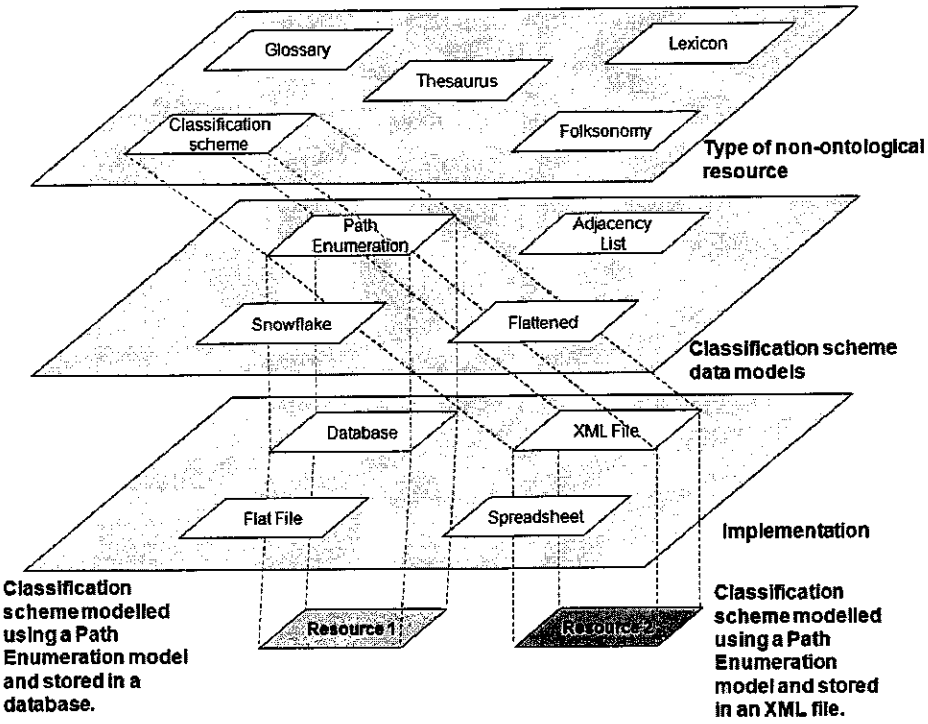
The motivation of this paper lies in this scenario of the NeOn Methodology and the use of re-engineering patterns to transform the non-ontological resources components into ontology representational primitives. Along this paper we will try to demonstrate that the use of re-engineering patterns for transforming non-ontological resources into ontologies has several advantages: (1) they embody expertise about how to guide a re-engineering process, (2) they improve the efficiency of the re-engineering process, and (3) they make the transformation process easier for ontology engineers.

In this paper we present first our proposed categorization of non-ontological resources. Then, we provide a framework for comparing methods for re-engineering non-ontological resources into ontologies, and the conclusions drawn on the comparative study. After that, we analyze the role of patterns in software engineering and ontology engineering with particular emphasis on re-engineering patterns. Then, we present our pattern-based method for re-engineering non-ontological resources into ontologies. Then, we explain NOR_2O, a software library that performs the transformation automatically. Next, we present an evaluation of the method, patterns and software library. Finally we draw the conclusions and provide future lines of work.

NON-ONTOLOGICAL RESOURCES

Non-Ontological Resources (NORs) are knowledge resources whose semantics have not yet been formalized by an ontology (García-Silva, Gómez-Pérez, Suárez-Figueroa & Villazón-Terrazas, 2008). There is a great number of NORs that embody knowledge about some particular domains and that represent some degree of consensus. These resources are present in the form of textual corpora, classification scheme, thesauri, lexicons, etc. NORs have usually implicit semantics that allows interpreting the knowledge they contain. Regardless of whether the semantics is explicit or not, the main problem is that the semantics of NORs is not always formalized, and this lack of formalization prevents them from being used as ontologies. Using non-ontological resources that already have reached a consensus for build-

Figure 1. NORs categorization



ing ontologies can have several benefits, e.g. interoperability in terms of the vocabulary used, browsing/searching information, decrease the knowledge acquisition bottleneck, and increase the reuse.

An analysis of the literature has revealed that there are different ways of categorizing NORs. Thus Maedche et al. (Maedche & Staab, 2001) classify NORs into unstructured (free text), semi-structured (folksonomies) and structured (databases) resources; whereas Gangemi et al. (Gangemi, Pisanelli & Steve, 1998) distinguish catalogues of normalized terms, glossed catalogues, and taxonomies; finally, Hodge (Hodge, 2000) proposes characteristics such as structure, complexity, relationships among terms, and historical functions for classifying them. However, an accepted and agreed on typology of NORs does not exist yet.

In this paper we propose a categorization of NORs, according to the three features presented in Figure 1: (1) type of NOR, which refers to the type of inner organization of the information; (2) data model[3], that is, the design data model used to represent the knowledge encoded by the resource; and (3) resource implementation.

According to the *type of NOR* we classify them into

- *Glossary*: A glossary is an alphabetical list of terms or words found in or relating to a specific topic or text. It may or may not include explanations, and its vocabulary may be monolingual, bilingual or multilingual (Wright & Budin, 1997). One example is the FAO Fisheries Glossary[4].
- *Classification scheme*: A classification scheme refers to the descriptive information required to arrange or divide objects into groups according to the characteristics that they have in common (ISO/IEC, 2004). A clear example is the Fishery Interna-

tional Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP)[5].

- *Thesaurus*: A thesaurus is a controlled vocabulary of terms in a particular domain with hierarchical, associative, and equivalence relations between terms. Thesauri are mainly used for indexing and retrieving articles in large databases (ISO, 1986). A good example is the AGROVOC[6] thesaurus.
- *Lexicon*: In a restricted sense, a computational lexicon is considered as a list of words or lexemes hierarchically organized and normally accompanied by meaning and linguistic behaviour information (Hirst, 2004). A relevant example is WordNet[7].
- *Folksonomy*: A folksonomy is the result of personal free tagging of information and objects (anything with a URL) for one's own retrieval. The tagging is done in a social environment (usually shared and open to others). A folksonomy is created from the act of tagging by the person consuming the information[8]. An example is del.icio.us[9].

According to the *data model*, there are different ways of representing the knowledge encoded by the resource. Next we present several *data models for classification schemes*, which are shown in Figure 2.

- *Path Enumeration* (Brandon, 2005): A path enumeration model (see Figure 2b) is a recursive structure for hierarchy representations defined as a model that stores for each node the path (as a string) from the root to the node. This string is the concatenation of the nodes code in the path from the root to the node.
- *Adjacency List* (Brandon, 2005): An adjacency list model is a recursive structure for hierarchy representations comprising a list of nodes with a column linking to their parent nodes. Figure 2c shows this model.
- *Snowflake* (Malinowski & Zimányi, 2006): A snowflake model is a normalized structure for hierarchy representations. For each hierarchy level a table is created. In this model each hierarchy node has a linked column to its parent node. Figure 2d shows this model.
- *Flattened* (Malinowski & Zimányi, 2006): A flattened model is a denormalized structure for hierarchy representations. The hierarchy is represented with a table in which each hierarchy level is stored on a different column. Figure 2e illustrates this model.

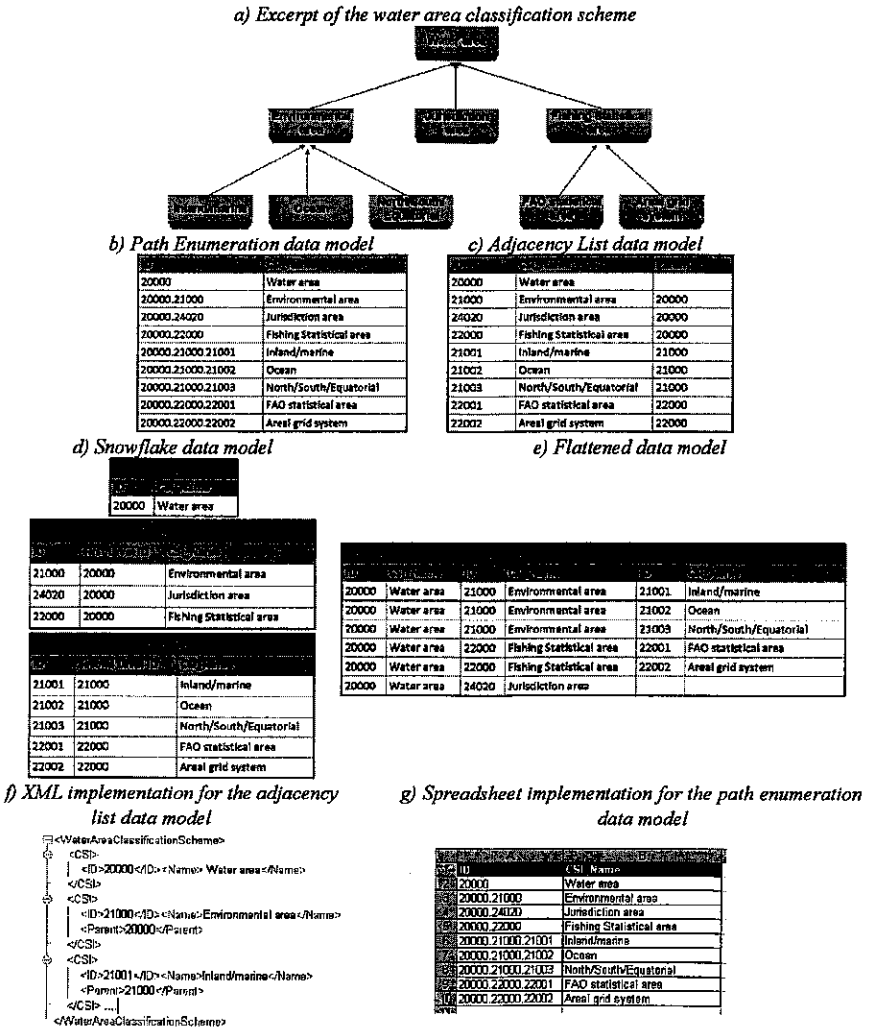
Next, we present two *data models for thesauri*:

- *Record-based model* (Soergel, 1995): A record-based model is a denormalized structure that uses a record for every term. The record keeps information about the term, such as synonyms, and broader, narrower and related terms. This model looks like the flattened model of the classification scheme.
- *Relation-based model* (Soergel, 1995): A relation-based model leads to a more elegant and efficient structure. Information is stored in individual pieces that can be arranged in different ways. Relationship types are not defined as fields in a record, they are simply data values in a relationship record, thus new relationship types can be introduced with ease. There are three entities: (1) a term entity, which contains the overall set of terms; (2) a term-term relationship entity, in which each record contains two different term codes and the relationship between them; and (3) a relationship source entity, which contains the overall resource relationships.

After analysing several *data models for lexicons*, we have identified the same data models already identified for thesauri:

- *Record-based model* (Soergel, 1995): This model can also be used for lexicons, because it is possible to use a record for

Figure 2. Classification scheme example



every lexical resource and for the information about that lexical resource.

- Relation-based model (Soergel, 1995): It can also be used for lexicons, because it is possible to store the information about any lexicon in individual pieces.

According to the *implementation* we classify NORs into databases, spreadsheets, XML files and flat files.

To sum up, Figure 1 shows how a given type of NOR can be modelled following one or more data models, each of which could be implemented in different ways at the implementation layer. As an example, Figure 1 shows a classification scheme modeled following a path enumeration model. In this case, the classification scheme is implemented in a database and in an XML file.

To exemplify the non-ontological categorization here presented with a real life classifica-

tion scheme, we use an excerpt from the FAO water area classification shown in Figure 2a. This classification schema is modelled following a path enumeration model (Figure 2b), an adjacency list model (Figure 2c), a snowflake model (Figure 2d), and a flattened model (Figure 2e). Figure 2f presents an XML implementation of the path enumeration model and Figure 2g presents a spreadsheet implementation of the path enumeration model of the same classification scheme.

It is worth mentioning that this first categorization of NORs is neither exhaustive nor complete. Currently, we are enriching it by adding examples taken from RosettaNet[10] and Electronic Data Interchange, EDI[11].

Moreover, we can map available non-ontological resources to our categorization. In the following we present a brief list of them.

- The United Nations Standard Products and Services Code, UNSPSC[12], is a classification scheme, modelled with the path enumeration data model, and it is stored in a relational database.
- WordNet[13], a lexical database for English, is a lexicon, modelled with the relation-based data model, and it is stored in several implementations, a particular implementation is a relational database.
- UMLS[14], a very large, multi-purpose, and multilingual thesaurus that contains information about biomedical and health related concepts, is modelled with the record-based model and it is stored in a flat file.
- MeSh[15], the Medical Subject Headings, is a classification scheme, modelled with the path enumeration data model.
- The Art and Architecture Thesaurus[16], is modelled with the record-based data model and it is implemented in XML.
- The ISCO-08 International Standard Classification of Occupations[17] is a classification scheme modelled with the path enumeration data model, and implemented in a database and spreadsheet.
- The European Training Thesaurus, ETT[18], is modelled with the record-based data model and it is implemented in XML.
- The Classification of Fields of Education and Training, FOET[19], is a classification scheme modelled with path enumeration data model, and implemented in XML and spreadsheet.
- The Aquatic Sciences and Fisheries Abstracts thesaurus, ASFA[20], is modelled with the record-based data model and it is implemented in XML.
- The AGROVOC thesaurus[21] is modelled with the relation-based data model and implemented in a database.
- The Fisheries Global Information System, FIGIS[22] is modelled with the adjacency list data model and implemented in a database.
- The Classification of Italian Education Titles published by the National Institute of Statistics, ISTAT[23], is a classification scheme modelled with the flattened data model and implemented in a spreadsheet.

A COMPARATIVE FRAMEWORK FOR RE-ENGINEERING NORS INTO ONTOLOGIES

In this section we provide an overview of a comparative study of the most outstanding methods for re-engineering NORs into ontologies. First, we describe briefly the representative methods. Then, we have established a common framework with which to compare the main characteristics of the different methods. Finally, we evaluate the methods against the proposed framework.

Methods for Transforming NORs into Ontologies

This section overviews existing methods for transforming NORs into ontologies. The methods are centred on the NOR type (classification schemes, lexica, and thesauri) and on the NOR implementation (databases, XML, flat files, and spreadsheets).

Methods Centred on the NOR Type

- **GenTax** is a method presented by Hepp et al. (Hepp & Brujin, 2007) for semi-automatically deriving consistent RDF(S) and OWL ontologies from hierarchical classifications, thesauri and informal taxonomies. These authors have implemented SKOS-2GenTax to support their method. Human intervention in the transformation is limited to checking some conceptual properties and to identifying frequent anomalies. The idea underlying this method is to derive two ontology classes: one generic concept and one broader taxonomic concept from each category. The method produces one single ontology in OWL-DLP or RDF(S). The ontology components generated are classes and relations.
- **Hakkarainen et al.** (Hakkarainen, Hella, Strasunskas & Tuxen, 2006) present a study of the semantic relationship between the ISO 15926-2[24] and OWL-DL. The ISO 15926-2 is built on EXPRESS[25]; and stored in a flat file to specify its data model. This method consists of (a) two transformation protocols, which are based on transformation rules; and (b) two inverse transformation protocols for examining the possible loss of semantics. Transformation protocols include a formal specification of the conversions and their functions are to manage a single ontology, expressed in OWL-DL. The ontology components generated are classes, attributes, and relations.
- **van Assem et al.** (Assem, Gangemi & Schreiber, 2006) propose a method for a standard conversion of WordNet into the RDF/OWL representation language. This method is based on version 2.0 of Princenton's WordNet Prolog distribution. The process for designing the conversion consists in (1) analyzing the existing conversions, which helps to understand the different ways in which WordNet is used on the Semantic Web; (2) formulating the requirements; (3) analyzing the source files and documentation; (4) designing the RDF/OWL schema; (5) designing a program for converting Prolog data to RDF/OWL; (6) drafting a Working Group Note explaining the requirements and design choices; and (7) reviewing the draft note and schema/data fields. The method produces one single ontology in RDF(S)/OWL Full. The ontology components generated are classes, attributes, relations, and instances.
- **Gangemi et al.** (Gangemi, Navigli & Velardi, 2003; Gangemi, Guarino, Masolo & Oltramari, 2003) present a method that explains how WordNet information can be bootstrapped, mapped, refined and modularized. Their method employs WordNet 1.6, stored in relational databases. The method automatically extracts association relations from WordNet, and interprets those associations in terms of a set of conceptual relations, formally defined in the DOLCE[26] ontology. It manages a single ontology implemented in DAML+OIL. The ontology components generated are classes, attributes, and relations.
- **Hahn et al.** present (Hahn, 2003; Hahn & Schulz, 2003) a method that extracts conceptual knowledge from UMLS[27], and semi-automatically converts this conceptual knowledge into a formal description logics model in LOOM[28]. It produces a single ontology. The ontology components generated are classes and relations.
- **van Assem et al.** present (Assem, Menken, Schreiber & Wielemaker, 2004) a method for converting thesauri from their native format to RDF(S)/OWL Full. This method deals with resources implemented in (1) a proprietary text format, (2) a relational database, and (3) an XML representation. It produces one single ontology in RDF(S)/OWL Full. The ontology components generated are classes, attributes, and relations.
- **van Assem et al.** present (Assem, Malaisé, Miles & Schreiber, 2006) a method for converting thesauri to the SKOS[29] RDF/OWL schema. This SKOS schema is a proposal for a standard that is being developed by the W3Cs Semantic Web Best Practices

Working Group. The method produces one single ontology expressed in SKOS RDF. The ontology components generated are classes, attributes, and relations.

- **Wielinga et al.** present (Wielinga, Schreiber, Wielemaker & Sandberg, 2001) a method for transforming the Art and Architecture Thesaurus (AAT) into an RDF(S) ontology. The AAT is available in XML files. The method produces one ontology. The ontology components generated are classes, attributes, and relations.
- **Hyvönen et al.** present (Hyvönen et al., 2008) a method for transforming thesauri into ontologies. This method has been applied to the YSA thesaurus[30] where DOLCE was employed for the transformation. The resultant ontology, based on the YSA thesaurus, is the General Finnish Ontology YSO[31]. The ontology components generated are classes, attributes, and relations, which are expressed in RDF(S).
- **Soergel et al.** (Soergel et al., 2004), and **Lauser et al.** (Lauser & Sini, 2006) present a method for the re-engineering of traditional thesaurus, AGROVOC, into a full-fledged ontology. The AGROVOC thesaurus is stored in a database. The authors plan to build an inventory of patterns, namely, content ontology design patterns which are specific for the agricultural domain. The method produces one ontology in OWL-DL, and the ontology components generated are classes, attributes, and relations.

Methods Centered on the NOR Implementation

- **Stojanovic et al.** present (Stojanovic, Stojanovic & Volz, 2002) an integrated and semi-automatic approach to generating shared and understandable metadata of data-intensive Web applications. This method is based on mapping a relational schema into F-Logic ontology by means of a reverse engineering process. The method deals with any NORs stored in a database and transforms the database content into instances of an existing ontology (in the form of RDF files) by applying the generic mapping rules specified by the authors.
- **Barrasa et al.** present (Barrasa, 2007; Barrasa, Corcho & Gómez-Pérez, 2004) an integrated framework for the formal specification, evaluation and exploitation of the semantic correspondences between legacy ontologies and legacy relational data sources. The framework consists of two main components: R_2O, which is a declarative language for the description of complex mapping expressions between ontology elements and relational elements, and ODEMapster processor, which generates RDF instances from relational instances based on the mapping description expressed in an R_2O document.
- **García et al.** (García & Celma, 2005) introduce a method to create an ontology from the XML schema and populate it with instances generated from the XML data. This method has been applied to the MPEG-7[32] XML Schemas and has generated a single MPEG-7 ontology[33] in OWL Full. The ontology components generated are classes, attributes, relations, and instances.
- **An et al.** present (An & Mylopoulos, 2005) a method to translate an XML web document into an instance of an OWL-DL ontology. In that work the authors take advantage of the semi-automatic mapping discovery tool (An, Borgida & Mylopoulos, 2005) for discovering the relationship between XML schema and the ontology. The method produces a single ontology in OWL-DL. The ontology components generated are classes, attributes, relations, and instances.
- **Cruz et al.** present (Cruz, Xiao & Hsu, 2004) a method to transform XML schema into RDF(S) ontology while preserving the XML document structure, i.e., modelling the knowledge implicit in XML schema with RDF(S). In order to support the method, a specific tool has been developed. This method produces a single ontology.

The ontology components generated are classes, attributes, and relations.

- **Foxvog et al.** present (Foxvog & Bussler, 2006) a method to transform Electronic Data Interchange (EDI)[34] messages into ontologies. The method produces several ontologies. The ontology components generated are classes, attributes, relations, and instances, expressed in OWL Full, CycL, and WSML.

Evaluation Framework

In this section we establish a framework for comparing the methods for re-engineering NORs. Next, we present the characteristics identified, which are grouped according to the NOR, the transformation process, and the resultant ontology.

NOR Characteristics

- *Types* of NOR: (1) classification schemes, (2) folksonomies, (3) glossaries, (4) lexica, and (5) thesauri.
- *Implementation* of a NOR: (1) databases, (2) XML files, (3) flat files, or (4) spreadsheets.
- A NOR belongs to a specific and concrete domain or it can fit in any domain.
- The NOR *data model* information is known. The data model depicts the logical entity types, the data attributes describing those entities, and the relationships between entities (Carkenord, 2002).

Characteristics of the Transformation Process

- The transformation approach can be: (1) an *ABox transformation* (Caracciolo, Heguiabehere, Presutti & Gangemi, 2009), which converts the resource schema into an ontology schema and the resource content into ontology instances; (2) an *TBox transformation* (Caracciolo et al., 2009), which transforms the resource content into an ontology schema; or (3) *Population*, for transforming the resource content into instances of an existing ontology.
- The transformation process can be (1) *automatic*, (2) *semi-automatic*, or (3) *manual*.
- The transformation process considers the *implicit, formal semantics of the NOR relationships* (*subClassOf*, *partOf*, etc.).
- The transformation process uses *additional resources* to carry out the conversion.
- The research work provides some *methodological guidelines* to support the transformation process.
- The list of the *techniques employed* serves to guide the transformation process, e.g., mapping rules, re-engineering patterns.
- If a specific *tool* is provided, then this should give technological support to the transformation process.

Characteristics of the resultant ontology

- The *ontology components* generated are classes, attributes, relations, or instances.
- The *ontology implementation language*: for instance OWL or RDF(S).
- The research work generates a *single ontology* or *several ontologies*. We do not distinguish whether the ontologies generated are interconnected or not.

Results

After having analyzed the state of the art of the methods available for re-engineering NORs, we present the results of applying the evaluation framework.

Table 1 summarizes the methods presented according to the characteristics of the NOR.

- According to the type of non-ontological resource, we can state that most of the methods are focused on thesauri, some on classification schemes, lexicons and folksonomies, and then there is a small group which does not contemplate the type of resource. Only one method is focused in thesauri and classification schemes. In

Table 1. NOR characteristics of the methods

Research work	Type of resource	Resource implemented in	Specific/Any	Data model is used
Hepp et al.	Classification scheme, thesaurus	Database	Any	No
Hakkarainen et al.	Classification scheme	Flat file	ISO15926-2	Yes
Abbasi et al.	Folksonomy	Not mentioned	Any	No
Maala et al.	Folksonomy	Not mentioned	Flickr	No
van Assem et al.	Lexicon	Prolog	Any	Yes
Gangemi et al.	Lexicon	Database	Any	Yes
Hahn et al.	Thesaurus	ASCII files	UMLS	Yes
van Assem et al.	Thesaurus	Proprietary text format, re-lational database, XML	Any	No
van Assem et al.	Thesaurus	Not mentioned	IPSV, GTAA, MeSH	No
Wielinga et al.	Thesaurus	XML	AAT	Yes
Hyvo¨nen et al.	Thesaurus	Not mentioned	YSA	Yes
Soergel et al.	Thesaurus	Database	AGROVOC	Yes
Stojanovic et al.	Not mentioned	Database	Any	Yes
Barrasa et al.	Not mentioned	Database	Any	Yes
Garc´ıa et al.	Not mentioned	XML	Any	No
An et al.	Not mentioned	XML	Any	No
Cruz et al.	Not mentioned	XML	Any	No
Foxvog et al.	Not mentioned	Flat file	EDI X12	No

general the methods consider only a particular type of resource.

- According to the implementation of the non-ontological resource, we can state that most of the methods are focused on databases, some on XML, and flat files, and some are independent of resource implementation. In addition, one method is focused on resources implemented in Prolog whereas another includes resources implemented in proprietary format, relational database, and XML.
- In relation to the data model, we can observe the half of the methods does not contemplate the data model of the resource for the transformation, whereas half does it.

To sum up, we can conclude that most of the methods presented are based on *ad-hoc* transformations for the resource type, and the resource implementation. Only a few take advantage of the resource data model, an important artifact for the re-engineering process (García-Silva et al., 2008). There is no any integrated framework, method or corresponding tool, that considers the resources types, data models and implementations identified in an unified way. In conclusion, we can state that there is a clear

need for some sort of re-engineering methods that simultaneously

- Cope with the overall set of non-ontological resources, i.e. classification schemes, thesauri, and lexica.
- Consider the internal data model of the resource.
- Cope with non-ontological resources implemented in databases, XML files, flat files, or spreadsheets.
- According to the explicitation of the hidden semantics in the relations of the resource components, we can state that the methods that perform a TBox transformation make explicit the semantics in the relations of the resource components. Most of those methods identify *subClassOf* relations, others identify *ad-hoc* relations, and some identify *partOf* relations. However, only a few methods make explicit the three types of relations.
- With regard to the transformation approach (Table 2), the majority of the methods perform a TBox transformation, many others perform an ABox transformation and some perform a population. However, no method includes the possibility to perform the three transformation approaches.
- Regarding to the degree of automation, almost all the methods perform a semi-automatic transformation of the resource, followed by three automatic methods, and then by a manual method.
- With respect to how the methods make explicit the hidden semantics in the relations of the resource terms, we can say that three methods rely on the domain expert for making explicit the semantics, and two rely on an external resource, e.g., DOLCE ontology. Moreover, there are two methods that rely on external resources but not for making explicit the hidden semantics, but for finding out a proper ontology for populating it.
- According to the provision of the methodological guidelines, almost all the methods provide methodological guidelines for the transformation. However these guidelines are not finely detailed; for instance, they do not provide information about who is in charge of performing a particular activity/task, nor when that activity/task has to be carried out.
- With regard to the techniques employed, most of the methods do not mention them at all. Only a few methods specify techniques as transformation rules, lexico-syntactic patterns, mapping rules and natural language techniques.
- According to the tool support, most of the methods rely on ad-hoc tools for the transformation. And only a few methods integrate a public available tool, such as KAON-REVERSE, ODEMapster, XSD2OWL, and XML2RDF.

In summary, after having analyzed the features related to the transformation process, we can conclude that (1) methods are mostly focused on the TBox transformation approach; (2) only a few methods make explicit the hidden semantics in the relations of the NOR terms, and most of them rely on domain expert for doing it; (3) almost all the methods provide methodological guidelines for the transformation, but they are not finely detailed; (4) only a few methods specify the techniques employed for the transformation; and (5) there is no method that considers the possibility to perform the three transformation approaches. In a nutshell, we can state that there is a clear need for some sort of re-engineering methods that

- Include the three transformation approaches (TBox, ABox and Population).
- Make explicit the hidden semantics in the relations of the NOR terms, by means of external resources in a semi-automatic way, for saving the transformation time.
- Provide fully detailed guidelines for the transformation, including information on who is in charge of performing a particular activity/task and when this activity/task has to be carried out.

Table 2. Transformation process of the methods

Research work	Transformation Approach	Automatic/ Semi-automatic/ Manual	Semantics of NOR relations	Additional Resources	Methodological Guidelines	Technique	Tool support
Hepp et al.	TBox	Semi-automatic	subClassOf, ad-hoc relation	No	Yes	Not mentioned	SKOS-2GenTax
Hakkarainen et al.	ABox	Semi-automatic	subClassOf, ad-hoc relation	No	Yes	Transformation rules	Not mentioned
Abbasi et al.	Population	Automatic	Not mentioned	Swoogle Google	Yes	Lexico Syntactic Patterns	T-ORG
Maala et al.	Population	Automatic	Not mentioned	WordNet	Yes	Not mentioned	Not mentioned
van Assem et al.	ABox	Semi-automatic	Not mentioned	No	Yes	Not mentioned	Swi-Prolog
Gangemi et al.	TBox	Semi-automatic	Not mentioned	DOLCE	Yes	NLP Techniques	Not mentioned
Hahn et al.	TBox	Semi-automatic	subClassOf, partOf, ad-hoc relation	No	Yes	Ontology Design Patterns	Ad-hoc tool
van Assem et al.	TBox	Semi-automatic	subClassOf, ad-hoc relation	No	Yes	Not mentioned	Ad-hoc tool
van Assem et al.	TBox	Automatic	Not mentioned	No	Yes	Not mentioned	Swi-Prolog
Wielinga et al.	TBox	Semi-automatic	Not mentioned	No	Yes	Not mentioned	Ad-hoc tool
Hyvo¨nen et al.	TBox	Semi-automatic	Not mentioned	DOLCE	Yes	Not mentioned	Ad-hoc tool
Soergel et al.	TBox	Manual	subClassOf, ad-hoc relation	No	Yes	Not mentioned	Not mentioned
Stojanovic et al.	Population	Semi-automatic	ad-hoc relation	No	Yes	Mapping rules	KAON-REVERSE
Barrasa et al.	Population	Semi-automatic	subClassOf, ad-hoc relation	No	Yes	Mapping rules	ODE-Mapster
Garc´ıa et al.	ABox	Semi-automatic	ad-hoc relation	No	Yes	Mapping rules	XS-D2OWL XML-2RDF
An et al.	ABox	Semi-automatic	ad-hoc relation	No	No	Not mentioned	Discovery tool
Cruz et al.	ABox	Semi-automatic	Not mentioned	No	Yes	Mapping rules	Ad-hoc tool

continued on following page

Table 2. continued

Research work	Transformation Approach	Automatic/ Semi-automatic/ Manual	Semantics of NOR relations	Additional Resources	Methodological Guidelines	Technique	Tool support
Foxvog et al.	ABox	Semi-automatic	Not mentioned	No	Yes	Not mentioned	Ad-hoc tool

- Integrate in a single framework the method and its corresponding tool support for the transformation.
- Employ techniques that improve the efficiency of the re-engineering process.

Table 3 summarizes the methods presented regarding the characteristics of the resultant ontology.

- In relation to the ontology components, we can observe that this feature is closely related to the transformation approach performed by the methods. Methods that perform TBox transformation generate classes, relations, and optionally attributes. Methods that perform ABox transformation generate classes, attributes, relations and instances. Methods that perform population generate instances.
- As for the ontology implementation language, despite of the variety of existing languages, the ontology languages mostly used are OWL for the ontology and RDF for the instances.
- As for whether the method generates one or several ontologies, almost all the methods generate a single ontology.

After having analyzed the characteristics related to the resultant ontology, we can conclude that there is a lack of re-engineering methods that support several ontologies, and the generation of ontologies that includes classes, attributes, relations and instances.

In this paper, we solve the needs above mentioned introduce a method that

- Includes the three transformation approaches (TBox, ABox and Population).
- Makes explicit the hidden semantics in the relations of the NOR terms, by means of external resources in an semi-automatic way, for saving the transformation time.
- Provides fully detailed guidelines for the transformation, including information on who is in charge of performing a particular activity/task and when this activity/task has to be carried out.
- Employs techniques that improve the efficiency of the re-engineering process.

PATTERNS FOR RE-ENGINEERING

In this section we introduce briefly the role that patterns play in software and ontology engineering, with a particular focus on re-engineering patterns.

Patterns were introduced by Christopher Alexander (Alexander, 1979) to encode knowledge and experience in designing buildings. He defines a pattern as the core of a solution to a problem in context. The solution can be applied in different situations and has to be adapted to fit the needs of the specific situation (Alexander, 1979).

In the (Object Oriented) software community, patterns are used to describe software design structures that can be used over and over again in different systems. Patterns provide a general solution that has to be applied in a particular context, where design considerations are used to decide whether the pattern is useful and how it could be best implemented (Edwards, Puckett & Jolly, 2006).

Table 3. Ontology characteristics of the methods

Research Work	Components	Implementation language	Single/Several
Hepp et al.	classes, relations	RDF(S) / OWL-DLP	Single
Hakkarainen et al.	classes, attributes, relations	OWL-DL	Single
Abbasi et al.	instances	Not mentioned	Several
Maala et al.	instances	RDF	Single
van Assem et al.	classes, attributes, relations, instances	RDF(S) / OWL Full	Single
Gangemi et al.	classes, attributes, relations, instances	DAML+OIL	Single
Hahn et al.	classes, relations	LOOM / ALC	Single
van Assem et al.	classes, attributes, relations	RDF(S) / OWL Full	Single
van Assem et al.	classes, attributes, relations	SKOS RDF	Single
Wielinga et al.	classes, attributes, relations	RDF(S)	Single
Hyvo¨ nen et al.	classes, attributes, relations	RDF(S)	Single
Soergel et al.	classes, attributes, relations	OWL-DL	Single
Stojanovic et al.	instances	F-Logic / RDF	Single
Barrasa et al.	instances	RDF	Single
Garc´ıa et al.	classes, attributes, relations, instances	OWL Full/ RDF	Single
An et al.	classes, attributes, relations, instances	OWL-DL	Single
Cruz et al.	classes, attributes, relations	RDF(S)	Single
Foxvog et al.	classes, attributes, relations, instances	CycL / OWL Full / WSML	Several

A kind of software patterns are the re-engineering software patterns (Pooley & Stevens, 1998). These describe how to change a legacy system into a new, refactored system that fits current conditions and requirements. Their main goal is to offer a solution for re-engineering problems. They are also on a specific level of abstraction and describe a process of re-engineering without proposing a complete methodology; sometimes these re-engineering patterns can suggest which type of tool should be used.

The idea of applying patterns for modelling ontologies was proposed by (Clark, Thompson & Porter, 2000). Since then, several relevant works on patterns have appeared, such as: Semantic Web Best Practices and Deployment Working Group[35], the Ontology Design Patterns Public Catalog[36], the Ontology Design Patterns (ODP) Portal[37] and the Linked Data Patterns[38], which is a catalogue of Linked Data (Bizer, 2009) patterns. According to (Presutti et al., 2008) Ontology Design Patterns are modelling solutions to solving a recurrent ontology design problem. They distinguish different types of Ontology Design Patterns by grouping them into six families (see first level in Figure 3). Each family addresses different kind of problems and can be represented with different levels of formality. For a detailed description of each family of patterns, please refer to (Presutti et al., 2008).

As shown in Figure 3, *Correspondence OPs* are templates for representing alignments

between models. They include Schema Re-engineering OPs, Re-engineering OPs and Alignment OPs. *Re-engineering OPs* are transformation rules applied to create a new ontology starting from elements of a source model, and *Refactoring OPs*.

In this paper we focus on *Patterns for Re-engineering NORs (PR-NOR)*. These patterns define a procedure that transforms the NOR components into ontology representational primitives. PR-NORs are considered as Re-engineering Patterns (Presutti et al., 2008), because they share the same goal, i.e., to generate a new ontology from elements of a source model. The rest of the patterns are out of the scope of this paper.

In this paper we propose the use of re-engineering patterns for transforming NORs into ontologies. The re-engineering patterns proposed take advantage of the use of logical patterns for creating the ontology code. So, most of the code generated follows the best practices already identified by the community. Up to present, none of the methods presented in the evaluation framework section use re-engineering patterns as a technique for the transformation.

PATTERN-BASED RE-ENGINEERING METHOD

In this section we present the method we have devised for re-engineering non-ontological resources into ontologies. The method is based on a model for re-engineering non-ontological resources. In it, we first provide a description of this re-engineering model for NORs, then we introduce the notion of patterns for re-engineering NORs. Finally, we depict the methodological guidelines for re-engineering NORs into ontologies.

Re-Engineering Model for NORs

Our model for NOR re-engineering, depicted in Figure 4, is based on the software re-engineering[39] model presented in (Byrne, 1992). The figure also shows some activities that can be defined as follows (Suárez-Figueroa & Gómez-Pérez, 2008):

- NOR reverse engineering is defined as the activity of analyzing a non-ontological resource to identify its underlying components and to create a representation of the resource at higher levels of abstraction.
- NOR transformation is defined as the activity of generating an ontological model at different levels of abstraction from the NOR.
- Ontology forward engineering refers to the activity of outputting a new implementation of the ontology on the basis of the new conceptual model.

Since we consider NORs as a kind of software resources, we use the software abstraction levels shown in Figure 4 to depict the reverse engineering of the NOR. Understanding how

Figure 3. Ontology Design Pattern Categorization (Presutti et al., 2008)

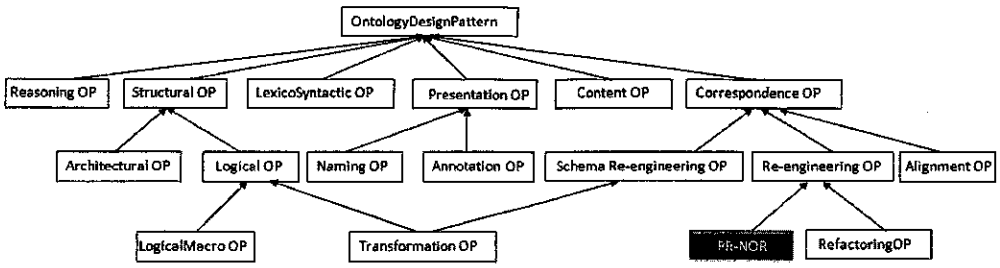
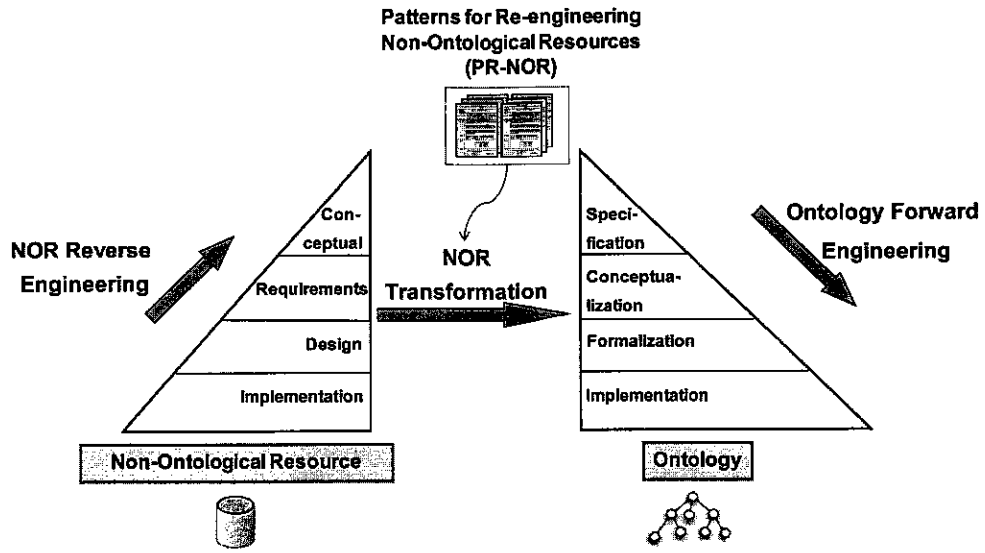


Figure 4. Re-engineering model for NORs



NOR was created is extremely useful for understanding how NOR can be re-engineered.

In the triangle on the left in Figure 4 we can distinguish four software abstraction levels:

- *The conceptual abstraction level*, which describes in general terms the functional characteristics of the NORs.
- *The requirements level*, which is the specification of the problem being solved.
- *The design level*, which is the specification of the solution.
- *The implementation level*, which refers to the coding, testing and delivering of the operational resource.

As the level of abstraction decreases, a resource description becomes more detailed and the amount of information increases. Moreover, the higher the abstraction level, the less information about a resource.

Since the goal, of the *forward engineering process,* is to transform the resources into ontologies, we use the ontology levels of abstraction (Gómez-Pérez, Fernández-López & Corcho, 2003) to depict the ontology forward engineering. In the triangle on the right in Figure 4 we can distinguish the four ontology abstraction levels that define each activity in ontology engineering:

- *The specification level*, which describes the collection of requirements that the ontology should fulfil.
- *The conceptualization level*, which organizes the information from the acquisition process into meaningful conceptual models.
- *The formalization level*, which represents the transformation of the conceptual model into a formal or semi-computable model according to a knowledge representation paradigm.
- *The implementation level*, which refers to the generation of computable models according to the syntax of a formal representation language.

Finally, the model in Figure 4 suggests the path from the existing NOR to the target ontology. This transformation is guided by a set of Patterns for Re-engineering NORs (PR-NOR), and goes from the NOR requirements/

design level to the conceptualization level of the ontology.

Requirements for the Transformation

In this section we describe the identified requirements for the transformation, we list the requirements according to the three transformation approaches identified in the Evaluation Framework section.

- *TBox transformation* (Caracciolo et al., 2009), that transforms the resource content into an ontology schema. This transformation approach tries to enforce a formal semantics to the re-engineered resources, even at the cost of changing their structure. The requirements for this transformation are:
  - Full conversion, the resultant ontology has all the information that is present in the original resource. In other words, all queries that are possible on the original source should be also possible on the ontology generated.
  - Conversion on the semantic level, which implies the schema translation to interpret the semantics of the data. In other words, the conversion should not stay agnostic to possible interpretations, e.g., relations among the NOR entities.
- *ABox transformation* (Caracciolo et al., 2009), that transforms the resource schema into an ontology schema, and the resource content into ontology instances. This transformation approach leaves the informal semantics of the re-engineered resources mostly untouched. The requirements for this transformation are:
  - Full conversion, the same requirement of the TBox transformation. Again, this implies that all queries that are possible on the original source should be also possible on the ontological version.
  - Structure preserving translation, the opposite of the second requirement of the TBox transformation. The translation should as much as possible reflect the original structure of the resource, in other words, the conversion should stay agnostic to possible interpretations.
- *Population* that transforms the resource content into instances of an existing ontology. The requirements of the transformation are:
  - Full conversion, the same requirement of the TBox and ABox transformation.
  - The ontology instances generated should reflect the target ontology structure as closely as possible. In this case, the class structure of the ontology already exists and is extended with instance data. In other words, the ontology instances must conform the existing ontology schema.

Patterns for Re-Engineering NORs

In this section we present the patterns that perform the transformations of NORs into ontologies. Patterns for re-engineering NORs (PR-NOR) define a procedure that transforms the NOR components into ontology representational primitives (García-Silva et al., 2008).

According to the NOR categorization presented in a previous section, the data model can be different even for the same type of NOR. For every data model we can define a process with a well-defined sequence of activities with which to extract the NORs components and then to map these components to a conceptual model of an ontology. Each of these processes can be expressed as a pattern for re-engineering NORs.

The re-engineering patterns take advantage of the use of Ontology Design Patterns[40] for creating the ontology code. Therefore, most of the code generated follows the best practices already identified by the community.

Template for the PR-NOR

In this section we present the template proposed, which describes the patterns for re-engineering NORs (PR-NOR). In order to present the patterns for re-engineering NORs we have modified the tabular template used in (García-Silva et al., 2008). The template and the meaning of each field are shown in Table 4.

Although we have identified five types of NORs, here we just list patterns for re-engineering three: classification schemes, thesauri, and lexica (see Table 5). It is worth pointing out that we refer to ontology schema as the TBox, and to ontology as the TBox and the ABox. These patterns are included in the ODP Portal. Regarding the time complexity of the algorithm included in the patterns, we can say that the time complexity of the TBox transformation algorithm is polynomial $O(n^2)$, and of the ABox transformation algorithm is linear $O(n)$.

Due to space reasons, Table 6 presents only an excerpt of the PR-NOR-CLTX-01 pattern which is instantiated to the classification scheme presented in Figure 2.

Semantics of the Relations between the NOR terms

The TBox transformation approach converts the resource content into an ontology schema. The TBox transformation tries to enforce a formal semantics to the resource, by making explicit the semantics hidden in the relations of the NOR terms. To this end, each NOR term is mapped to a class, and then, the semantics of the relations among those entities must be discovered and then made explicit. Thus, patterns that follow the TBox transformation approach must discover first the semantics of the relations among the NOR terms. To perform this task, we rely on WordNet, which organizes the lexical information into meanings (senses) and synsets. What makes WordNet remarkable is the existence of various relations explicitly declared between the word forms (e.g. lexical relations, such as synonymy and antonymy) and the synsets (meaning to meaning or semantic relations e.g. hyponymy/hypernymy relation, meronymy relation). In this paper, we want to prove that we can rely on an external resource for making explicit the relations. For this purpose, we rely first on WordNet, but for a future work we may rely on other information resources, such as DBpedia[41].

Algorithm 1 describes how to make explicit the semantics of the relations in the NOR terms. The shortname of the algorithm is *getRelation*.

The most important parts of algorithm are explained:

- (Line 1) Take two related terms from the NOR.
- (Line 2) For the *userDefinedRelation* one recommendation is to use the *subClassOf* relation by default. However, we recommend considering the type of non-ontological resource and the source relation. For instance, if the input terms come from a classification scheme from the classification scheme item relation, we recommend using the *subClassOf* relation by default. If the input terms come from a thesaurus (1) of the BT/NT relation, we recommend to use the *subClassOf* relation by default, and (2) of the RT relation, we recommend using the *relatedTerm* relation by default.
- (Lines 3-6) Check whether it is possible to get the *subClassOf* relation by identifying attributive adjetives[42] within the two terms.
- (Line 7) If it is not possible to get the *subClassOf* relation.
  - (Line 8) Search in WordNet for a relation between those two terms.
    - (Line 9-10) the hyponym in the relation is interpreted as *subClassOf*.
    - (Line 11-12) the hypernym in the relation is interpreted as *superClass*.
    - (Line 13-14) the member meronym in the relation is interpreted as *Part*.

Table 4. PR-NOR Template

Slot	Value
General Information	
Name	Name of the pattern.
Identifier	An acronym composed of component type + abbreviated name of the component + number
Component Type	Pattern for Re-engineering Non-Ontological Resource (PR-NOR).
Use Case	
General	Description in natural language of the re-engineering problem addressed by the pattern for re-engineering NORs.
Example	Description in natural language of an example of the re-engineering problem.
Pattern for Re-engineering Non-Ontological Resource	
INPUT: Resource to be Re-engineered	
General	Description in natural language of the NOR.
Example	Description in natural language of an example of the NOR.
Graphical Representation	
General	Graphical representation of the NOR.
Example	Graphical representation of the example of NOR.
OUTPUT: Designed Ontology	
General	Description in natural language of the ontology created after applying the pattern for re-engineering the NOR.
Graphical Representation	
General	Graphical representation, using the UML profile (Brockmans & Haase, 2006), of the ontology created for the NOR being re-engineered.
Example	Example showing a graphical representation, using the UML profile (Brockmans & Haase, 2006), of the ontology created for the NOR being re-engineered.
PROCESS: How to Re-engineer	
General	Algorithm for the re-engineering process.
Example	Application of the algorithm to the NOR example.
Time Complexity	The time complexity of the algorithm.
Additional Notes	Additional notes of the algorithm.
Formal Transformation	
General	Formal description of the transformation by using the formal definitions of the resources.
Relationships (Optional)	
General	Description of any relation to other PR-NOR patterns or other ontology design patterns.

- (Line 15-16) the member holonym in the relation is interpreted as *Whole*.
- (Line 18) if WordNet gives an empty result, relate the two terms by means of the default relation, which was set by the user (Line 1).

Table 5. Set of patterns for re-engineering NORs

Identifier	Type of NOR	NOR Data Model	Target
PR-NOR-CLTX-01	C. Scheme	Path Enumeration	O. Schema
PR-NOR-CLTX-02	C. Scheme	Adjacency List	O. Schema
PR-NOR-CLTX-03	C. Scheme	Snowflake	O. Schema
PR-NOR-CLTX-04	C. Scheme	Flattened	O. Schema
PR-NOR-CLAX-10	C. Scheme	Path Enumeration	Ontology
PR-NOR-CLAX-11	C. Scheme	Adjacency List	Ontology
PR-NOR-CLAX-12	C. Scheme	Snowflake	Ontology
PR-NOR-CLAX-13	C. Scheme	Flattened	Ontology
PR-NOR-TSTX-01	Thesaurus	Record-based	O. Schema
PR-NOR-TSTX-02	Thesaurus	Relation-based	O. Schema
PR-NOR-TSAX-10	Thesaurus	Record-based	Ontology
PR-NOR-TSAX-11	Thesaurus	Relation-based	Ontology
PR-NOR-LXTX-01	Lexicon	Record-based	O. Schema
PR-NOR-LXTX-02	Lexicon	Relation-based	O. Schema
PR-NOR-LXAX-10	Lexicon	Record-based	Ontology
PR-NOR-LXAX-11	Lexicon	Relation-based	Ontology

It is worth mentioning that for asserting the *partOf* relation the algorithm takes advantage of the use of the *PartOf content pattern*[43], to guarantee that the OWL code generated follows common practices in Ontological Engineering.

Regarding the time complexity of the algorithm, it is constant, i.e. $O(1) + K$. Where K represents the time complexity of accessing WordNet method.

Method for Re-engineering NORs into Ontologies

The aim of the Method for Re-engineering Non-Ontological Resources is to transform a non-ontological resource into an ontology. Therefore, the output of the process is an ontology. The pattern-based method consists of three activities, which are depicted in Figure 5.

Activity 1. NOR Reverse Engineering. The goal is to analyze a NOR, to identify its underlying components, and to create representations of the resource at the different levels of abstraction, i.e. design, requirements and conceptual (see Figure 4). This activity is carried out by domain experts, software developers and ontology practitioners.

- *Task 1.1 Data gathering.* The goal is to search and compile all the available data and documentation about the NOR, including purpose, components, data model and implementation details.
- *Task 1.2 Conceptual abstraction.* The goal is to identify the schema of the NOR, including the conceptual components and their relationships. If the conceptual schema is not available in the documentation, the schema should be reconstructed manually or with a data modelling tool.
- *Task 1.3 Information exploration.* The goal is to find out how the conceptual schema of the NOR and its content are represented in the data model. If the NOR data model is not available in the documentation, the data

Table 6. Excerpt of the PR-NOR-CLTX-01 pattern

PR-NOR-CLTX-01
PROCESS: How to Re-engineer
General
1: noParentT erms ← classification scheme terms without parent 2: if noParentT erms.length > 1 then 3: entityName ← name of the entity that contains the classification scheme terms 4: rootClass ← createClass(entityName) 5: for ri ∈ noParentT erms do 6: Ri ← createClass(ri) 7: relation ← ExternalResource.getRelation(rootClass,Ri) 8: relate(relation,rootClass,Ri) 9: end for 10: end if 11: repeat 12: for cei ∈ noParentT erms do 13: if not alreadyCreatedClassFor(cei) then 14: Ci ← createClass(cei) 15: end if 16: children ← childrenOf(cei) 17: for ce j ∈ children do 18: if not alreadyCreatedClassFor(ce j) then 19: C j ← createClass(ce j) 20: end if 21: relation ← ExternalResource.getRelation(cei,ce j) 22: relate(relation,cei,ce j) 23: end for 24: add(allChildren,children) 25: end for 26: noParentT erms ← allChildren 27: removeAllTerms(allchildren) 28: until isEmpty(noParentT erms)
Example
1: noParentT erms ← [Legislators, senior officials and man- agers;Professionals] 2: // noParentT erms.length=2 > 1 3: entityName ← Occupation 4: rootClass ← createClass(entityName) 6: R1 ← createClass(Legislators, senior officials and managers) 7: relation1 ← ExternalResource.getRelation(rootClass,R1) 8: relate(relation1,rootClass,R1) 6: R2 ← createClass(Professionals) 7: relation2 ← ExternalResource.getRelation(rootClass,R2) 8: relate(relation2,rootClass,R2) 13: // Legislators, senior officials and managers class, R1, already created 16: children ← childrenOf(Legislators, senior officials and managers) 16: children ← [Legislators and senior officials;Corporate managers] // using the path enumeration model 19: C1 ← createClass(Legislators and senior officials) 21: rel1 ← ExternalResource.getRelation(R1,C1) 22: relate(rel1,R1,C1) 19: C2 ← createClass(Corporate managers) 21: rel2 ← ExternalResource.getRelation(R1,C2) 22: relate(rel2,R1,C2) 24: allChildren ← [Legislators and senior officials;Corporate managers] 13: // Professionals, R2, already created 16: children ← Ø ← childrenOf(Professionals) 26: noParentT erms ← [Legislators and senior officials;Corporate managers] 27: removeAllTerms(allChildren) 13: // Legislators and senior officials, C1, already created 16: children ← Ø ← childrenOf(Legislators and senior officials) 13: // Corporate managers, C2, already created 16: children ← Ø ← childrenOf(Corporate managers) 24: allChildren ← Ø 26: noParentT erms ← allChildren ← Ø

Algorithm 1. Discovering the semantics of the relations

```
1:  Take two related terms from the NOR, ti and tj
2:  defaultRelation ← userDefinedRelation
3:  if contains(ti,tj) then
4:    relation ← ti.subClassOf.tj
5:  else if contains (tj,ti) then
6:    relation ← tj.subClassOf.ti
7:  else
8:    wordnetRelation ← W ordNet(ti, tj)
9:    if wordnetRelation == hyponym then
10:     relation ← ti.subClassOf.tj
11:    else if wordnetRelation == hypernym then
12:     relation ← tj.subClassOf.ti
13:    else if wordnetRelation == meronym then
14:     relation ← ti.partOf.tj
15:    else if wordnetRelation == holonym then
16:     relation ← tj.partOf.ti
17:    else
18:     relation ← defaultRelation
19:    end if
20:  end if
21:  return relation
```

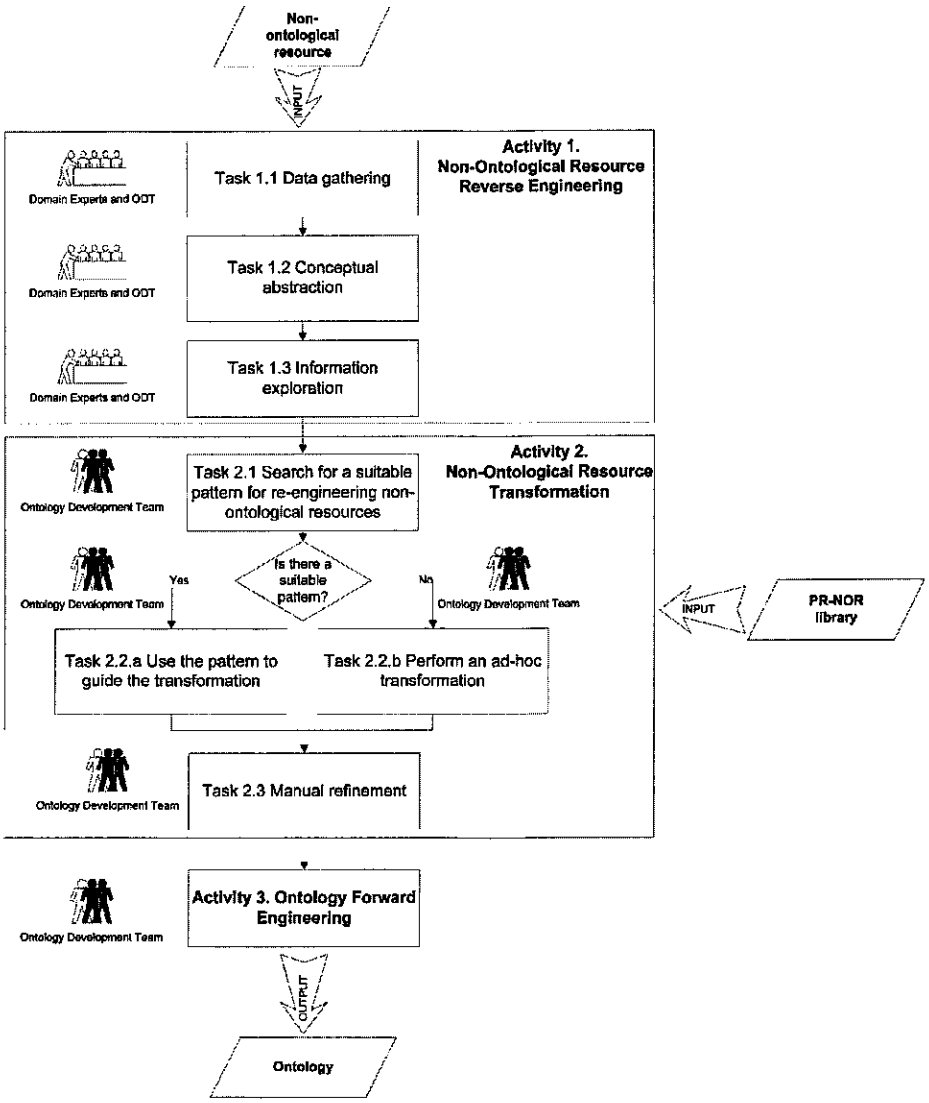
model should be reconstructed manually or with a data modelling tool.

Activity 2. NOR Transformation. The goal here is to generate a conceptual model from the NOR, using for that purpose the Patterns for Re-engineering NORs (PR-NOR), which guide the transformation process. This activity is carried out by software developers and ontology practitioners.

- *Task 2.1 Search for a suitable pattern for re-engineering NOR.* The goal is to find out if there is any applicable re-engineering pattern for transforming the NOR into a conceptual model. This search is performed in the ODP Portal, with the following criteria: (1) NOR type, (2) internal data model of the resource, and (3) the transformation approach.
- *Task 2.2.a Use of re-engineering patterns for guiding the transformation.* The goal of this task is to apply the re-engineering pattern obtained in task 2.1 in order to transform the NOR into a conceptual model. If a suitable pattern is found, then the conceptual model is created following the procedure established in the pattern. Alternatively, the NOR_2O software library, described in the next section, can be used for generating the ontology automatically.
- *Task 2.2.b Carry out an ad-hoc transformation.* The goal is to set up an *ad-hoc* procedure to transform the NOR into a conceptual model, when a suitable pattern was not found. This *ad-hoc* procedure may be generalized to create a new pattern for re-engineering NORs.
- *Task 2.3 Manual refinement.* The goal is to check whether some inconsistency is present after the transformation. Ontology engineers with the support of domain experts can fix manually some of inconsistencies generated after the transformation.

Activity 3. Ontology Forward Engineering. The goal here is to generate the ontology. We use the ontology levels of abstraction to depict this activity because they are directly related to the ontology development process. This activity is carry out by software developers and ontology practitioners.

Figure 5. Re-engineering process for NORs



Example

To describe the proposed guidelines in a more practical way, we present an example from the SEEMP Project[44], specifically we exemplify the generation an Occupation Ontology from the EURES proprietary occupation classification[45].

Activity 1. NOR Reverse Engineering. Within this activity we gathered documentation about EURES occupation classification from European Dynamics SEEMP user partner. From this documentation we extracted the schema of the classification scheme which consists of two tables *CVO_OCCGROUP* and *CVO_OCCUGROUP_NAME*. Since the data model was not available in the documentation, it was necessary to extract it for the resource implementation itself. EURES occupation classification is modelled following the snowflake data model, and it is implemented in a MS Access database.

Activity 2. NOR Transformation. Within this activity we carried out the following tasks:

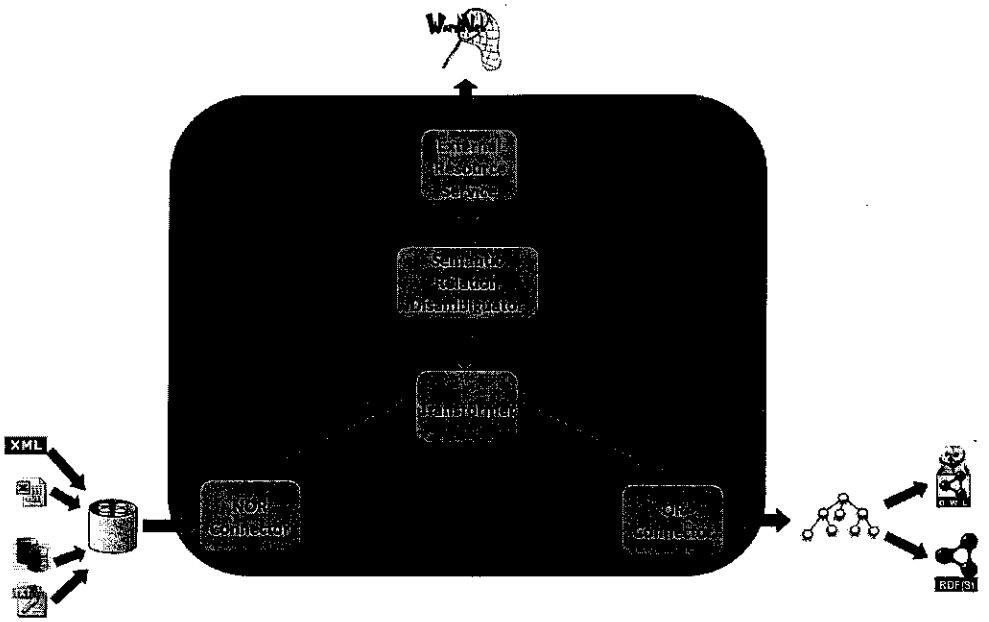
- We identified the transformation approach, the TBox transformation, i.e. transforming the resource content into an ontology schema.
- Then, we looked into in our local pattern repository for a suitable pattern to re-engineer NORs taking into account the transformation approach (TBox transformation), the non-ontological resource type (classification scheme), and the data model (snowflake data model) of the resource.
- The most appropriate pattern for this case is the PR-NOR-CLTX-03 pattern. This pattern takes as input a classification scheme modelled with a snowflake data model and produces an ontology schema.
- Because of the number of occupations of the EURES classification, more than five hundred occupations, it was not practical to create the ontology manually. Therefore, we used the NOR_2O software library, described in the next section, for generating the ontology automatically.

NOR_2O SOFTWARE LIBRARY

In this section we present NOR_2O[46] (Villazón-Terrazas, Gómez-Pérez & Calbimonte, 2010), a Java library that implements the transformation process suggested by the Patterns for Re-engineering Non-Ontological Resources (PR-NOR) described in the previous section. This library performs the ETL process[47] for transforming the non-ontological resource components into ontology primitives. A high-level conceptual architecture diagram of the modules involved is shown in Figure 6.

Figure 6 depicts the modules of the NOR_2O software library: NOR Connector, Transformer, Semantic Relation Disambiguator, External Resource Service, and OR Connector. These modules are described in detail in the following

Figure 6. Modules of the NOR2O software library



section. For illustrating the modules, the transformation of the ASFA thesaurus[48] into an ontology schema[49] has been taken as example.

NOR Connector

The NOR Connector loads classification schemes, thesauri, and lexicons modelled with their corresponding data models, and implemented in databases, XML, flat files and spreadsheets.

This module utilizes an XML configuration file for describing the NOR. Figure 7 shows the graphical representation of the NOR connector XSD file, including the following main sections:

- The *Schema* section. It describes the schema entities of the resource and the relationships among the entities.
- The *DataModel* section. It includes the descriptions of the resource's internal data model.
- The *Implementation* section. It defines the information needed to physically access the resource.

An example of the XML configuration file is presented in Figure 8. The figure shows that the file describes a thesaurus. The thesaurus has two schema entities, *Term* and *NonPreferredTerm*, and is modelled following the record-based data model; it is implemented in XML.

Transformer

This module performs the transformation suggested by the patterns, by implementing the sequence of activities included in such patterns. The module transforms the NOR elements, loaded by the NOR Connector module, into internal model representation elements. And it interacts with the Semantic Relation Disambiguator module for obtaining the suggested semantic relations of the NOR elements.

The Transformer also utilizes an XML configuration file, called prnor.xml, for describing the transformation between the NOR elements and the ontology elements. This XML configuration file has only one section, *PRNOR*, which includes the description of the transformation from the NOR schema components (e.g. schema entities, attributes and relations) into the ontology elements (e.g. classes, object properties, data properties and individuals). Additionally, it indicates the transformation approach, e.g. TBox, ABox or Population.

Figure 7. Graphical representation of the NOR connector XSD file

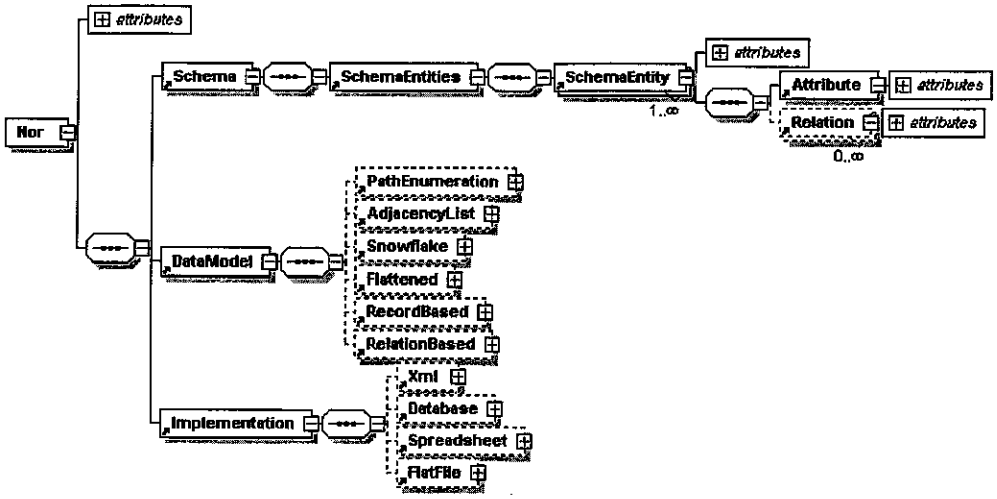


Figure 8. NOR Connector configuration file example

```
<Nor type="Thesaurus" name="ASFA">
  <Schema>
    <SchemaEntities>
      <SchemaEntity name="Term">
        <Attribute name="Identifier" valueFrom="DESCRIPTOR" type="string"/>
        <Relation name="NT" using="RecordBased" valueId="NT" destination="Term"/>
        <Relation name="BT" using="RecordBased" valueId="BT" destination="Term"/>
        <Relation name="RT" using="RecordBased" valueId="RT" destination="Term"/>
        <Relation name="UF" using="RecordBased" valueId="UF" destination="NonPreferredTerm"/>
      </SchemaEntity>
      <SchemaEntity name="NonPreferredTerm">
        <Attribute name="Identifier" valueFrom="NON-DESCRIPTOR" type="string"/>
      </SchemaEntity>
    </SchemaEntities>
  </Schema>
  <DataModel>
    <RecordBased>
      <Entity>CONCEPT</Entity>
    </RecordBased>
  </DataModel>
  <Implementation>
    <Xml xmlFile="asfa.xml" xsdFile="asfa.xsd"/>
  </Implementation>
</Nor>
```

Figure 9 shows the graphical representation of the PRNOR XSD file. Two examples of the XML configuration file are shown in Figure 10.

The Figure 10a indicates that the pattern follows the TBox transformation approach and transforms the elements of the *CSItem* schema component into ontology classes. Also by default, it transforms the *subType* schema relation into a *subClassOf* relation and the *superType* schema relation into a *superClassOf* relation, unless the Semantic Relation Disambiguator module suggests another relation. Figure 10b indicates that the pattern follows the TBox transformation approach and it transforms the elements of the *Term* schema component into ontology classes. Also by default, it transforms the *NT* schema relation into a *superClassOf* relation, the *RT* schema relation into a *relatedTerm* relation, and the *BT* schema relation into a *subClassOf* relation, unless the Semantic Relation Disambiguator module suggests another relation. Finally, the *UF* schema relation is transformed into a *rdfs:label*, and the module uses WordNet as external resource for disambiguating

Semantic Relation Disambiguator

This module works only with the TBox transformation approach, which converts the resource content into an ontology schema. To this end, each NOR term is mapped to a class, and then, the semantics of the relations among those entities is made explicit. The algorithm presented in previous section describes how to make explicit the semantics of the relations in the NOR terms.

External Resource Service

This is in charge of interacting with external resources for obtaining the semantic relations between two NOR terms. At this moment the module interacts with WordNet. We are implementing the access to DBpedia[50].

OR Connector

The Ontological Resource (OR) Connector generates the ontology in OWL-DL. To this end, this module relies on the OWL API[51]. It also utilizes an XML configuration file for

Figure 9. Graphical representation of the PRNOR XSD file

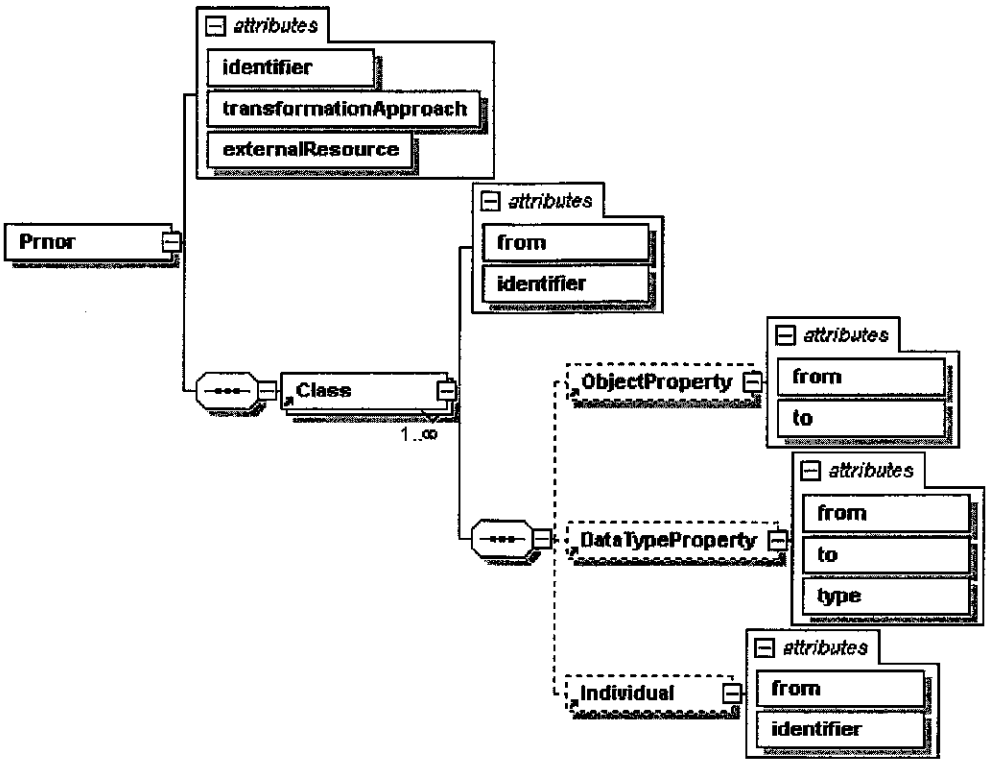


Figure 10. PRNOR configuration file examples

```
<Prnor identifier="PR-NOR-CLTX-01" transformationApproach="TBox"
  externalResource="WordNet">
   <Class from="CSItem" identifier="[CSIdentifier]">
      <ObjectProperty from="subType" to="subClassOf"/>
      <ObjectProperty from="superType" to="superClassOf"/>
   </Class>
</Prnor>
```

a) For Classification Schemes

```
<Prnor identifier="PR-NOR-TSLO-01" transformationApproach="TBox"
  externalResource="WordNet">
   <Class from="Term" identifier="[Identifier]">
      <ObjectProperty from="NT" to="superClassOf"/>
      <ObjectProperty from="RT" to="relatedTerm"/>
      <ObjectProperty from="BT" to="subClassOf"/>
      <ObjectProperty from="UF" to="rdfs:label"/>
   </Class>
</Prnor>
```

b) For Thesauri

Figure 11. Graphical representation of the OR XSD file

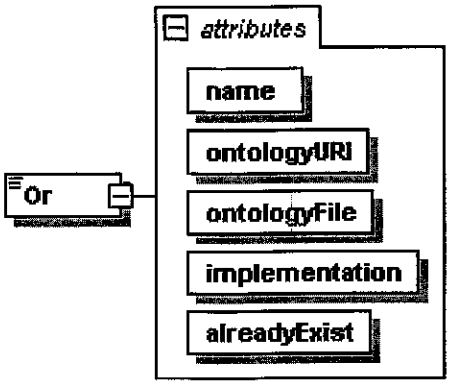


Figure 12. OR Connector configuration file example

```
<Or name="asfa ontology"
    ontologyURI="http://droz.dia.fi.upm.es/ontologies/asfa.owl"
    ontologyFile="asfa.owl" implementation="OWL" alreadyExist="no">
</Or>
```

describing the ontology to be generated. Figure 11 shows the graphical representation of the OR connector XSD file. The XML configuration file has only one section, *OR*, which includes the descriptions of the name, URI, file and implementation language of the ontology. Additionally, this section indicates if the ontology already exists, in the case we want to populate an existing ontology.

An example of the XML configuration file is shown in Figure 12. The figure indicates that the ontology generated will be stored in the *asfa.owl* file, its name will be *asfa ontology* and it will be implemented in OWL.

Finally, to conclude the description of the software library, it is worth mentioning that the implementation of this library follows a modular approach; therefore, it is possible to extend it in order to include other types of NORs, data models, and implementations in a simple way, as well as to exploit other external resources for relation disambiguation.

EXPERIMENTAL EVALUATION

In this section we describe two experiments we conducted with the objective of evaluating the methodological and technological aspects of our NOR Re-engineering approach. First, we assess the understandability and usability of the methodological guidelines. Second, we evaluate the quality of the set of patterns proposed in this paper. We focus on the TBox transformation because it is the most challenging transformation, however we intend to experiment the rest of transformation approaches in the future.

Methodological Evaluation

In this section we present the setting of the experiment we have carried out on the methodological guidelines proponed in this paper. This example refers to the manual transformation of an excerpt from a thesaurus, by using the guidelines and the proposed set of patterns. The purpose here is to assess the understandability and usability both the methodological

guidelines and the set of patterns for carrying out the NOR Re-engineering when WordNet is used for disambiguating the relations between terms. Moreover, we present a comparison of the three most representative methods, introduced previously, with our pattern-based method.

Settings

The evaluation was carried out with participants whose background included databases, software engineering, AI, and had some experience in ontology engineering.

- The "Ontologies and Semantic Web" course within the "Athens Programme" taught at the Facultad de Informática (UPM). Fourteen international students attended the course.
- The "Ontologies and Semantic Web" course within the "Information Technology" Master, taught at the Facultad de Informática (UPM). Twenty Spanish students attended the Master course.

We carried out the evaluations in two separate experiments that were performed at different points in time, leaving an interval of one or two weeks between them. During this time, we did not modify the patterns and disambiguation algorithm.

We selected the ETT thesaurus. The excerpt from this thesaurus contains twenty terms[52]. The participants had to build manually the conceptual model from the resource, by analyzing the methodological guidelines and the set of patterns. They had 30 minutes for generating the conceptual model.

Execution

The experiment was executed in four phases:

1. The students were provided with the proposed guidelines.
2. The students were organized in groups of two.
3. The groups of students analyzed the methodological guidelines and the set of patterns in order to carry out the NOR re-engineering process. They generated manually the conceptual model.
4. The students filled in a questionnaire.

Next, we show the tasks performed within phase 3 to generate the conceptual model from the excerpt of the resource.

NOR Reverse Engineering. Within this activity the groups gathered documentation about the thesaurus from the ETT web site. From this documentation they extracted the schema of the thesaurus. Since the data model was not available in the documentation, they extracted it for the resource implementation itself.

NOR Transformation. Within this activity the groups looked into the ODP portal for a suitable PR-NOR, taking into account the following criteria: (1) the resource type, the thesaurus; (2) the resource data model, the record-based model; and (3) the transformation approach selected, the TBox transformation. The most appropriate pattern was the PR-NOR-TSTX-01. Finally, students followed the procedure suggested by the pattern, for creating manually the conceptual model. Each thesaurus term was mapped to a class. For the disambiguation of the semantics of the BT, NT relations among thesaurus terms, the participants checked whether they could get the *subClassOf* relation by identifying attribute adjectives. If this was not the case, they searched the WordNet web site. When the query results were empty, they related the terms by means of the default relations (see Figure 10).

Ontology Forward Engineering. Since the goal was to create a conceptual model, the students did not perform this activity.

Collecting Results

Students were asked to answer the following questionnaire.

Q1. Are the guidelines proposed well explained?
Q2. Do the guidelines need to be more detailed? If so, please elaborate on your comments.
Q3. Do you think that more techniques and patterns should be provided?
Q4. How can we improve the guidelines proposed? And in which tasks?
Q5. Did you find these guidelines useful?

Findings and Observations

Table 7 presents the 34 answers to the questionnaire. As a general conclusion we can say that the students did not seem to find any problems with the use and understanding of each of the activities and tasks identified in the methodological guidelines.

Based on the comments obtained in this experiment, we can say that the main strength is that the methodological guidelines were useful and understandable.

Comparison with the Methods of the State of the Art

In this section we review the main contributions of our pattern-based method, against the existing methods introduced in the comparative framework for re-engineering NORs into ontologies. Table 8 presents the three most representative methods: Hepp et al. (Hepp & Brujin, 2007), Hyvönen et al. (Hyvönen et al., 2008), and Soergel et al. (Soergel et al., 2004), compared against our pattern-based method by using the evaluation framework defined previously.

Technological Evaluation: Quality of the Patterns and NOR_2O

The main purpose of this study is to evaluate the quality of the re-engineering patterns being used for transforming the NOR into an ontology by a disambiguation algorithm that lies on WordNet, the algorithm is implemented in the NOR_2O software library. The ontology generated is compared against a reference ontology (or gold standard) that was built manually by external ontology experts not involved in the experiment.

Settings

For this experiment, two ontology engineering experts built five ontologies, in OWL, from existing NORs (two classification schemes, two thesauri and one lexicon) of different domains. One expert built two ontologies and the other built three ontologies. Then, the experts exchanged their ontologies in order to evaluate them. Later, the experts refined the ontologies by following the comments provided in the review. At the end of this process we obtained five "gold standard" ontologies[53]. It is worth

Table 7. Answers to the questionnaire proposed

Questions	Answers
Q1.	Ninety seven percent of the participants indicated that guidelines were well explained.
Q2.	Eighty eight percent of the students considered that more detail was not necessary in the guidelines; however twelve percent explained that they would welcome the improvement of the explanations of: i) how to search for a suitable pattern (task 2.1 in the guidelines), and ii) how to perform the ontology formalization (activity 3 in the guidelines).
Q3.	One hundred percent of the participants believed that the techniques and patterns to execute each ac- tivity of the guidelines were sufficient.
Q4.	Eighty five percent of the participants proposed to include more examples of how to use the proposed guidelines and what results were expected.
Q5.	One hundred percent of the participants believed that the guidelines were useful, but also necessary.

Table 8. Comparative analysis of the three representative methods and the pattern-based method

Features	Heep et al.	Hyvönen et al.	Soerger et al.	Pattern-Based Method
Non-Ontological Resource				
Type	classification scheme, thesaurus	thesaurus	thesaurus	classification scheme, thesaurus, lexicon
Data model is used	No	No	Yes	Yes
Implementation	database	Not mentioned	database	database, XML, spreadsheet, flat file
Transformation				
Transformation approach	TBox	TBox	TBox	TBox, ABox, Population
Semantics of the NOR relations	*subClassOf*, ad-hoc relation	*subClassOf*, *partOf*	*subClassOf*, ad-hoc relation	*subClassOf*, *partOf*
Additional resources	No	DOLCE	Domain expert	WordNet
Technique	Not mentioned	Not mentioned	Not mentioned	Re-engineering patterns
Tool support	SKOS2GenTax	ad-hoc tool	Not mentioned	NOR_2O
Ontology				
Components	classes, relations	classes, attributes, relations	classes, attributes, relations	classes, attributes, relations, instances
Language	RDF(S)/OWL-DLP	RDF(S)	OWL-DL	OWL-DL/RDF
Single / Several	single	single	single	single

mentioning that the ontologies cover an excerpt of the resources. Table 9 shows the resources utilized in this experiment:

Execution

The experiment was executed in the three phases:

1. Each NOR has been transformed automatically using the following patterns
   - ASFA, using the PR-NOR-TSTX-01 pattern.
   - ETT, using the PR-NOR-TSTX-01 pattern.
   - ACM, using the PR-NOR-CLTX-02 pattern.
   - FOET, using the PR-NOR-CLTX-01 pattern.
   - BioLexicon, using the PR-NOR-LXTX-02 pattern.
2. For disambiguating the relations between entities of a particular resource we have executed the disambiguation algorithm using WordNet.
3. In order to assess the quality of the ontologies generated, we compared, the "gold standard" ontologies with the five ontologies generated automatically by means of similarity measures based on (1) the Cider System (Gracia, 2009), which considers the structure of the ontologies, i.e. classes, object properties and datatype properties; and (2) the *StrucSubsDistAlignment* measure taken from the Ontology Alignment Evaluation Initiative[54], which contemplates the structure of the ontologies.

Table 9. Resources utilized in the experiment

Name	Type	Data Model	Implementation	N. of terms	N.of terms covered
ASFA	thesaurus	record- based	XML	9882	188
ETT	thesaurus	record- based	XML	2522	337
ACM	classification scheme	adjacency list	XML	1606	223
FOET	classification scheme	path enumeration	spreadsheet	127	112
BioLexicon	lexicon	relation- based	database	53876	150

Collecting Results

We built a table for comparing each one of the "Gold Standard" ontologies to the ontologies generated, by means of the similarity measures.

Finding and Observations

Table 10 presents the similarity values of each of the ontologies generated. We can say that the ontologies generated have an acceptable similarity degree to the gold standard ones.

Based on the results obtained, we can say that the main strength of the NOR_2O software library is that generates ontologies with an acceptable level of quality, meaning by quality how similar are the ontologies to the gold standard.

CONCLUSION

In this paper we have introduced a categorization of NORs according to three different features: type of NOR, data model, and implementation. Additionally, we have presented a framework that compares the existing methods for re-engineering NORs and we provided conclusions for the comparative study. The main contributions of this paper are (1) the model for re-engineering NORs into ontologies; (2) the patterns for re-engineering NORs; (3) the NOR_2O software library that implements the transformations suggested by the patterns; and (4) the method for re-engineering NORs into ontologies. Finally, we have depicted an evaluation of the method, patterns and software library.

We have shown that the approach presented: (1) copes with the following set of NORs, i.e., classification schemes, thesauri, and lexica, in an uniform way; (2) deals with NORs implemented in databases, XML files, flat files or spreadsheets; (3) takes into account the internal data model of the resources; and (4) contemplates the semantics of the NOR relations. Moreover, we have demonstrated that the use of patterns (1) embodies expertise about how

Table 10. Similarity values of each of the ontologies generated with the "Gold Standard" ontology

Similarity values between ontologies generated with the gold standard		
	Cider	StrucSubsDistAlignment
ASFA	0.754	0.631
ETT	0.713	0.745
ACM	0.620	0.870
FOET	0.621	0.753
BioLexicon	0.515	0.793

to guide a re-engineering process, (2) improves the efficiency of the re-engineering process, and (3) makes the transformation process easier for both ontology engineers and domain experts.

Regarding the evaluation of the methods, patterns and software library, the main conclusions are: (1) NOR_2O software library generates ontologies with an acceptable level of quality; (2) the majority of participants find that the methodological guidelines are useful and understandable; and (3) NOR_2O software library really makes ontology development easier and faster. Therefore, the user saves time and effort in the development of ontologies. The results of the experiments provide an indication of the real value and practical usability of the method, patterns, and software library proposed in this paper. Moreover, we have compared our pattern-based method against the three most representative methods of the state of the art.

Although in this paper we address open research problems in the context of re-engineering non-ontological resources, there are still further works that can be done in the near future, and they are (1) the improvement of the disambiguation algorithm by including DBpedia as an additional external resource; (2) the creation of richer and more complex ontologies, by including more knowledge in the patterns, for example the disjoint knowledge; (3) the integration of several NORs into one ontological model; and (4) the enrichment of the patterns by including a section that explains how to generate instances following the Linking Open Data[55] recommendations.

ACKNOWLEDGMENTS

This work has been partially supported by the European Commission projects NeOn (FP6-027595) and SEEMP (FP6-027347), as well as by an R+D grant from the UPM. We would like to kindly thanks María Poveda, Esther Nuñez, and Rosario Plaza.

REFERENCES

Alexander, C. (1979). *The Timeless Way of Building*. New York: Oxford University Press.

An, Y., Borgida, A., & Mylopoulos, J. (2005). Constructing Complex Semantic Mappings between XML Data and Ontologies. In *Proceedings of the International Semantic Web Conference* (pp. 6-20).

An, Y., & Mylopoulos, J. (2005). Translating XML Web Data into Ontologies. In *Proceedings of the OTM Workshops* (pp. 967-976).

Barrasa, J. (2007). *Modelo para la Definición Automática de Correspondencias Semánticas entre Ontologías y Modelos Relacionales*. Madrid, Spain: Facultad de Informática, Universidad Politécnica de Madrid.

Barrasa, J., Corcho, O., & Gómez-Pérez, A. (2004). R_2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In *Proceedings of the Second Workshop on Semantic Web and Databases (SWDB2004)*.

Bizer, C. (2009). The Emerging Web of Linked Data. *IEEE Intelligent Systems*, *24*(5), 87–92. doi:10.1109/MIS.2009.102

Brandon, D. (2005). Recursive Database Structures. *Journal of Computing Sciences in Colleges*.

Byrne, E. J. (1992). A Conceptual Foundation for Software Re-engineering. In *Proceedings of the International Conference on Software Maintenance and Reengineering* (pp. 226-235). Washington, DC: IEEE Computer Society.

Caracciolo, C., Heguiabehere, J., Presutti, V., & Gangemi, A. (2009). *Initial Network of Fisheries Ontologies*. NeOn project.

Carkenord, B. (2002). *Why Build a Logical Data Model*. Retrieved from http://etnaweb04.embarcadero.com/resources/tech_papers/datamodel.pdf

Clark, P., Thompson, J., & Porter, B. W. (2000). Knowledge Patterns. In *Proceedings of KR2000* (pp. 591–600). Principles of Knowledge Representation and Reasoning.

Cruz, I. F., Xiao, H., & Hsu, F. (2004). An Ontology-Based Framework for XML Semantic Integration. In *IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium* (pp. 217-226). Washington, DC: IEEE Computer Society.

Edwards, H., Puckettm, R., & Jolly, A. (2006). Analyzing Communication Patterns in Software Engineering Projects. In *Software Engineering Research and Practice* (pp. 310-315).

Foxvog, D., & Bussler, C. (2006). Ontologizing EDI Semantics. In *Proceedings of the ER Workshops* (pp. 301-311).

Gangemi, A., Guarino, N., Masolo, C., & Oltramari, A. (2003). Sweetening WordNet with DOLCE. *AI Magazine, 24*(3), 13–24.

Gangemi, A., Navigli, R., & Velardi, P. (2003). The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. In *Proceedings of the CoopIS/DOA/ODBASE Conference.*

Gangemi, A., Pisanelli, D., & Steve, G. (1998). Experiences with Medical Terminologies. In *Ontology in Information Systems* (pp. 163–178). Ontology Integration.

García, R., & Celma, O. (2005). Semantic Integration and Retrieval of Multimedia Metadata. In *Proceedings of the ISWC 2005 Workshop on Knowledge Markup and Semantic Annotation (Semannot'2005).*

García-Silva, A., Gómez-Pérez, A., Suárez-Figueroa, M. C., & Villazón-Terrazas, B. (2008). A Pattern Based Approach for Re-engineering Non-Ontological Resources into Ontologies. In *ASWC '08: Proceedings of the 3rd Asian Semantic Web Conference* (pp. 167-181). Berlin: Springer-Verlag.

Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2003). *Ontological Engineering.* Berlin: Springer-Verlag.

Gómez-Pérez, A., & Manzano-Macho, D. (2004). An overview of methods and tools for ontology learning from text. *The Knowledge Engineering Review, 19*(3), 187–212. doi:10.1017/S0269888905000251

Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2009). Scenarios for Building Ontology Networks within the NeOn Methodology. In *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009).*

Gracia, J. (2009). *Integration and Disambiguation Techniques for Semantic Heterogeneity Reduction on the Web.* Zaragoza, Spain: University of Zaragoza.

Haase, P., Rudolph, S., Wang, Y., & Brockmans, S. (2006). *Networked Ontology ModelNetworked Ontology Model.* NeOn project.

Hahn, U., & Schulz, S. (2003). Towards a broad-coverage biomedical ontology based on description logics. *Pacific Symposium on Biocomputing, 8*, 577-588).

Hahn, V. (2003). Turning informal thesauri into formal ontologies: a feasibility study on biomedical knowledge re-use. *Comparative and Functional Genomics, 4*(1), 94–97. doi:10.1002/cfg.247

Hakkarainen, S., Hella, L., Strasunskas, D., & Tuxen, S. (2006). A Semantic Transformation Approach for ISO 15926. In *Proceedings of the OIS 2006 First International Workshop on Ontologizing Industrial Standards.*

Hepp, M. (2006). Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards. *International Journal on Semantic Web and Information Systems, 2*(1), 72–99.

Hepp, M. (2007). Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. *IEEE Internet Computing, 11*(1), 90–96. doi:10.1109/MIC.2007.20

Hepp, M., & de Brujin, J. (2007). GenTax: A generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In *Proceedings of the 4th European Semantic Web Conference (ESWC2007).* Berlin: Springer-Verlag.

Hirst, G. (2004). Ontology and the Lexicon . In *Handbook on Ontologies in Information Systems* (pp. 209–230). Berlin: Springer.

Hodge, G. (2000). Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files.

Hyvönen, E., Viljanen, K., Tuominen, J., & Seppälä, K. (2008). Building a National Semantic Web Ontology and Ontology Service Infrastructure - The FinnONTO Approach. In *Proceedings of the European Semantic Web Conference, ESWC* (pp. 95-109).

ISO. (1986). *Documentation - Guidelines for the establishment and development of monolingual thesaurus (Rep. ISO 2788).* Geneva, Switzerland: ISO.

ISO/IEC. (2004). *Information technology - Metadata registries - Part 1: Framework (Rep. ISO/IEC FDIS 11179-1).* Geneva, Switzerland: ISO.

Kimball, R., & Caserta, J. (2004). The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning . In *The Data Warehouse ETL Toolkit.* New York: John Wiley & Sons.

Lauser, B., & Sini, M. (2006). From AGROVOC to the agricultural ontology service/concept server: an OWL model for creating ontologies in the agricultural domain in the agricultural domain. In *DCMI '06: Proceedings of the 2006 International Conference on Dublin Core and Metadata Applications* (pp. 76-88). Dublin, Ireland: Dublin Core Metadata Initiative.

Maedche, A., & Staab, S. (2001). Ontology Learning for the Semantic Web. *IEEE Intelligent Systems.*

Malinowski, E., & Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering, 59*(2). doi:10.1016/j.datak.2005.08.003

Pinto, H. S., Tempich, C., & Staab, S. (2004). DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)* (pp. 393-397).

Pooley, R., & Stevens, P. (1998). *Software Reengineering Patterns*. Retrieved from http://www.reengineering.ed.ac.uk/csgrep.pdf

Presutti, V., Gangemi, A., David, S., de Cea, G. A., Suárez-Figueroa, M. C., & Montiel-Ponsoda, E. (2008). *NeOn Deliverable D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies.* Retrieved from http://www.neon-project.org

Soergel, D. (1995). *Data models for an integrated thesaurus database.* Retrieved from http://www.dsoergel.com/cv/B54.pdf

Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., & Katz, S. (2004). *Reengineering Thesauri for New Applications: The AGROVOC Example.* Rome, Italy: FAO.

Staab, S., Schnurr, H., Studer, R., & Sure, Y. (2001). Knowledge Processes and Ontologies. *IEEE Intelligent Systems, 16*(1), 26–34. doi:10.1109/5254.912382

Stojanovic, L., Stojanovic, N., & Volz, R. (2002). A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web. In *Proceedings of the Conference on Intelligent Information Processing.*

Suárez-Figueroa, M. C. (2010). *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse.* Madrid, Spain: Facultad de Informática, Universidad Politécnica de Madrid.

Suárez-Figueroa, M. C., & Gómez-Pérez, A. (2008). First Attempt towards a Standard Glossary of Ontology Engineering Terminology. In *Proceedings of the 8th International Conference on Terminology and Knowledge Engineering (TKE2008),* Copenhagen, Denmark.

Van Assem, M., Gangemi, A., & Schreiber, G. (2006). Conversion of WordNet to a standard RDF/OWL representation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06),* Genova, Italy.

Van Assem, M., Malaisé, V., Miles, A., & Schreiber, G. (2006). A Method to Convert Thesauri to SKOS. In *The Semantic Web* (pp. 95–109). Research and Applications.

Van Assem, M., Menken, M., Schreiber, G., & Wielemaker, J. (2004). A Method for Converting Thesauri to RDF/OWL. In *Proceedings of the Third International Semantic Web Conference (ISWC).* Berlin: Springer.

Villazón-Terrazas, B., Gómez-Pérez, A., & Calbimonte, J. P. (2010). NOR$_2$O: a Library for Transforming Non-Ontological Resources to Ontologies. In *Proceedings of the Seventh Extended Semantic Web Conference (ESWC 2010).*

Wielinga, B., Schreiber, A. T., Wielemaker, J., & Sandberg, J. (2001). From thesaurus to ontology. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture* (pp. 194-201). New York: ACM Press.

Wright, S., & Budin, G. (1997). *Handbook of terminology management, Basic aspects of terminology management.* Amsterdam, The Netherlands: John Benjamins Publishing Company.

ENDNOTES

1 http://www.neon-project.org

2 An ontology network is a collection of ontologies together through a variety of different relationships such as mapping, modularization, version, and dependency relationships (Haase, Rudolph, Wang & Brockmans, 2006).

3 A data model (Carkenord, 2002) is an abstract model that describes how data is represented and accessed. There are three basic styles of a data model: (1) the conceptual data model, which presents the primary entities and relationships of concern to a specific domain; (2) the logical data model, which depicts the logical entity types, the data attributes describing

those entities, and the relationships between entities; and (3) the physical data model, which is related to a specific implementation of the resource. In this paper we will use the term data model when referring to the logical data model.

4 http://www.fao.org/fi/glossary/default.asp
5 http://www.fao.org/figis/servlet/RefServlet
6 http://www.fao.org/agrovoc/
7 http://wordnet.princeton.edu/
8 http://www.vanderwal.net/folksonomy.html
9 http://del.icio.us/
10 http://www.rosettanet.org/
11 http://www.edibasics.co.uk/
12 http://www.unspsc.org/
13 http://wordnet.princeton.edu/
14 http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html
15 http://www.nlm.nih.gov/mesh/
16 http://www.getty.edu/research/tools/vocabularies/aat/index.html
17 http://www.ilo.org/public/english/bureau/stat/isco/index.htm
18 http://libserver.cedefop.europa.eu/ett/en/
19 http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=DSP_GEN_DESC_VIEW_NOHDR&StrNom=EDU_TRAINI&StrLanguageCode=EN
20 http://www.fao.org/fishery/asfa/8/en
21 http://aims.fao.org/website/AGROVOC-Thesaurus/sub
22 http://www.fao.org/figis/servlet/RefServlet
23 http://en.istat.it/
24 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29557
25 The EXPRESS file is a computer-interpretable of ISO 15926-2 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047
26 http://www.loa-cnr.it/DOLCE.html
27 http://www.nlm.nih.gov/research/umls/
28 http://www.isi.edu/isd/LOOM/
29 http://www.w3.org/2004/02/skos/
30 http://vesa.lib.helsinki.fi/
31 http://www.yso.fi/onto/yso
32 http://mpeg.chiariglione.org/
33 http://rhizomik.net/html/ontologies/mpeg7ontos/
34 http://www.ifla.org/VI/5/reports/rep4/42.htm\#chap2
35 http://www.w3.org/2001/sw/BestPractices/OEP/
36 http://www.gong.manchester.ac.uk/odp/html/index.html
37 http://ontologydesignpatterns.org
38 http://patterns.dataincubator.org/book/
39 According to (Byrne, 1992) software re-engineering is the examination and alteration of a software system to reconstitute it in a new form and the subsequent implementation of the new form.
40 Ontology Design Patterns are included in the ODP portal http://ontologydesignpatterns.org. The ODP portal is a Semantic Web portal dedicated to ontology design best practices for the Semantic Web, specially focused on ontology design patterns (OPs).
41 http://www.dbpedia.org/
42 Attributive adjectives are part of the noun phrase headed by the noun they modify; for example, happy is an attributive adjective in "happy people". In English, attributive adjectives usually precede their nouns in simple phrases, but often follow their nouns when the adjective is modified or qualified by a phrase acting as an adverb.
43 http://ontologydesignpatterns.org/wiki/Submissions:PartOf
44 http://www.seemp.org
45 http://www.eurodyn.com/
46 http://mccarthy.dia.fi.upm.es/nor2o/
47 The extraction, transformation, and loading (ETL) of legacy data sources, is a process that involves: (1) extracting data from the outside resources, (2) transforming it to fit operational needs, and (3) loading into the end target resources (Kimball, Ralph & Caserta, 2004).
48 http://www4.fao.org/asfa/asfa.htm
49 http://droz.dia.fi.upm.es/ontologies/asfa.owl
50 http://dbpedia.org/
51 http://owlapi.sourceforge.net/
52 http://droz.dia.fi.upm.es/master/rd/homework/resources/ett.xml
53 Ontologies available at http://droz.dia.fi.upm.es/ontologies
54 http://oaei.ontologymatching.org/
55 http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

Boris Villazón-Terrazas is a researcher and a PhD student in Artificial Intelligence at the Universidad Politécnica de Madrid. His University granted him a 4 year fellowship to carry out his PhD studies in the Ontology Engineering Group (OEG). Previously, the Ontology Engineering Group granted him a fellowship to carry out his master studies. He has previously worked as a researcher and software developer at the Research Institute of Informatics at the Universidad Católica Boliviana San Pablo. He obtained a BSc in computer Science (2002) from the Universidad Católica Boliviana San Pablo in Bolivia. He obtained the "Diploma de Estudios Avanzados (DEA)", equivalent to the current Master from the Universidad Politécnica de Madrid (2007). His research interests are focus on Semantic Web and Ontology Engineering, among others. He has participated in several European research projects such as Knowledge Web, SEEMP and NeOn as well as in national projects such as Reimdoc, Servicios Semánticos, Plata, Gis4Gov, and WebN+1. He has published more than 20 papers in journals, conferences and workshops.

Mari Carmen Suárez-Figueroa belongs to the Ontology Engineering Group (OEG) of the Artificial Intelligence Department of the Computer Science School (http://www.fi.upm.es) at Universidad Politécnica de Madrid (UPM). She graduated in Computer Science from UPM in 2001. She got the PhD in Artificial Intelligence in UPM in June 2010. She was Associate Professor (January-March 2002) and now she is Teaching Assistant from 1st September 2008 at the Computer Science School at UPM. Her research activities are focused on Ontology Engineering and the Semantic Web. Particularly, her research lines include methodologies for ontology network development, ontology network development, ontology development tools, ontology evaluation, and the Semantic Web. She has participated in different European and national projects: OntoWeb, Esperonto, PIKON, Knowledge Web, OntoGrid, REIMDOC, SEEMP, NeOn, mIO!, and BuscaMedia. She has been research visitor at Department of Computer Science (University of Liverpool) in 2004 and at KMi at the Open University in 2007. She has published more than 20 papers in journals, conferences and workshops. She co-organized the EON 2006 Workshop at WWW'06, the KRRSW 2008 Workshop at ESWC 2008, the tutorial called "NeOn Methodology: how to build ontology networks?" at EKAW 2008, and the tutorial called "Ontology Engineering: the NeOn Methodology through the NeOn Toolkit" at ISWC 2009.

Asunción Gómez-Pérez is Full Professor at the Universidad Politécnica de Madrid (UPM). She is the Director of the Artificial Intelligence Department (2008) and Director of the Ontology Engineering Group (OEG) at UPM (1995). She is a member of the AENOR CTN_148 Geographical Information Committee, the corresponding Spanish Committee that participates in the Working Group of ISO 1950. She has a B.A. in Computer Science (1990), M.S.C. on Knowledge Engineering (1991), Ph.D. in Computer Sciences (1993) and MS.C. on Business Administration (1994). She was visiting (1994-1995) the Knowledge Systems Laboratory at Stanford University. Her main research areas are: Ontological Engineering, Semantic Web and Knowledge Management. She led at UPM the following EU projects: MKBEEM, Ontoweb, Esperonto, Knowledge Web, NeOn, SEEMP, OntoGrid, Admire, Dynalearn, SemSorGrid4Env, SEALS and Monnet. She coordinated OntoGrid and now she is coordinating SemSorGrid4Env and SEALS. She is also leading at UPM projects funded by Spanish agencies. The most relevants are: España Virtual, Webn+1, Geobuddies and the Spanish network on Semantic Web. She has published more than 150 papers and she is author of one book on Ontological Engineering and co-author of a book on Knowledge Engineering. She has been co-director of the summer school on Ontological Engineering and the Semantic Web since 2003 up to now. She was program chair of ASWC'09, ESWC'05 and EKAW'02 and co-organiser of many workshops on ontologies. She reviews papers in many conferences and journals.