A Pattern Based Approach for Re-engineering Non-Ontological Resources into Ontologies

Andrés García-Silva, Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa, and Boris Villazón-Terrazas

Ontology Engineering Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Spain
hagarcia@delicias.dia.fi.upm.es,{asun,mcsuarez,bvillazon}@fi.upm.es

Abstract. With the goal of speeding up the ontology development process, ontology engineers are starting to reuse as much as possible available ontologies and non-ontological resources such as classification schemes, thesauri, lexicons and folksonomies, that already have some degree of consensus. The reuse of such non-ontological resources necessarily involves their re-engineering into ontologies. Non-ontological resources are highly heterogeneous in their data model and contents: they encode different types of knowledge, and they can be modeled and implemented in different ways. In this paper we present (1) a typology for non-ontological resources, (2) a pattern based approach for re-engineering non-ontological resources into ontologies, and (3) a use case of the proposed approach.

Keywords: Patterns for Re-engineering, Ontologies, Non-Ontological Resources.

1 Introduction

Research on Ontology Engineering methodologies has provided methods and techniques for developing ontologies from scratch. Well-recognized methodological approaches such as METHONTOLOGY [6], On-To-Knowledge [21], and DILIGENT [17] provide guidelines to help researchers in the development of ontologies. However, they have one important limitation: the lack of guidelines for building ontologies by reusing and re-engineering existing knowledge-aware resources widely used in a particular domain.

There are some initial works related to the re-engineering of non-ontological resources (NORs). Examples of projects that perform re-engineering are: (1) the NeOn Project[1], in which Fisheries Ontologies were developed for their use within the Fish Stock Depletion Assessment System (FSDAS) [4], by reusing resources available for the fisheries domain; and (2) the SEEMP[2] project in which a Reference Ontology has been built by reusing human resources management

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

standards. However, none of these projects propose any guidelines about how to carry out that re-engineering process of NORs.

Within the context of the NeOn project, we are proposing a novel scenario-based methodology for builing ontology networks[3]. One of the scenarios in the NeOn methodology is *Building Ontology Networks by Reusing and Re-engineering Non-Ontological Resources*. For such scenario we propose methodological guidelines for reusing and re-engineering NORs. In this paper we present our approach for re-engineering NORs, which refers to the process of taking an existing non-ontological resource and transforming it into an ontology. The rest of the paper is organized as follows: Section 2 depicts the proposed typology of NORs. Section 3 presents the state of the art on re-engineering NORs. Section 4 presents our approach for re-engineering NORs. Section 5 presents a particular use case of our approach. Finally, section 6 concludes the paper and proposes future lines of work.

2 Types of Non-Ontological Resources

Non-Ontological Resources are existing knowledge-aware resources whose semantics have not been formalized yet by means of an ontology.

There is a big amount of NORs that embody knowledge about some particular domains, and that represent some degree of consensus for a user comunity. These resources present the form of free texts, textual corpora, web pages, standards, catalogues, web directories, classifications, thesauri, lexicons and folksonomies, among others. NORs have related semantics which allow to interpret the knowledge they contain. Regardless of whether the semantic is explicit or not, the main problem is that the semantics of NORs are not always formalized, and this lack of formalization avoids the use of them as ontologies.

The analysis of the literature has revealed that there are different ways of categorizing NORs [14,20,7,13]. Maedche et al. [14] and Sabou et al. [20] classify NORs into unstructured (e.g. free text), semi-structured (e.g. folksonomies) and structured (e.g. databases) resources. Gangemi et al. [7] distinguish catalogues of normalized terms, glossed catalogues, and taxonomies. Hodge [13] proposes characteristics such as structure, complexity, relationships among terms, and historical functions for classifying them. However, an accepted typology of NORs does not exist yet. Additionally, the existing NOR categorizations do not take into account the NOR data model, an important artifact the re-engineering process.

In this paper we propose a new categorization of NORs according to three different features: (1) the type of NOR, which refers to the type of knowledge encoded by the resource; (2) the data model, that is, the design data model used to represent the knowledge encoded by the resource; and (3) the resource implementation. Below we explain in more detail the proposed classification.

[3] An ontology network or a network of ontologies is a collection of ontologies together through a variety of different relationships such as mapping, modularization, version, and dependency relationships [10].

1. According to the **type of NOR** we classify them into:
   - *Glossaries*: A glossary is a terminological dictionary that contains designations and definitions from one or more specific subject fields. The vocabulary may be monolingual, bilingual or multilingual. As an example we mention the FAO Fisheries Glossary[4].
   - *Lexicons*: 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. An example is WordNet[5], the best known computational lexicon of English.
   - *Classification schemes*: A classification scheme is the descriptive information for an arrangement or division of objects into groups based on characteristics the objects have in common. For example, the Fishery International Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP)[6].
   - *Thesauri*: Thesauri are controlled vocabularies of terms in a particular domain with hierarchical, associative and equivalence relations between terms. Thesauri are mainly used for indexing and retrieval of articles in large databases. As an example we can mention the AGROVOC[7] thesaurus.
   - *Folksonomies*: A folksonomy is the result of personal free tagging of information and objects (anything with a URI) for one's own retrieval. An example of the use of folksonomies is the *del.icio.us*[8] website.
2. There are different ways for representing the knowledge encoded by the resource. In the following we present several **data models** for classification schemes, which are shown in Fig. 1.
   - *Path Enumeration* [2]: A path enumeration model is a recursive structure for hierarchy representations defined as a model which 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. Fig. 1-a) shows this model.
   - *Adjacency List* [2]: An adjacency list model is a recursive structure for hierarchy representations comprising a list of nodes with a linking column to their parent nodes. Fig. 1-b) shows this model.
   - *Snowflake* [15]: An 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. Fig. 1-c) shows this model.
   - *Flattened* [15]: A flattened model is a denormalized structure for hierarchy representations. The hierarchy is represented using one table where each hierarchy level is stored on a different column. Fig. 1-d) shows this model.

[4] http://www.fao.org/fi/glossary/default.asp
[5] http://wordnet.princeton.edu/
[6] http://www.fao.org/figis/servlet/RefServlet
[7] http://www.fao.org/agrovoc/
[8] http://del.icio.us/

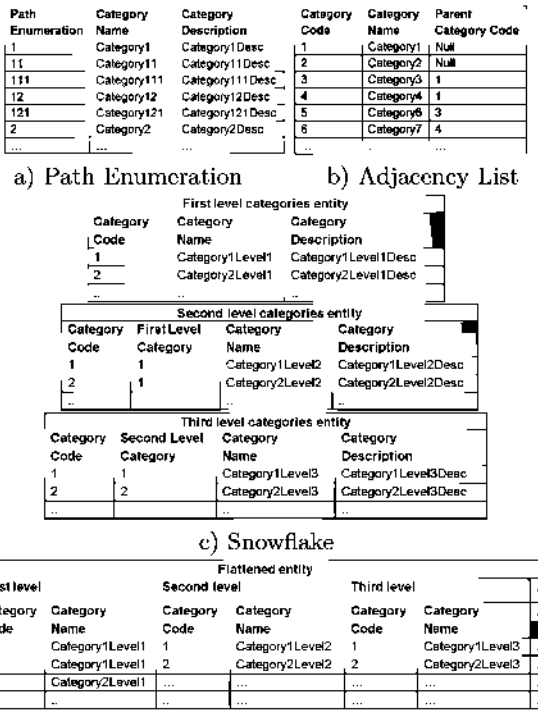


Fig. 1. Classification Schemes Data Models

3. According to the **implementation** we classify NORs into:
   - *Databases*: A collection of logically related data stored together in one or more files.
   - *XML file*: eXtensible Markup Language is a simple, open, and flexible format used to exchange a wide variety of data on and off the Web. XML is a tree structure of nodes and nested nodes of information, in which the user defines the names of the nodes.
   - *Flat file*: A flat file is a file that is usually read or written sequentially. In general, a flat file is a file containing records that have no structured inter-relationships.
   - *Spreadsheets*: An electronic spreadsheet consists of an array of cells into which a user can enter formulas and values.

Fig. 2 shows how a given type of NOR can be modeled following one or more data models, each of which could be implemented in different ways at the implementation layer. As an example, Fig. 2 shows a classification scheme

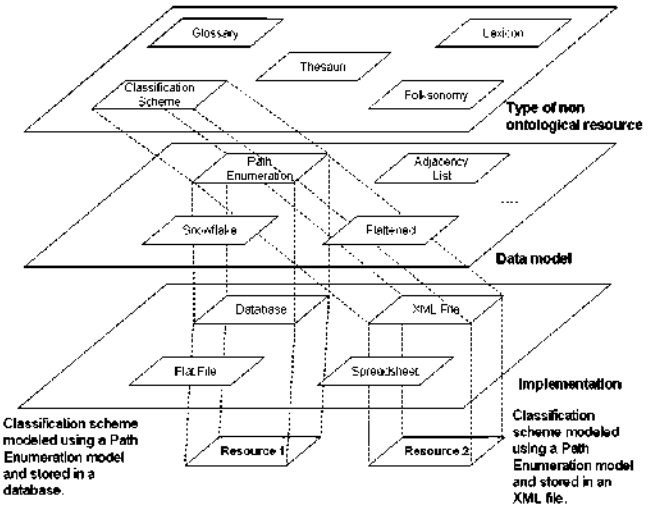


Fig. 2. Non-Ontological Resources (NORs) Categorization

modeled following a path enumeration model. In this case, the classification scheme is implemented in a database and in an XML file.

3 Related Work

In this section we present an overview of sofware re-engineering and a review of the state of the art on NOR re-engineering.

3.1 Software Re-engineering

Software re-engineering [5] is defined as the (1) examination of the design and implementation of an existing legacy system, and (2) application of the different techniques and methods to redesign and reshape that system into hopefully better and more suitable sofware.

Software re-engineering main activities are:

1. *Reverse engineering* [5] is the process of analyzing a subject system to identify the system components and their interrelationships, and create representations of the system in another form or at a higher level of abstraction.
2. *Alteration*, also called restructuring [5], is the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behaviour.
3. *Forward engineering* [5] is the traditional process of moving from high level abstractions and logical, implementation-independent designs to the physical implementation of a system.

Re-engineering patterns [18] are patterns that 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. They describe a process of re-engineering without proposing a complete methodology, and they can sometimes suggest a type of tool that one could use.

3.2 Non-Ontological Resource Re-engineering

Non-ontolgical resource re-engineering, defined in the Glossary of Activities in Ontology Engineering [24], refers to the process of taking an existing non-ontological resource and transforms it into an ontology.

The research in NOR re-engineering has been mainly centered on the transformation of standards [16,12], thesauri and lexicons [12,20,25], XML files [8], hierarchical classifications [9,12], folksonomies [20], relational databases [1,22], and spreadsheets [11]. These works only concentrate on the re-engineering process of the type and implementation of NOR.

In [20] Sabou et al. two approaches for the non-ontological resource transformation are distinguished. The first one consists in transforming resource schema into an ontology schema, and then resource content into instances of the ontology (Approach 1). The second one transforms resource content into an ontology schema (Approach 2). We add a third transformation approach which consists in transforming the resource content into instances of an existing ontology (Approach 3).

Table 1 shows a summary of the analyzed research works which have been focused on NOR type. Table 2 shows a summary of the research works which have been focused on the implementation of NORs. Both tables show the transformation approach, and also, if available, the name of the tool which supports the transformation approach. These research works just include *ad-hoc* methods and techniques for the transformation, i.e. the research works are specific of the NOR type or NOR implementation.

Re-engineering patterns are defined in [19] as transformation rules applied in order to create a new ontology (target model) from elements of a source model. The target model is an ontology, while the source model can either be an ontology or a NOR, e.g., a thesaurus concept, a data model pattern, a UML model, a linguistic structure, etc. In fact, [19] presents a unique example of a schema re-engineering pattern, which includes four rules to transform a knowledge organization system into SKOS[9]. These rules just identify the elements of the source model that are mapped to their corresponding elements of the target model, but the rules do not provide information about how to carry out the mapping. Re-engineering patterns are not integrated within a method to carry out the re-engineering process. Moreover, a template to describe re-engineering patterns in a unified way is not proposed.

[9] http://www.w3.org/2004/02/skos/

Table 1. Research works centered in the NOR type

Research Work	NOR Type	Transformation approach	Tool
Hepp et al. [12]	Classification schemes, thesauri, taxonomies	2	SKOS2GenTax
Mochol et al. [16]	Classification schemes	2	-
Sabou et al. [20]	Folksonomies	2	-
Sabou et al. [20]	Lexica	1,2	-
van Assem et al. [25]	Thesauri	1	-

Table 2. Research works centered in the NOR implementation

Research Work	NOR Implementation	Transformation approach	Tool
Stojanovic et al. [22]	Relational Database	1	KAON REVERSE
Barrasa et al. [1]	Relational Database	3	R_2O, ODEMapster
Garcia et al. [8]	XML files	1	XSD2OWL, XML2RDF
Han et al. [11]	SpreadSheet	3	RDF123

After having analyzed the state of the art on NORs re-engineering, we conclude that research efforts have been mainly devoted to the implementation and the type of NOR. It has also been analyzed how to map NORs content and schema into ontology instances and schema, but none of the analyzed research works have taken advantage from the data model which underlies the NOR to guide the re-engineering process. Finally, it is left to say that none of the analyzed re-engineering approaches propose a set of re-engineering patterns to guide the re-engineering process, and that there is also a lack of re-engineering methods.

4 Approach for Non-Ontological Resource Re-engineering

In this section we present our approach for NOR re-engineering. We describe a proposal for carrying out the NOR re-engineering process. Then, we present an example of the patterns for re-engineering NORs.

4.1 General Model for Non-Ontological Resource Re-engineering

In a nutshell, our approach for NOR re-engineering considers as input a pool of NORs and patterns for re-engineering NORs. NORs, as we mentioned in section 3, include lexica, classification schemes, thesauri, etc. Regarding patterns for

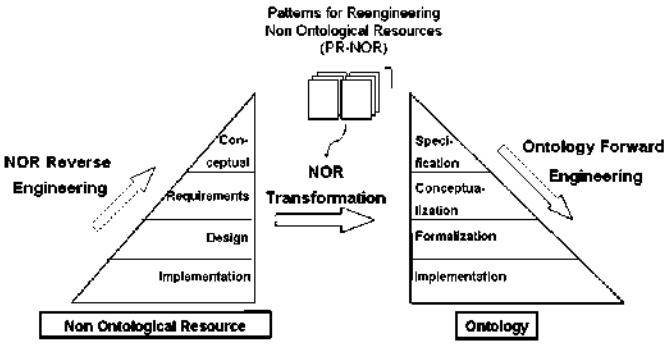


Fig. 3. Re-engineering Model for Non-Ontological Resources

re-engineering NORs, they provide solutions to the problem of transforming NORs into ontologies. These patterns will be included in the NeOn project patterns library[10].

Based on the software re-engineering model presented in [3] we propose our re-engineering model for NOR re-engineering in Fig.3.

The NOR re-engineering process consists of the following activities, which are defined in a Glossary of Activities in the Ontology Engineering[24]:

1. *Non-Ontological Resource Reverse Engineering*, whose goal is to analyze a NOR to identify its underlying components and create representations of the resource at the different levels of abstraction (design, requirements and conceptual). Since NORs can be implemented as XML files, databases or spreadsheet among others, we can consider them as software resources, and therefore, we use the software abstraction levels shown in Fig. 3 within this activity. Here the requirements and the essential design, structure and content of the NOR must be recaptured.
2. *Non-Ontological Resource Transformation*, whose goal is to generate a conceptual model from the NOR. We propose the use of Patterns for Re-engineering Non-Ontological Resources (PR-NOR) to guide the transformation process. First, the transformation approach has to be selected: (1) transforming resource schema into an ontology schema, and then resource content into instances of the ontology, (2) transforming resource content into an ontology schema, or (3) transforming the resource content into instances of an existing ontology. Second, the semantics of the relations between the NOR entities have to be identified, these semantics can be a)*subClassOf*, b)an *ad-hoc* relation like *partOf* or c)a mix of *subClassOf* and *ad-hoc* relations. Finally a pattern for re-engineering NORs according to the type of NOR, as well as the selected transformation approach, and the semantics of the relations between the NOR entities, has to be searched.

[10] http://www.ontologydesignpatterns.org

3. *Ontology Forward Engineering*, whose goal is to output a new implementation of the ontology on the basis of the new conceptual model. We use the ontology levels of abstraction to depict this activity because they are directly related to the ontology development process.

4.2 Patterns for Re-engineering Non-Ontological Resources

Patterns for re-engineering non-ontological resources (PR-NOR) define a procedure to transform the NOR components into ontology representational primitives. To this end, patterns take advantage of the NOR underlying data model. The data model defines how the different components of the NOR are represented.

According to the NOR categorization presented in section 3, 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 to extract the NORs components and then map them to the conceptual model of an ontology. Each process can be expressed as a pattern for re-engineering NORs.

The resultant ontologies proposed by the patterns for re-engineering NORs are modeled following the recommendations provided by some other ontological patterns such as logical and architectural patterns [23]. The current inventory of *NeOn Ontology Modelling Components* considered as Architectural Patterns includes the following ones: taxonomy, lightweight ontology and modular architecture. A taxonomy is the way of organizing an ontology as a hierarchical structure of classes only related by subsumption relations. A lightweight ontology adds the following features to the taxonomy structure: (a) a class can be related to other classes through the *disjointWith* relation, (b) object and datatype properties can be defined and used to relate classes, (c) a specific domain and range can be associated with defined object and datatype properties. Finally, the modular architecture consists in structuring an ontology as a configuration of components, each having its own identity based on some design criteria.

Moreover, the patterns for re-engineering NORs define the transformation process but they do not provide either an algorithm or an implementation of the process. We plan to include the algorithms and implementations later on in a framework which will implement the transformation process.

We have created eight patterns for re-engineering classifications schemes into taxonomies and lightweight ontologies, two for each data model identified (path enumeration, adjacency list, snowflake and flattened). We plan to extend this pool of patterns with more patterns for the rest of transformation approaches. Also we plan to include patterns for re-engineering the other types of NORs.

Next, we present an example of a re-engineering pattern identified in our ongoing research work on transforming classification schemes into ontologies. To present the patterns for re-engineering NORs we adapted the tabular template for ontology design patterns used in [23].

The pattern for re-engineering NOR shown in Table 4.2 suggests a guide to transform a classification scheme into a lightweight ontology. The classification scheme is modeled with a snowflake data model. This pattern aims at creating a lightweight ontology from the classification scheme.

Table 3. Pattern for Re-engineering a Classification Scheme

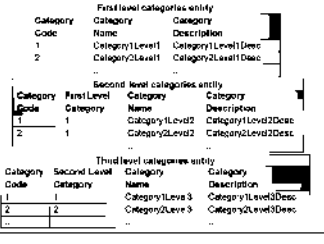
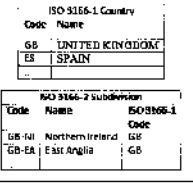
Slot	Value
General Information	
Name	Classification scheme to Lightweight Ontology (Snowflake model)
Identifier	PR-NOR-CLLO-01
Type of Component	Pattern for Re-engineering Non-Ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a classification scheme which follows the snowflake model to design a Lightweight Ontology.
Example	Suppose that someone wants to build a lightweight ontology based on the ISO 3166 standard for the representation of names of countries and their subdivisions. This standard is divided in ISO 3166-1 for countries, and ISO 3166-2 for subdivisions (regions).
Pattern for Re-engineering Non-Ontological Resources	
Resource to be Re-engineered	
General	A NOR holds a classification scheme which follows the snowflake model. A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children concepts may vary depending on the context. The snowflake model for hierarchical classifications proposes to create a fixed but separated entity (table, file) for each level of the hierarchy.
Example	The ISO 3166 standard (codes for the representation of names of countries and their subdivisions) is divided in ISO 3166-1 for countries, and ISO 3166-2 for country subdivisions (regions). For the example, ISO 3166-1 and ISO 3166-2 are hold on different entities. The relation semantics between the sub-ordinate and the super-ordinate concepts is *partOf*.
Graphical Representation	
General	
Example	
Designed Ontology	

Table 3. (*continued*)

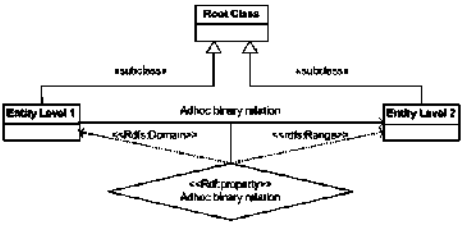
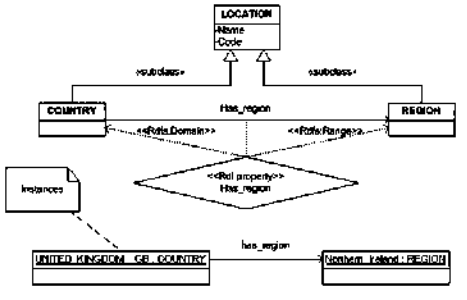
Slot	Value
General	The generated ontology will be based on the lightweight ontology architectural pattern (AP-LW-01)[23]. Each snowflake entity is mapped to a class. An *ad-hoc* binary relation is defined between the new classes according to the semantics of the relation between super-ordinate and sub-ordinate categories. Each data included on an entity is mapped to an instance of the entity class. The semantics of the relationship between sub-ordinate and super-ordinate instances is mapped to an *ad-hoc* binary relation instance.
Graphical Representation	
(UML)General Solution Ontology	
(UML)Example Solution Ontology	
How to Re-engineer	
General	1. Create a class for each entity in the snowflake model. 2. If there is a relationship between the entity classes then create it as an *ad-hoc* binary relation. 3. If there is a super-class for the new entity related classes then create it and set the appropriate *subClassOf* relation between the entity classes and the super-class. 4. For each record on each entity of the snowflake model, create an instance of the appropriate entity class. 5. If you have created an *ad-hoc* binary relation between the entity classes then you have to create the relation instance between the entity class instance.

Table 3. (*continued*)

Slot	Value
Example	1. Create a COUNTRY class for the ISO 3166-1 Countries entity and a REGION class for the ISO 3166-2 Subdivisions entity. 2. Create the *Has_region* binary relation with COUNTRY as domain and REGION as range. 3. Create a LOCATION class and assert that COUNTRY and REGION are *subClassOf* LOCATION. 4. For each record on the ISO 3166-1 Countries entity create an instance of the COUNTRY class. 5. For each COUNTRY instance look for its REGION on the ISO 3166-2 Subdivisions entity and create an instance of REGION for each subdivision found. Also create an instance of the *Has_region* relation associated to the current country instance and related to the current region instance.
Relationships	
Relations	Use the Architectural Pattern: AP-LW-01 [23]

5 SEEMP Use Case

A preliminary experimentation of our approach was done within the SEEMP project, in which NORs of the human resources domain were transformed into ontologies. We re-engineered four classification schemes using the overall set of patterns. We obtained the following ontologies:

- Occupation, Education, Economic activity ontologies. We applied the pattern *classification scheme (path enumeration) to lightweight ontology* (PR-NOR-CLTX-01), to re-engineer the ISCO-88 (COM), FOET, and NACE standards. These standards are classification schemes modeled following a path enumeration data model and they are stored in a MS Access database.
- Geography ontology. We applied the pattern *Classification scheme (adjacency list) to lightweight ontology* (PR-NOR-CLLO-02), to re-engineer the ISTAT[11] geography italian standard. This standard is a classification scheme modeled following an adjcency list data model and it is stored in a MS Excel spreadsheet.

In this section we present the activities carried out to re-engineer the ISTAT standard. This standard contains information about the divisions, regions and provinces of Italy. It is available in MS Excel spreadsheet format.

- *Non-Ontological Resource Reverse Engineering.* Within this activity we gathered documentation about ISTAT from domain web sites such as ISTAT web site itself and Eurostat. From this documentation we extracted the schema of the classification scheme which consists of 4 divisions, 20 regions and 106 provinces. Since the data model was not available in the documentation, it was necessary to extract it for the resource implementation itself.

[11] http://www.istat.it/

ISTAT is modeled following the adjacency list data model, i.e. each row of the spreadsheet contains the information related to a province, its region and its division.

- *Non-Ontological Resource Transformation.* Within this activity we carried out the following tasks:
  1. We followed approach 1, described in section 3, to carry out the transformation. This approach consists in transforming resource schema into an ontology schema, and then resource content into instances of the ontology.
  2. We identified the semantic of the relations between the NOR entities. In this case the relation was identified as *part Of*.
  3. Then, we looked in our local pattern repository for a suitable pattern to re-engineer NORs taking into account the selected transformation approach, the semantics of the relations between the NOR entities, and the data model of the resource.
  4. The most appropriate pattern for this case is the PR-NOR-CLLO-02 pattern. This pattern takes as input a classification scheme modeled with an adjacency list data model and produces a lightweight ontology.
  5. The selected pattern suggests to create a class for each one of the columns related to the main entities of the ISTAT standard. With this information we outlined the conceptual model for the ontology.
     (a) Create the DIVISION, REGION, and PROVINCE classes according to the ISTAT entities.
     (b) Create the *has_region* binary relation with DIVISION as domain and REGION as range.
     (c) Create the *has_province* binary relation with REGION as domain and PROVINCE as range.
     (d) Create a LOCATION class and assert that DIVISION, REGION and PROVINCE are *subClassOf* LOCATION.
     (e) Create an instance of the DIVISION class for each distinct ISTAT division .
     (f) Look for the REGIONS of each DIVISION instance in the ISTAT regions and create an instance of REGION for each distinct region. Create an instance of the *has_region* relation associated to the current division instance and related to the current region instance.
     (g) Look for the PROVINCES of each REGIONS instance in the ISTAT provinces and create an instance of PROVINCE for each distinct province. Create an instance of the *has_province* relation associated to the current region instance and related to the current province instance.
- *Ontology Forward Engineering.* WSML[12] is the ontology implementation language used in the SEEMP project. Because of the number of divisions, regions and provinces of the ISTAT standard, it was not practical to create the ontology manually. Therefore, we created an *ad-hoc* wrapper, implemented

[12] http://www.wsmo.org/wsml/

in Java, that reads the data from the resource implementation and automatically creates the corresponding classes, attributes and relations of the new ontology following the suggestion given by the pattern for re-engineering NORs and the conceptual model. The resultant ontology is available at `http://droz.dia.fi.upm.es/ontologies/`.

6 Conclusions and Future Work

In this paper we have introduced a three level categorization of NORs according to three different features: type of NOR, data model and implementation. Moreover, we present a pattern based approach for re-engineering NORs into ontologies. We take advantage of the NOR data model to define patterns for re-engineering NORs. We also describe a pattern for re-engineering a classification scheme into an ontology. Additionally, we present a use case of the proposed approach. Further work needs to be done to consider data models of the other NORs. If we can identify data models as we made for classification schemes we will be able to create more patterns to guide the re-engineering process. This approach will be extended for creating richer and more complex ontologies. We also need to calculate how much effort do we save re-engineering NORs using patterns compared with re-engineering NORs without them.

Acknowledgments. This work has been partially supported by the European Comission projects NeOn(FP6-027595) and SEEMP(FP6-027347), as well as by a UPM-BSCH grant, and an I+D grant from the UPM.

References

1. Barrasa, J., Corcho, O., Gómez-Pérez, A.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372. Springer, Heidelberg (2005)
2. Brandon, D.: Recursive database structures. Journal of Computing Sciences in Colleges (2005)
3. Byrne, E.J.: A conceptual foundation for software re-engineering. In: Proceedings of the International Conference on Software Maintenance and Reengineering. IEEE Computer Society Press, Los Alamitos (1992)
4. Caracciolo, C., Gangemi, A.: Revised and Enhanced Fisheries Ontologies. Technical report, NeOn project deliverable D7.2.2 (2007)
5. Chikofsky, E.J., Cross, J.H.: Reverse engineering and design recovery: a taxonomy. In: IEEE Software (1990)
6. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. In: Advanced Information and Knowledge Processing. Springer, Heidelberg (2003)
7. Gangemi, A., Pisanelli, D., Steve, G.: Ontology integration: Experiences with medical terminologies. Ontology in Information Systems, 163–178 (1998)
8. García, R., Celma, O.: Semantic Integration and Retrieval of Multimedia Metadata. In: Proceedings of the ISWC 2005 Workshop on Knowledge Markup and Semantic Annotation, Semannot 2005 (2005)

9. Giunchiglia, F., Marchese, M., Zaihrayeu, I.: Encoding Classifications into Lightweight Ontologies.. In: The Semantic Web: Research and Applications. Springer, Heidelberg (2006)
10. Haase, P., Rudolph, S., Wang, Y., Brockmans, S.: Networked Ontology Model. Technical report, NeOn project deliverable D1.1.1 (2006)
11. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: a mechanism to transform spreadsheets to RDF. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 2006). AAAI Press, Menlo Park (2006)
12. Hepp, M., de Bruijn, J.: GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 129–144. Springer, Heidelberg (2007)
13. Hodge, G.: Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files (2000), `http://www.clir.org/pubs/reports/pub91/contents.html`
14. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems (2001)
15. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: From conceptual modeling to logical representation. Data and Knowledge Engineering (2006)
16. Mochol, M., Paslaru, E.: Practical Guidelines for Building Semantic eRecruitment Applications. In: International Conference on Knowledge Management (iKnow 2006), Special Track: Advanced Semantic Technologies (2006)
17. Pinto, H.S., Tempich, C., Staab, S.: 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. IOS Press, Amsterdam (2004)
18. Pooley, R., Stevens, P.: Software reengineering patterns. Technical report (1998)
19. Presutti, V., Gangemi, A., David, S., Aguado de Cea, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: NeOn Deliverable D2.5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. In: NeOn Project (2008), `http://www.neon-project.org`
20. Sabou, M., Angeletou, S., dAquin, M., Barrasa, J., Dellschaft, K., Gangemi, A., Lehman, J., Lewen, H., Maynard, D., Mladenic, D., Nissim, M., Peters, W., Presutti, V., Villazón, B.: Selection and integration of reusable components from formal or informal specifications. Technical report, NeOn project deliverable D2.2.1 (2007)
21. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems (16), 26–34 (2001)
22. Stojanovic, L., Stojanovic, N., Volz, R.: A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web. In: Proceedings of the Conference on Intelligent Information Processing (2002)
23. Suárez-Figueroa, M.C., Brockmans, S., Gangemi, A., Gómez-Pérez, A., Lehmann, J., Lewen, H., Presutti, V., Sabou, M.: Neon modelling components. Technical report, NeOn project deliverable D5.1.1 (2007)
24. Suárez-Figueroa, M.C., Gómez-Pérez, A.: Towards a Glossary of Activities in the Ontology Engineering Field. In: Proceedings of the 6th Language Resources and Evaluation Conference, LREC 2008 (2008)
25. van Assem, M., Menken, M., Schreiber, G., Wielemaker, J.: A method for converting thesauri to RDF/OWL. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 17–31. Springer, Heidelberg (2004)