Departamento de Inteligencia Artificial
Facultad de Informática

PhD Thesis

A Method for Reusing and Re-engineering Non-ontological Resources for Building Ontologies

Author : Msc. Boris Marcelo Villazón Terrazas
Advisor : Prof. Dr. Asunción Gómez Pérez

2011

Tribunal nombrado por el Sr. Rector Magfco. de la Universidad Politécnica de Madrid, el día...............de.............................de 20....

Presidente : ____

Vocal : ____

Vocal : ____

Vocal : ____

Secretario : ____

Suplente : ____

Suplente : ____

Realizado el acto de defensa y lectura de la Tesis el día..........de......................de 20...... en la E.T.S.I. /Facultad..

Calificación ....

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

Abstract

Current well-known methodologies for building ontologies do not consider the reuse and possible subsequent re-engineering of existing knowledge resources. The ontologization of non-ontological resources has led to the design of several specific methods, techniques and tools. These are mainly specific to a particular resource type, or to a particular resource implementation. Thus, everytime ontology engineers are confronted with the task of re-engineering a new resource into an ontology, they develop *ad-hoc* solutions for transforming such resource into a single ontology.

Within the context of the NeOn project, we propose a novel methodology for building ontology networks: the NeOn Methodology, a methodology based on scenarios. One of these scenarios is *Building Ontology Networks by Reusing and Re-engineering Non-Ontological Resources*. As opposed to custom-building silos of single ontologies from scracth, this new scenario emphasizes the re-engineering of knowledge resources for building ontologies that are connected with other ontologies in the ontology network. The scope of this thesis lies in this scenario of the NeOn Methodology and in the use of re-engineering patterns for transforming non-ontological resources components into ontology representational primitives. Specifically, this thesis presents the following main contributions:

- A categorization of non-ontological resources, made by defining the term *non-ontological resources* and by proposing a three-level categorization of them according to the type, data model, and implementation of the resource.
- A metadata vocabulary, NoRMV, for describing non-ontological resources.
- Methodological guidelines for reusing available non-ontological resources, which have reached some degree of consensus by the community when building ontologies.
- Methodological guidelines for transforming the non-ontological resources selected into ontologies by re-engineering patterns.
- A set of re-engineering patterns for transforming classification schemes, thesauri, and lexica into ontologies.
- A software library, NOR$_2$O, that implements the transformations suggested by the re-engineering patterns when building ontologies.

The integrated framework proposed in this thesis allows speeding up the ontology development, thus saving time and effort.

Resumen

Las metodologías disponibles para el desarrollo de ontologías no tienen en cuenta la reutilización y posible re-ingeniería de recursos de conocimiento disponibles. La *ontologización* de recursos no-ontológicos ha dado lugar al diseño de varios métodos, técnicas y herramientas. Éstas son específicas para un tipo o implementación particular del recurso que se va transformar. Por lo tanto, cada vez que los ingenieros ontológicos se enfrentan a la re-ingeniería de un nuevo recurso en una ontología, tienen que desarrollar soluciones *ad-hoc* para poder transformar dicho recurso en una ontología.

Dentro del contexto del proyecto NeOn, se ha propuesto la Metodología NeOn, una metodología novedosa basada en escenarios, para desarrollar ontologías en red. Uno de estos novedosos escenarios es el de la Construcción de Ontologías mediante la Reutilización y Re-ingeniería de Recursos No-ontológicos. Al contrario que la construcción personalizada de silos de ontologías simples partiendo desde cero, este nuevo escenario destaca la re-ingeniería de recursos de conocimiento para la construcción de ontologías que están conectadas con otras dentro de la red de ontologías. El ámbito de esta tesis se circunscribe al escenario de la Metodología NeOn así como al uso de patrones de re-ingeniería para transformar los componentes de recursos no-ontológicos en elementos de una ontología. Esta tesis presenta específicamente las siguientes contribuciones:

- Una categorización de recursos de conocimiento, definiendo el término de recurso no-ontológico y presentado una clasificación de los mismos de acuerdo al tipo, modelo de datos e implementación.

- Un vocabulario de metadatos para describir los recursos no-ontológicos, NoRMV.

- Guías metodológicas para el proceso de reutilización de recursos no-ontológicos, que hayan alcanzado un grado de consenso dentro de una comunidad, para la construcción de ontologías.

- Guías metodológicas para el proceso de re-ingeniería de recursos no-ontológicos en ontologías, mediante el uso de patrones de re-ingeniería.

- Una librería de patrones de re-ingeniería para transformar esquemas de clasificación, tesauros y lexicones en ontologías.

- Una librería de software, NOR$_2$O, que implementa las transformaciones sugeridas por los patrones de re-ingeniería para la construcción de ontologías.

Todo el marco propuesto en esta tesis permite acelerar el desarrollo de ontologías reduciendo así costes de tiempo y esfuerzo.

Acknowledgements

This thesis is the result of part of my work as a research assistant within the Ontology Engineering Group (OEG) at the Universidad Politécnica de Madrid, Spain. I want to mention that most of this work was performed in the context of two European Projects: the NeOn Project (FP6-027595), and the SEEMP Project (FP6-027347).

There are many people I want to thank since they kindly supported me in so many different ways for the successful completion of this thesis.

First of all, I would like to thank in a very special way Prof. Dr. Asunción Gómez-Pérez, my supervisor and guide during these years. In this period of my research, she gave me a fair amount of freedom to develop myself and, at the same time she encouraged and supported me, providing the necessary guidance to achieve the best results. Actually, she is the reason I decided to start my research in the semantic web field.

During the elaboration of this thesis, I found out that, in many cases, informal conversations with friends and colleagues were of great importance to clarify ideas. So, I want to thank all the members of the OEG group for the interesting discussions we had in all these years: Mari Carmen, Elena, Ándres, Ángel, José Ángel, Víctor, Raúl, Óscar, Lupe, Mariano, Jaime, Inma, among others. In addition, I would like to thank Rosario Plaza for helping me with the writing of the document. I also want to extend my gratitude to all the friends with whom I shared many wonderful moments during my staying in Madrid. I will always remember them, specially Fernando, Jan, Víctor, Elena, Mauricio, Alex, Jean Paul, Ángel, Óscar, Mari Carmen, José Ángel, Raúl, María, Freddy, Dani(s), Miguel Angel, Luis, Idafen, Carlos, Ghis, César, Rafa, Miguel, Jorge, Esther(s), José, Adrian, Ana, Sole, and José Alberto. Also to the OEG outsiders, Ester, Nicholas, Moisés, Mar, Miguel, David, Guillermo, Ana, Juan, Jezz, Olga, Miriam, Ivan, Miguelon, German, Marco, Edwin, María Cecilia, Yadira, Juani, Raziel, and Ernesto.

I want to thank the people involved in the NeOn and SEEMP projects: Caterina, Margarita, Marta, Jane, Yves, Soho, Claudio, Peter, Mathieu, Yimin, Martin, José Manuel, Tómas, German, Emanuelle, Irene, Salvatore, Luka, Jerome, Sofia, Dario, and Mick.

I must acknowledge Aldo Gangemi and the people I met during my research visit to the CNR in Italy: Valentina, Eva, Alessandro, Enrico, María, Alison, and Juani.

My sincere thanks are due to Michael Hausenblas and Richard Cyganiak, who gave me the opportunity to visit DERI in Ireland. Thanks also to Souleiman, Szymon, Antoine, Jürgen, Aidan, Axel, Nuno, Fadi, Lukas, Franco as well as the other foosball-mates. I also want to extend my thanks to my flat-mates Katie, Sarah, Myles, Khavin, Rajiv, Marc, and Martionus for standing by me through the good and the bad.

My special gratitude to my family in Spain: Magin, Paula, Isabel, Don Alfredo, Dra. Edith, Doña Betzy, Don Micky, Don José, Doña Charo, and Fabricio.

I want to thank my friends, Consuelo, Nestor, Omar, Juanjo, Beto, Nippur, Ronald, Mauricio, Gonzalo, Telma, Pomky, and the Asterix group because they gave me peace and confidence, especially in those moments of weakness for being so far from home and helped me to concentrate on the things that matter.

I would like to express my deepest thank to my brother, Javier, who always believed in me. Without his support I would not have been able to make this thesis come true. Finally, I owe my family my sincerest gratitude because they always supported and believed in me: my grandmother María Tráncito, my parents Demetrio and Chela, and my aunt Mardonia.

To all of them, very many thanks.

Agradecimientos

Esta tesis es el resultado de una parte de mi trabajo como investigador en el *Ontology Engineering Group*, OEG, de la Universidad Politécnica de Madrid. La mayor parte de este trabajo ha sido realizado en el contexto de los proyectos europeos NeOn (FP6-027595) y SEEMP (FP6-027347).

Me gustaría agradecer a muchas personas por el apoyo recibido para la culminación de esta tesis.

En primer lugar, quiero agradecer de manera especial a mi tutora Asunción Gómez-Pérez. Durante todo este tiempo de investigación, ella me concedió cierta libertad, pero al mismo tiempo, ella me ayudaba proporcionándome la guia necesaria para conseguir los resultados esperados. De hecho, gracias a ella me decanté a investigar en el campo de la Web Semántica.

A lo largo de la elaboración de esta tesis, he podido darme cuenta que en muchos casos las conversaciones con amigos y colegas fueron de gran importancia para clarificar algunas ideas. Por lo tanto, quiero agradecer a todos los miembros del grupo OEG por las discusiones interesantes que tuvismos durante todos estos años: Mari Carmen, Elena, Ándres, Ángel, José Ángel, Víctor, Raúl, Óscar, Lupe, Mariano, Jaime e Inma. Además, quiero agradecer a Rosario Plaza por ayudarme en la escritura del documento de tesis. Al mismo tiempo quiero agradecer a todos los amigos con los que compartí gratos momentos durante mi estancia en Madrid. Siempre los recordaré, especialmente a Fernando, Jan, Víctor, Elena, Mauricio, Alex, Jean Paul, Ángel, Óscar, Mari Carmen, José Ángel, Raúl, María, Freddy, Dani(s), Miguel Angel, Luis, Idafen, Carlos, Ghis, César, Rafa, Miguel, Jorge, Esther(s), José, Adrian, Ana, Sole y José Alberto. También a los amigos fuera de OEG, Ester, Nicholas, Moisés, Mar, Miguel, David, Guillermo, Ana, Juan, Jezz, Olga, Miriam, Ivan, Miguelon, German, Marco, Edwin, María Cecilia, Yadira, Juani, Raziel y Ernesto.

También quiero dar las gracias a todas las personas con las que trabajé en los proyectos NeOn y SEEMP, particularmente a Caterina, Margarita, Marta, Jane, Yves, Soho, Claudio, Peter, Mathieu, Yimin, Martin, José Manuel, Tómas, German, Emanuelle, Irene, Salvatore, Luka, Jerome, Sofia, Dario y Mick.

Agradecer afectuosamente a Aldo Gangemi y todas las personas con las que trabajé y conocí durante mi estancia en el CNR en Italia, a Valentina, Eva, Alessandro, Enrico, María, Alison, y Juani.

Mi más sincero agradecimiento a Michael Hausenblas y Richard Cyganiak, que me dieron la oportunidad de visitar DERI en Irlanda. Gracias a Souleiman, Szymon, Antoine, Jürgen, Aidan, Axel, Nuno, Fadi, Lukas, y los colegas del futbolín. También tengo que agradecer a mis compañeros de piso y amigos Katie, Sarah, Myles, Khavin, Rajiv, Marc, y Martionus. La estancia en Galway me enriqueció profesionalmente y personalmente.

Mi gratitud especial a mi familia en España: Magin, Paula, Isabel, Don Alfredo, Dra. Edith, Doña Betzy, Don Micky, Don José, Doña Charo, y Fabricio.

Quiero agradecer a mis amigos, Consuelo, Nestor, Omar, Juanjo, Beto, Nippur, Ronald, Mauricio, Gonzalo, Telma, Pomky, y los miembros del grupo Asterix. Ellos me brindaron paz y confianza, especialmente en aquellos momentos de debilidad por estar lejos de casa, que me ayudaron a concentrarme en las cosas que importan.

Quiero expresar mi agradecimiento más profundo a mi hermano, Javier, que siempre confió en mí. Sin su apoyo y paciencia no hubiera sido capaz de realizar esta tesis realidad. Finalmente, quiero agradecer a mi familia, que siempre confió en mí, mi abuelita María Tráncito, mis padres, Demetrio y Chela, y mi tía Mardonia.

Para todos ellos, muchísimas gracias.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Context

Ontologies are being used to model a domain of knowledge and to share information. They are found in knowledge engineering, artificial intelligence, computer science, and the Semantic Web, among others fields, as a form of knowledge representation of the world, or some part of it.

The word ontology is taken from Philosophy, where it means a systematic explanation of existence. In the field of Artificial Intelligence there are many definitions for it, a collection of which appears in [GPFLC03]. Neches [NFF$^+$91] was the first to define an ontology, which he did as follows: “Ontology defines the basic terms and the relations that include the vocabulary of a specific area, in addition to the rules to combine terms and relations to define extensions to the vocabulary”. Gruber [Gru93b, Gru93a] defines the ontology as “An explicit specification of a conceptualization”, being this definition the most referenced in the literature. Borst [Bor97] slightly modifies Gruber’s definition by saying that “Ontologies are defined as a formal specification of a shared conceptualization”. These last two definitions have been merged and explained by Studer et al. [SBF98] as “An ontology is a formal, explicit specification of a shared conceptualization. *Conceptualization* refers to an abstract model of some phenomenon. *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined. *Formal* refers to the fact that the ontology should be machine-readable. And *shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group”.

1.1.1 Overview of Some Methodologies for Building Ontologies

Research on Ontology Engineering methodologies has provided methods and techniques for developing ontologies from scratch. Well-recognized methodological approaches such as METHONTOLOGY [GPFLC03], On-To-Knowledge [SSSS01], and DILIGENT [PTS04] issue guidelines that help researchers to develop ontologies. However, researchers face an important limitation: no guidelines are provided for building ontologies by re-engineering some knowledge resources widely used within a particular community.

During the last decade, specific methods, techniques and tools were proposed for building ontologies from available knowledge resources. First, ontology learning methods and tools were proposed to extract relevant concepts and relations from structured, semi-structured, and non-structured resources [GPMM04, MS01] in order to form a single ontology. One important constraint of these methods and tools is that they propose *ad-hoc* solutions to transforming such resources, mainly texts, into ontologies. Hepp et al. [Hep06, HdB07, Hep07] stated that employing methods and techniques when ontologizing non-ontological resources to the level of ontologies is key for the success of semantic technology and this 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 literature presents a wide set of methods and tools for the ontologization of non-ontological resources. This ontologization of resources has led to the design of several specific methods, techniques and tools [HdB07, HVTS08, GGMO03, GC05]. These are mainly specific to a particular resource type, or to a particular resource implementation. Thus, every time ontology engineers are faced with a new resource type or implementation, they develop *ad-hoc* solutions to transforming such resource into a single ontology.

The analysis of the ontologies developed by distinct research groups in different international and national projects have revealed that there are different alternative ways or possibilities to build ontologies by reusing and re-engineering the available knowledge resources used by a particular community. However, at this stage we can state that all the projects perform an *ad-hoc* transformation of the resources available for building ontologies.

- Knowledge Web[1] deals with the aligning and versioning of ontologies as well as the use of best practices or patterns related to W3C activities.

- The SEKT[2] project focuses on argumentative development of ontologies.

- The UMLS Project[3] describes the experiences gained while transforming the UMLS Semantic Network into OWL ontology.

- The UK PRODIGY[4] describes the transformation of tangled hierarchies - those derived from ambiguous "broader than / narrower than" thesauri in library science - into formal ontologies.

[1] http://knowledgeweb.semanticweb.org

[2] http://www.sekt-project.org

[3] http://www.nlm.nih.gov/mesh/umlsforelis.html

[4] http://www.cks.nhs.uk/home

- The Knowledge Nets[5] project aims to investigate the impact of the Semantic Web technologies on electronic markets; one particular objective of this project is to build ontologies by reusing existing taxonomies for the description of skills as well as the classification of job profiles and industrial sectors within the job recruitment domain.
- Jimeno-Yepes et al. [JYJRBLRS09] explore how to use terminological resources for ontology engineering. They describe an approach for the proper creation and use of a shared thesaurus in the development of ontologies. They have applied their approach to a real scenario, the Health-e-Child (HeC) project[6] and have evaluated the impact of filtering and re-organizing several resources.
- The e-POWER project[7] aims to integrate heterogeneous components by means of a semantically-enhanced middleware, which operates between the portal and the web services interfacing the functionalities of back-offices.

As it can be inferred from above, a new ontology development paradigm started approximately in 2007, whose emphasis was on the **reuse and possible subsequent re-engineering of knowledge resources**, as opposed to custom-building new ontologies from scratch. However, in order to support and promote such reuse-based approach, new methods, techniques, and tools are needed.

1.1.2 Non-ontological Resources

The knowledge resources, reused in the aforementioned projects for building ontologies, contain, readily available, a wealth of category definitions and reflect some degree of community consensus. In this thesis, we refer to **non-ontological resources (NOR)**[8] as those knowledge resources whose semantics have not yet been formalized explicitly by means of ontologies. Examples of NORs are classification schemes, thesauri, lexica, and folksonomies, among others. This type of resources encodes different types of knowledge and can be implemented in different ways.

Our analysis of the literature has revealed different ways of categorizing non-ontological resources. Thus Maedche et al. [MS01] and Sabou et al. [SAd^{+}07] classify non-ontological resources into unstructured (e.g. free text), semi-structured (e.g. folksonomies) and structured (e.g. databases) resources; whereas Gangemi et al. [GPS98] distinguish catalogues of normalized terms, glossed catalogues, and taxonomies. Finally, Hodge [Hod00] proposes characteristics such as structure, complexity, relationships among terms, and historical functions for classifying them. However, an accepted and agreed upon typology of non-ontological resources does not exist yet.

[5] http://wissensnetze.ag-nbi.de/

[6] http://www.health-e-child.org/

[7] http://lri.jur.uva.nl/~epower/

[8] Along this thesis we use either *NOR* or *Non-ontological resource* without distinction.

As mentioned before, an ontology [SBF98] has to reflect the notion of capturing consensual knowledge. Capturing this consensual knowledge is not an easy task to accomplish, especially if the ontology is built from scratch.

Furthermore, non-ontological resources usually contain terminology already agreed upon by a broad community of people, who have a given protocol for that. So, at least the labels used for naming terms are agreed on by consensus. Therefore, it is important to (1) select the appropriated resources, and (2) transform them into ontologies. In this way, the ontologies generated will have reached consensus within the ontological community. In addition, this reuse and possible subsequent re-engineering of existing resources will also speed up the ontology development; therefore, we will save time, effort and resources.

Along with this thesis we will work with non-ontological resources, specifically classification schemes, thesauri and lexica.

1.1.3 NeOn Methodology for Building Ontology Networks

Starting in 2007, the NeOn project[9] has made some noteworthy contributions to ontological engineering; however, the most important one is the design of the NeOn Methodology [SF10, GPSF09], which includes the definition of a new ontology development process, life cycle models based on methods, as well as the techniques and tools to be used during the ontology building.

The NeOn Methodology is a scenario-based methodology. It proposes nine scenarios for building ontology networks collaboratively, emphasizing the reuse and re-engineering of knowledge resources. The identified scenarios that may arise during the ontology (network) development are the following:

- *Scenario 1:* From specification to implementation.
- *Scenario 2:* Reusing and re-engineering non-ontological resources.
- *Scenario 3:* Reusing ontological resources.
- *Scenario 4:* Reusing and re-engineering ontological resources.
- *Scenario 5:* Reusing and merging ontological resources.
- *Scenario 6:* Reusing, merging and re-engineering ontological resources.
- *Scenario 7:* Reusing ontology design patterns.
- *Scenario 8:* Restructuring ontological resources.
- *Scenario 9:* Localizing ontological resources.

Figure 1.1 presents a set of the nine identified scenarios for building ontologies and ontology networks. The directed arrows with numbered circles associated represent the different scenarios. Each scenario is decomposed into different processes or activities that are represented with coloured circles or with rounded boxes. Such processes and activities are defined in the NeOn Glossary of Activities [SFGP08]. The figure also shows (as dotted boxes) the available knowledge resources to be reused and possible outputs (implemented ontology networks and alignments) resulting from the execution of some of the scenarios presented.

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

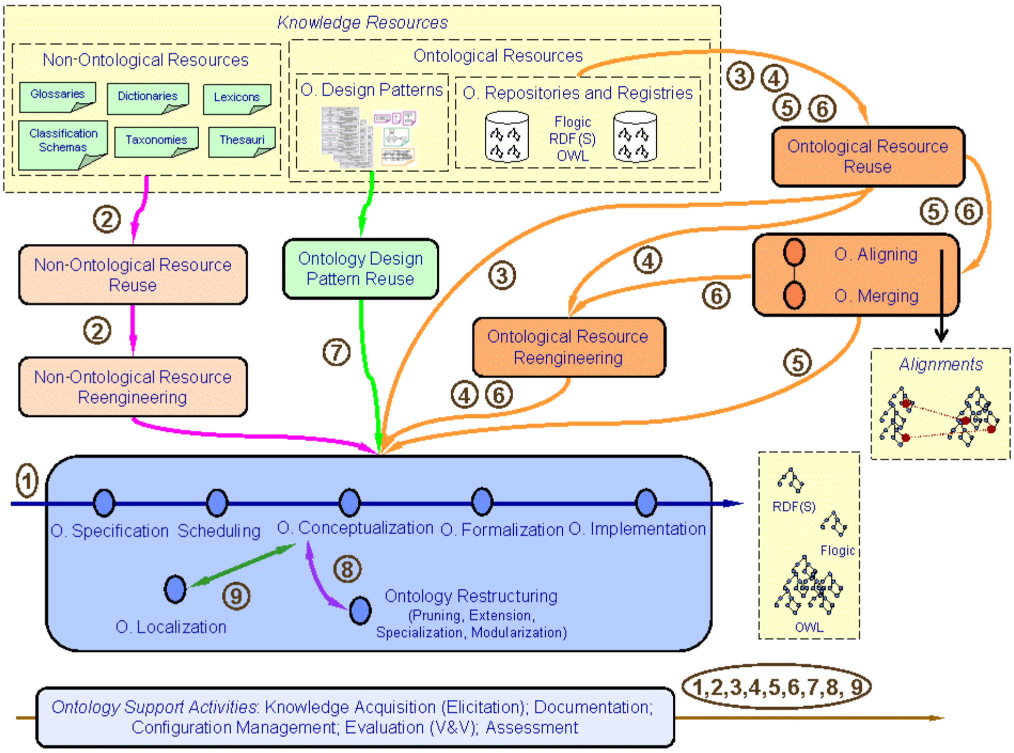

Knowledge Resources
Non-Ontological Resources
Glossaries
Dictionaries
Lexicons
Classification Schemas
Taxonomies
Thesauri
Ontological Resources
O. Design Patterns
O. Repositories and Registries
Flogic
RDF(S)
OWL
③ ④
⑤ ⑥
Ontological Resource Reuse
⑤ ⑥
②
Non-Ontological Resource Reuse
Ontology Design Pattern Reuse
③
④
⑥
O. Aligning
O. Merging
②
Non-Ontological Resource Reengineering
⑦
Ontological Resource Reengineering
④ ⑥
⑤
Alignments
①
O. Specification
Scheduling
O. Conceptualization
O. Formalization
O. Implementation
RDF(S)
Flogic
OWL
⑧
⑨
O. Localization
Ontology Restructuring
(Pruning, Extension, Specialization, Modularization)
1,2,3,4,5,6,7,8, 9
Ontology Support Activities: Knowledge Acquisition (Elicitation); Documentation; Configuration Management; Evaluation (V&V); Assessment

Figure 1.1: Scenarios for building ontologies and ontology networks [SF10]

The scope of this thesis is to propose novel methods, techniques and tools for supporting scenario 2, which reuses and re-engineers non-ontological resources for building ontology networks.

1.1.4 Patterns in Ontology Engineering

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. They provide a general solution that has to be applied in a particular context, in which the design considerations serve to decide whether the pattern is useful and how it could be implemented best [EPJ06]. A kind of software patterns are the re-engineering software patterns [PS98]. These patterns 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, describe a process of re-engineering without proposing a complete methodology, and sometimes can suggest which type of tool to use.

In the ontology engineering community the idea of applying patterns for modelling ontologies was proposed by Peter Clark [CTP00]. Since then, relevant works on patterns have appeared, such as the Semantic Web Best Practices and Deployment Working Group[10], the Ontology Design Patterns Public Catalog[11], the Ontology Design Patterns (ODP) Portal[12], and the Linked Data Patterns[13], which is a catalogue of Linked Data [Biz09] patterns. According to Presutti et al. [GP08] Ontology Design Patterns are modelling solutions used to solve a recurrent ontology design problem. They distinguish different types of Ontology Design Patterns by grouping them into six families. Each family addresses different kinds of problems and can be represented with different levels of formality. ODPs reduce the effort of building ontologies.

As stated in the section 1.1.1, the re-engineering methods, techniques and tools that ontologize non-ontological resources are mainly specific to a particular resource type or to a particular resource implementation. Along this thesis we propose a set of re-engineering patterns for transforming available non-ontological resources, which have reached some degree of consensus, into ontologies. Also, we will try to demonstrate that the use of re-engineering patterns for transforming non-ontological resources into ontologies has several advantages: patterns (1) embody expertise in guiding a re-engineering process, (2) improve the efficiency of the re-engineering process, (3) make the transformation process easier for ontology engineers, and (4) speed up the ontology development process.

Thus, in this thesis we propose methodological guidelines to address these research problems, as well as a set of patterns that make explicit the hidden transformation decisions in the conversion scripts used in the *ad-hoc* approaches. Hence, it will be easier for ontology engineers to (1) reuse the hidden transformation decisions according to the type and implementation of the resources, and (2) perform the transformation of the resources into ontologies by saving time and effort.

Therefore, here we propose a model as well as methods and tools for transforming non-ontological resources into ontologies by using re-engineering patterns.

1.2 Thesis Structure

The remainder of the thesis proceeds as follows:

- Chapter 2 describes the **state of the art** of the topics of interest in this work. For each topic we analyze the limitations and open research problems, emphasizing those to which we provide solutions.

- Chapter 3 provides a presentation of the **objectives and main contributions** of the thesis. Because of the limitations found in the state of the art, we describe first the overall objectives of the thesis and then the specific ones.

[10] http://www.w3.org/2001/sw/BestPractices/OEP/

[11] http://www.gong.manchester.ac.uk/odp/html/index.html

[12] http://ontologydesignpatterns.org

[13] http://patterns.dataincubator.org/book/

Then, we introduce our contributions to the current state of the art, followed by the presentation of the **assumptions, hypotheses and restrictions** of this work.

- Chapter 4 explains the **research methodology** followed when designing the method for reusing and re-engineering non-ontological resources into ontologies, and the general framework for describing such a method.
- Chapter 5 presents our contribution to the topic of **reusing non-ontological resources**. First we provide a definition of the non-ontological resources. Then, we present a categorization of non-ontological resources. Finally, we describe the methodological guidelines proposed for the non-ontological reuse process.
- Chapter 6 introduces a general **method for re-engineering non-ontological resources into ontologies**. We start by introducing the model for re-enginee-ring non-ontological resources with re-engineering patterns. Then, we describe the methodological guidelines proposed for the non-ontological resource re-engineering process.
- Chapter 7 presents the **patterns for re-engineering classification schemes into ontologies**. First we define classification schemes and describe their main characteristics. Then we depict the set of patterns we have created for transforming classification schemes into ontologies.
- Chapter 8 describes the **patterns for re-engineering thesauri into ontologies**. We start by defining thesauri and describing their main characteristics. Then we present the set of patterns we have created for transforming thesauri into ontologies.
- Chapter 9 presents the **patterns for re-engineering lexica into ontologies**. First we define lexica and describe their main characteristics. Then we depict the set of patterns we have created for transforming lexica into ontologies.
- Chapter 10 describes the **technological support** for the model and method proposed. First, we present the software library, NOR$_2$O, that carries out the transformation process suggested by the patterns. Next, we depict a pattern library that includes the set of patterns for re-engineering non-ontological resources into ontologies.
- Chapter 11 is dedicated to **evaluation**. We have divided this chapter into two parts. The first one describes the evaluation, which is focused on all the methodological aspects related to the reuse and re-engineering of non-ontological resources for building ontologies. The second part deals with the evaluation of the technology.
- Chapter 12 provides the **conclusions** and **future lines of work**.

1.3 Dissemination of Results

To conclude the introduction, it is important to remark that parts of this thesis have been internationally disseminated.

A summary of the whole thesis has been published in

B. Villazón-Terrazas, M. C. Suárez-Figueroa, and A. Gómez Pérez. "A Pattern-Based Method for Re-engineering Non-Ontological Resources into Ontologies". International Journal on Semantic Web and Information Systems (IJSWIS). Amit Sheth (Ed.) (Kno.e.sis Center, Wright State University, USA) Volume 7 (2010).

Our contribution presented in Chapter 5 has been partially published in:

- A. Gómez-Pérez, J. Ramírez and B. Villazón-Terrazas. "Reusing Human Resources Management Standards for Employment Services". Proceedings of the Workshop on First Industrial Results of Semantic Technologies, co-located with ISWC 2007 + ASWC 2007, Busan, Korea, November 11th, 2007.
- A. Gómez-Pérez, J. Ramírez and B. Villazón-Terrazas "An Ontology for Modelling Human Resources Management based on standards". In: 11th International Conference on Knowledge-Based Intelligent Information & Engineering Systems, 12-14 September, 2007, Vietri sul Mare, Italy.
- A. Gómez-Pérez, J. Ramírez and B. Villazón-Terrazas. "Methodology for Reusing Human Resources Management Standards". In: 19th International Conference on Software Engineering and Knowledge Engineering, 9-11 July, 2007, Boston, USA.

Some of the contributions of Chapters 6, 7, 8, 9, and 10 have been published in:

- A. Garcia-Silva, A. Gómez-Pérez, M.C. Suárez-Figueroa, B. Villazón-Terrazas (2008). "A Pattern Based Approach for Reengineering Non-ontological Resources into Ontologies". In ASWC 08: Proceedings of the 3rd Asian Semantic Web Conference on the Semantic Web (pp. 167181). Berlin, Heidelberg : Springer-Verlag.

Contributions of Chapter 10 have been published in:

- B. Villazón-Terrazas, A. Gómez Pérez, and J.P. Calbimonte. " NOR$_2$O: a Library for Transforming Non-Ontological Resources to Ontologies ". Extended Semantic Web Conference, ESWC 2010, Greece.
- B. Villazón-Terrazas, M. C. Suárez-Figueroa, and A. Gómez Pérez. "Pattern for Re-engineering a Term-based Thesaurus, which Follows the Record-based model, to a Lightweight Ontology". Workshop on Ontology Patterns

(WOP 2009). International Semantic Web Conference (ISWC 2009). Washington D.C. USA.

- B. Villazón-Terrazas, M. C. Suárez-Figueroa, and A. Gómez Pérez. "Pattern for Re-engineering a Classification Scheme, which Follows the Path Enumeration Data Model, to a Taxonomy". Workshop on Ontology Patterns (WOP 2009). International Semantic Web Conference (ISWC 2009). Washington D.C. USA.

- B. Villazón-Terrazas, M. C. Suárez-Figueroa, and A. Gómez Pérez. "Pattern for Re-engineering a Classification Scheme, which Follows the Adjacency List Data Model, to a Taxonomy". Workshop on Ontology Patterns (WOP 2009). International Semantic Web Conference (ISWC 2009). Washington D.C. USA.

Part of the contribution presented in Chapter 11 has been partially published in:

- **B. Villazón-Terrazas, J. Ramírez, M. C. Suárez-Figueroa, and A. Gómez Pérez. "A Network of Ontology Networks for building e-Employment Advanced Systems". International Journal on Expert Systems with Applications (ESWA). J. Liebowitz (Ed.) ELSEVIER. (2011).**

- A. de León, V. Saquicela, L.M. Vilches, B. Villazón-Terrazas, F. Priyatna, and O. Corcho. "Geographical linked data: a Spanish use case". In Proceedings of the 6th international Conference on Semantic Systems (Graz, Austria, September 01 - 03, 2010). A. Paschke, N. Henze, and T. Pellegrini, Eds. I-SEMANTICS '10. ACM, New York, NY, 1-3.

- I. Celino, D. Cerizza, M. Cesarini, E. Della Valle, F. De Paoli, J. Estublier, M. Grazia Fugini, A. Gómez Pérez, M. Kerrigan, P. Guarrera, M. Mezzanzanica, J. Ramirez, B. Villazón-Terrazas, and G. Zhao. "SEEMP: A Networked Marketplace for Employment Services". In Vassilios Peristeras, Tomas Vitvar, and Konstantinos Tarabanis (eds.) Semantic Technologies for E-Government, Springer, 2009.

- E. Della Valle, D. Cerizza, I. Celino, M. Grazia Fugini, J. Estublier, G. Vega, M. Kerrigan, A. Gómez Pérez, J. Ramírez, B. Villazón-Terrazas, G. Zhao, M. Cesarini, and F. De Paoli. "The SEEMP Approach to Semantic Interoperability for e-Employment". In Robeto García (ed.): Semantic Web Methodologies for E-Business Applications: Ontologies, Processes and Management Practices, (IDEA Group Publishing), 2008.

The ontology built within the SEEMP project[14] by applying the methodological guidelines introduced in this thesis, which appears in Chapter 11, was registered

[14] http://www.seemp.org

as a patent in the *Registro de la Propiedad Intelectual “Comunidad de Madrid”*. Reference M-404/2009.

Chapter 2

STATE OF THE ART

In this chapter we present an exhaustive analysis of the state of the art of the topics of interest covered in this thesis and a discussion on the limitations of the research works on the state of the art. We start with the presentation of a framework for comparing the methods and tools that permit reusing and re-engineering non-ontological resources when building ontologies. Then, we provide the background on patterns for re-engineering. Finally, we conclude by summarizing the limitations found in the state of the art.

2.1 A Comparative Framework of Methods and Tools for Reusing and Re-engineering NORs into Ontologies

In this section we present a comparative study of the most outstanding methods and tools for reusing and re-engineering non-ontological resources into ontologies. To carry out this study we have established a common framework with which to compare the main characteristics of the different methods and tools.

The section is organized as follows: Section 2.1.1 introduces the framework for evaluating the methods and tools employed when reusing non-ontological resources and re-engineering them into ontologies. Section 2.1.2 describes the methods for reusing and re-engineering non-ontological resources. Section 2.1.3 depicts the tools available for this purpose. Finally, Section 2.1.4 presents the results and conclusions of the methods and tools evaluated in this comparative study.

2.1.1 Evaluation Framework

In this section we set up a framework for comparing the methods and tools required to reuse non-ontological resources and re-engineer them into ontologies. The framework is organized in four dimensions (the non-ontological resource, the reuse process, the transformation process, and the resultant ontology) that gather several features with which to compare the approaches proposed in the literature. The first dimension analyses the features of a NOR; the second covers the selection of the NORs; then, the third is focused on the transformation process, whereas the fourth is centred on the features of the resultant ontology.

2.1.1.1 Features of the non-ontological resource

In this thesis we propose a categorization of non-ontological resources, according to three different features presented in Figure 2.1: (1) the type, which refers to the kind of inner organization of the information; (2) the data model, that is, the design data model used to represent the knowledge encoded by the resource; and (3) the resource implementation.

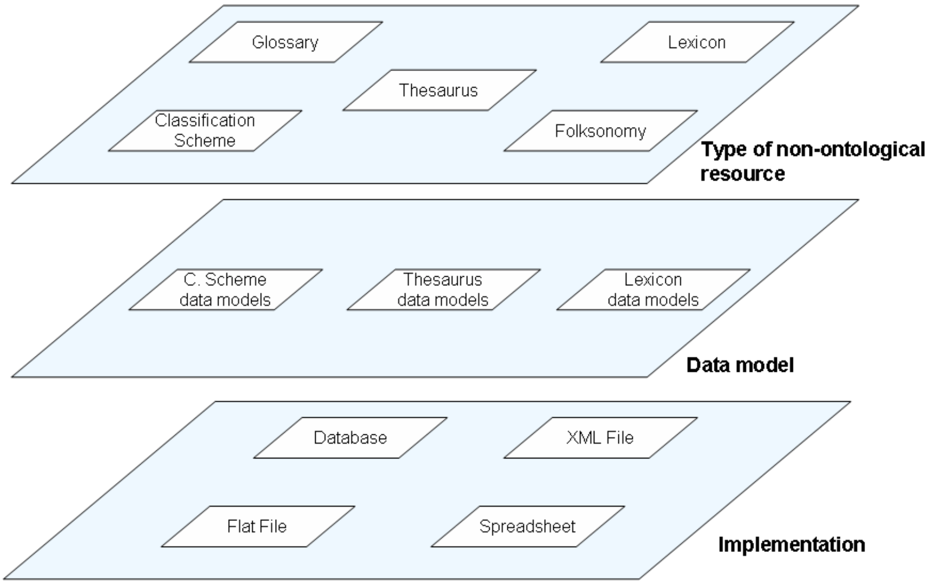


Figure 2.1: Non-ontological resources categorization

- With respect to the ***type of non-ontological resources*** we classify them into (1) glossaries, (2) lexicons, (3) classification schemes, (4) thesauri, and (5) folksonomies.

- The ***data model*** [Car02] is the abstract model that describes how data is represented and accessed. It can be different even for the same type of non-ontological resource. Besides, it is an important artefact for the re-engineering process, because it helps to understand how the resource information is organized [GGPSFVT08]. In the following chapters we present several data models for each of the non-ontological resources.

- With regard to the ***implementation*** we can classify non-ontological resources into (1) databases, (2) XML files, (3) flat files, and (4) spreadsheets.

2.1.1.2 Features of the reuse process

- The research work provides some ***methodological guidelines*** that support the selection of the resources to be transformed.

- The reuse process is supported by a ***tool or a set of tools***.

- The research work keeps track of the ***provenance***[1] of the resource, i.e., a reference to the non-ontological resource for every ontology generated.

2.1.1.3 Features of the transformation process

- The transformation process may follow one of the following approaches: (1) ***ABox transformation*** [CHPG09], which transforms the resource schema into an ontology schema, and the resource content, into ontology instances; (2) ***TBox transformation*** [CHPG09], which transforms the resource content into an ontology schema; or (3) ***Population***, which transforms the resource content into instances of an available ontology. Figure 2.2 depicts each of the possible transformations. The ABox transformation leaves the informal semantics of the transformed resources mostly untouched, whereas, the TBox transformation tries to enforce a formal semantics into them.

- The research work performs the transformation either (1) on the ***syntactic level***; or (2) on the ***semantic level***. The syntactic level deals with the ability to structure the representation in structured sentences, formulas or assertions. This level includes the transformations of resource component definitions according to the grammars of the source and target formats [Cor05], in other words, it includes a structure-preserving transformation that should reflect the resource structure as closely as possible. The semantic level deals with the ability to construct the propositional meaning of the representation [Cor05], which basically is a specific interpretation of the non-ontological resource.

- The research work makes explicit the semantics hidden in the relations of the non-ontological resource terms, e.g., ***subClassOf***, ***partOf***.

- The research work relies on (1) ***additional resources***, or (2) ***a human domain expert*** for making explicit the semantics hidden in the relations of the NOR terms.

- The transformation process can be (1) ***automatic***, (2) ***semi-automatic***, and (3) ***manual***.

[1] Provenance focuses on describing and understanding where and how data is produced, actors involved in its production, and processes applied before the arrival of data to the collection from which it is now accessed [GPC08].

- The research work provides some ***methodological guidelines*** that support the transformation process.
- The list of the ***techniques employed*** in the transformation process are clearly identified: mapping rules, re-engineering patterns, etc.
- The transformation process is supported by a ***tool or a set of tools***.

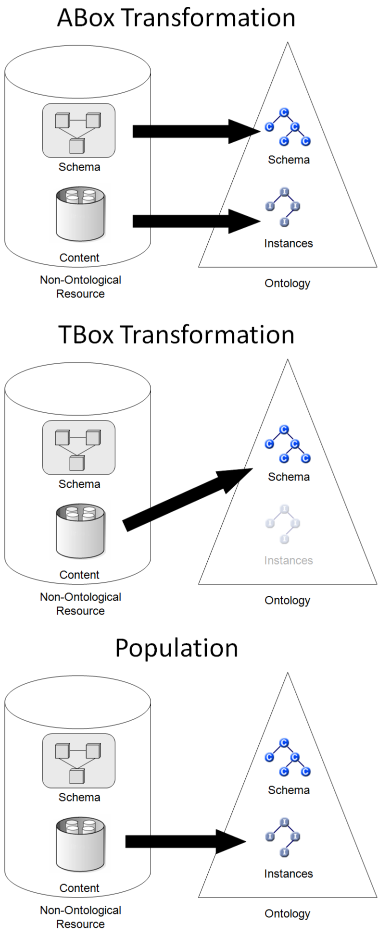


Figure 2.2: Transformation approaches

2.1.1.4 Features of the resultant ontology

- The ontology generated is either ***lightweight*** or ***heavyweight***.
- The ***components*** of the ontology generated are classes, attributes, relations, or instances.

- The ***ontology implementation language*** is OWL or RDF(S).
- The research work generates a ***single ontology*** or ***several ontologies***. However, we do not distinguish whether the ontologies generated are interconnected or not.

2.1.2 Methods for Reusing and Re-engineering Non-ontological Resources

In this section, we describe the most significant methods for reusing and re-engineering non-ontological resources taking into account the features previously identified in the framework. To do this, we analyse the literature from two complementary perspectives: First, in Section 2.1.2.1 we describe the methods for building ontologies by means of transforming different types of resources (classification schemes, thesauri, lexicons and folksonomies). Second, in Section 2.1.2.2 we analyze the methods based on the implementation of the resources (database, XML, flat file and spreadsheet).

2.1.2.1 Methods centred on the non-ontological resource type

In this section we present the most outstanding methods on reusing and re-engineering non-ontological resources. Specifically, we summarize methods for building ontologies from classification schemes, folksonomies, lexica and thesauri.

Methods for building ontologies from classification schemes The two main methods for transforming classification schemes are the GenTax [HdB07] method and Hakkarai-nen et al's method [HHST06].

- **GenTax** is a method presented by Hepp et al. [HdB07] for deriving semi-automatically consistent RDF(S) and OWL ontologies from hierarchical categorization schemas. For this method, a hierarchical categorization schema can be a taxonomy, a thesaurus, or a hierarchical classification, and any of them has to be implemented in a database. The three types of resources have in common the inclusion of a set of categories and some form of hierarchical order. This method does not take into consideration the data model of the input resource.

  Gentax assumes that the non-ontological resource is already selected for its transformation; therefore, it provides neither methodological guidelines nor tools for the reuse process. Moreover, GenTax does not manage the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

  This method performs a semi-automatic TBox transformation, considering the syntactic and semantic levels. It makes explicit the semantics of the relations of the NOR categories. It also sets an *ad-hoc* and some taxonomic relations among the NOR categories. It provides methodological guidelines for the transformation, but does not clearly identify the techniques employed in the transformation. The transformation process is supported by a tool, SKOS2GenTax, which consists of a Java program, that accesses the NOR categories via an ODBC link.

Finally, the method produces a single lightweight ontology in OWL DLP or RDF(S). The ontology components generated are classes and relations.

- **Hakkarainen et al.** [HHST06] present a study of the semantic relationship between the ISO 15926-2[2] and OWL DL. The ISO 15926 is a standard for integrating, sharing, exchanging, and handing-over data between computer systems. The ISO 15926-2 is built on EXPRESS[3] and stored in a flat file. This standard consists of 201 entity data types; the top level entity data type is *thing*, with its subtypes of a *possible_individual* and *abstract_object*. All other entities are subtypes of them.

  The method presented by Hakkarainen et al. transforms a specific non-ontological resource, the ISO 15926-2 standard, but it provides neither methodological guidelines nor tools for the reuse process. Moreover, the method does not keep the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

  This method consists of (a) two transformation protocols, which are based on transformation rules, and (b) two inverse transformation protocols, which have the purpose of examining the possible loss of semantics. The method peforms a semi-automatic TBox transformation and considers the syntactic and semantic levels. It translates the whole set of subtype relations into *subClassOf* relations. Additionally, it provides a set of methodological guidelines for the transformation and relies on transformation rules as the technique employed. However, not a single tool supporting the method is mentioned.

  The transformation protocols generate a lightweight single ontology in OWL DL. The ontology components generated are classes, attributes, and relations.

Methods for building ontologies from folksonomies The two main methods for transforming folksonomies are T-ORG [ASC07], developed by Abbasi et al., and the one developed by Maala et al. [MDA07].

- **Abbasi et al.** [ASC07] present a mechanism to transform a set of tags of a given folksonomy into instances of an available ontology. These authors, however, do not mention where tags are stored.

[2] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=29557

[3] 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

Their method assumes that non-ontological resource is already selected for its transformation, so it provides neither methodological guidelines nor tools for the reuse process. Moreover, the method does not consider the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

The method consists in (1) selecting the ontologies relevant to the tags by means of Swoogle[4]; (2) pruning and refining the ontology; and (3) classifying the tags with lexico-syntactic patterns. It performs an automatic population, taking into consideration the syntactic level. However, it does not make explicit the semantics of the relations of the NOR categories, though it provides methodological guidelines for the transformation. The T-ORG tool, described in section 2.1.3, gives support to this method.

This method populates several lightweight ontologies, i.e., the ontology components generated are instances, although it does not mention the ontology language used.

- **Maala et al.**'s method [MDA07] describes a conversion process from Flickr[5] tags to RDF descriptions. These authors present a method for automatically converting a set of tags into a RDF description in the context of photos on Flickr. It must be observed, however, that they do not mention where the Flickr tags are stored.

  This method transforms a specific non-ontological resource, Flickr tags, but it provides neither methodological guidelines nor tools for the reuse process. Nor does the method take into account the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

  Its authors analyse the tagging habits and the tagging content of the photos. To accomplish this, they rely on additional resources for the conversion such as (1) WordNet, which has been completed with extra information, (2) a database containing geographical locations and (3) an ontology of things. The method performs an automatic population, though it only considers the syntactic level. However, it does not make explicit the semantics of the relations of the NOR elements, although it provides methodological guidelines for the transformation. Nor does it clearly identify the techniques employed in the transformation. In addition, not a single tool supporting the method is mentioned.

  This method populates a lightweight single ontology and the ontology components generated are instances, which are expressed in RDF.

[4] http://swoogle.umbc.edu

[5] http://www.flickr.com/

Methods for building ontologies from lexica The two main methods for transforming lexica are presented in [vAGS06] and [GNV03, GGMO03] and both are focused on WordNet.

- The method of **van Assem et al.** [vAGS06] proposes a standard conversion of WordNet [Fel98] into the RDF/OWL representation language. This method employs version 2.0 of Princenton's WordNet Prolog distribution[6], which contains eighteen files: one file represents synsets, word senses and words, and the remaining seventeen represent their relationship. This method takes into account the internal data model of the lexicon, and devises how the lexicon data is represented and accessed for the transformation. It also provides resource provenance information, so the resultant ontology keeps the reference to WordNet. However, it provides neither methodological guidelines nor tools for the reuse process.

  In this method the authors include a process for designing the conversion of the resource, as well as a set of the requirements for the conversion. Some of the requirements include the following recommendations: (a) the conversion should fully transform WordNet into RDF/OWL; (b) the conversion should be convenient to carry out; (c) the conversion should reflect as much as possible the original structure of WordNet; and (d) the conversion should provide OWL semantics while still being interpretable by pure RDF(S) tools.

  Basically, the method consists in (1) creating a set of classes for each of the main components of WordNet: classes for every word, synset and sense; (2) modelling words, synsets and senses belonging to WordNet as instances of the previously created classes; and (3) coding part/s of the semantics related to each instance by means of the URIs used to identify each instance.

  The method performs a semi-automatic ABox transformation, considering the syntactic level, but it does not make explicit the semantics of the relations of the NOR elements. However, it provides methodological guidelines for the transformation although it does not clearly identify the techniques employed in the transformation. The transformation process is supported by Swi-Prolog[7] tool.

  This method produces a single lightweight ontology in RDF(S)/OWL Full. The ontology components generated are classes, attributes, relations, and instances.

- **Gangemi et al.** [GNV03, GGMO03] present a method that explains how WordNet information can be bootstrapped, mapped, refined and modularized. It employs WordNet 1.6, which is stored in relational databases. It also takes into account the internal data model of the resource and devises how the lexicon data is represented and accessed for the transformation.

[6] `http://wordnet.princeton.edu/obtain`

[7] `http://www.swi-prolog.org/packages/semweb.html`

This method assumes that the non-ontological resource is already selected for its transformation; however, it provides neither methodological guidelines nor tools for the reuse process. Nor does it keep the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

This is a hybrid method because it employs top-down techniques and tools from formal ontologies and bottom-up techniques from computational linguistics and machine learning. It can automatically extract association relations from WordNet elements and interpret those associations in terms of a set of conceptual relations, formally defined in the DOLCE[8] ontology.

The method consists in (1) bottom-up learning of association links (A-links), in which WordNed glosses are analysed, and A-links between a synset and the synsets in its gloss are created; and(2) top-down learning, in which the DOLCE ontology is used to interpret A-links in terms of axiomatic conceptual relations.

The method performs a semi-automatic TBox transformation, regarding the syntatic and semantic levels. Moreover, it makes explicit the semantics of the relations of the WordNet concepts (originally called synsets) by applying natural language techniques to their glosses and then using DOLCE ontology for making explicit the semantics of the ad-hoc relations (A-links). It provides methodological guidelines for the transformation. However, not a single tool supporting the method is mentioned.

The method generates a single lightweight ontology implemented in DAML+OIL. The components of the ontology generated are classes, attributes, and relations.

Methods for building ontologies from thesauri The six main methods for transforming thesauri are presented in [Hah03, HS03], [vAMSW04], [vAMMS06], [WSWS01],[HVTS08],[SLL$^+$04, LS06].

- **Hahn et al.** [Hah03, HS03] present a method that extracts conceptual knowledge from an informal medical thesaurus, UMLS[9], which is stored in ASCII files, and semi-automatically converts this conceptual knowledge into LOOM[10]. This method takes into account the internal data model of the thesaurus.

  The method assumes that the non-ontological resource, UMLS, is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Besides, it ignores the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

[8] `http://www.loa-cnr.it/DOLCE.html`

[9] `http://www.nlm.nih.gov/research/umls/`

[10] `http://www.isi.edu/isd/LOOM/`

Its authors formalize a model of partonomic reasoning that does not exceed the expressiveness of the well-understood concept language ALC[11]. Hahn et al. aim to extract conceptual knowledge from two major sub-domains of the UMLS, anatomy and pathology, in order to construct a formally sound knowledge base founded on an/the ALC-type description logic language.

This method performs a semi-automatic TBox transformation and takes into account the syntactic and semantic levels. It makes explicit the semantics of the relations of the UMLS elements by means of a biomedical domain expert. It treats the *partOf/hasPart*, *isA*, *siblingOf*, and *associatedWith* relations. It also introduces methodological guidelines for performing the transformation of the resource and relies on Ontology Design Patterns for partonomic relations. Besides, it utilizes a LOOM classifier for automatic consistency checking of the ontology generated.

This method produces a single heavyweight ontology expressed in formal description logics, LOOM. The ontology components generated are classes and relations.

- **Van Assem et al.** [vAMSW04] present a method for converting thesauri from their native format to RDF(S) and OWL Full. The method deals with resources implemented in (1) a proprietary text format, (2) a relational database, and (3) an XML representation. However, it ignores the internal data model of the resource.

  This method provides neither methodological guidelines nor tools for the reuse process. And nor does it take into account the resource provenance information; therefore, the resultant ontology does not keep the reference to the non-ontological resource.

  The method consists of (1) preparation, in which the following characteristics of the thesaurus are analysed: conceptual model, relation between the conceptual and implementation model, and relations to standards; (2) syntactic conversion, which includes a structure-preserving translation and explication of the syntax of the resource; (3) semantic conversion, which includes the explication of semantics and specific interpretation of the thesaurus; and (4) standardization, which is an optional step for mapping a thesaurus onto a standard schema.

  Additionally, this method performs a semi-automatic TBox transformation, taking into account the syntactic and semantic levels. It makes explicit the semantics of the relations of the thesaurus terms by means of a domain expert. Besides, it considers the subClassOf and ad-hoc relations. It also provides methodological guidelines for performing the transformation of the thesaurus though it provides neither information about the techniques employed nor a tool to support the method.

[11] ALC allows for the construction of concept hierarchies.

This method produces a single lightweight ontology in RDF(S)/OWL Full. The ontology components generated are classes, attributes, and relations.

- **Van Assem et al.** [vAMMS06] present a method for converting thesauri into the SKOS [MB05] RDF/OWL schema. This SKOS schema is a W3C recommendation developed by the W3C Semantic Web Best Practices Working Group. The method provides neither information about the format of the thesaurus, nor its internal data model. It is worth mentioning that the authors provide a description of the development of the method and of the method itself.

  The development of this method is based on a process that has the following tasks: (1) defining the goal and requirements of the method; (2) comparing the available methods for transforming thesauri into ontologies; (3) developing the steps of the method (for this task the authors relied on the method of Miles et al. [Mil05]); (4) applying the method to IPSV[12], GTAA[13] and MeSH[14] thesauri; and (5) evaluating the method.

  The method provides neither methodological guidelines nor tools for the reuse process. And it ignores the resource provenance information; therefore, the resultant ontology does not keep the reference to the non-ontological resource.

  In a nutshell, the steps of the method are (1) to analyse the implementation and the documentation of the resource; (2) to define mappings between input data items and output SKOS RDF; and (3) to develop an algorithm for the transformation program.

  This method performs an automatic population, i.e., creates instances of the SKOS schema, taking into account the syntactic level. However, it does not make explicit the semantics of the relations of the thesaurus terms. In addition, it provides methodological guidelines for performing the transformation of the thesaurus though it does not provide information about the techniques employed. It relies on an *ad-hoc* tool for performing the transformation.

  This method populates a single lightweight ontology. The ontology components generated are instances expressed in SKOS RDF.

- **Wielinga et al.** [WSWS01] present a method for transforming the Art and Architecture Thesaurus (AAT) into an RDF(S) ontology. The AAT is the most elaborate and standardized body of knowledge concerning classifications of art objects. AAT is published via a searchable online Web interface[15]

[12] Integrated Public Sector Vocabulary http://www.esd.org.uk/standards/ipsv/

[13] Common Thesaurus for Audiovisual Archives http://informatieprofessional.googlepages.com/gtaa

[14] Medical Subject Headings http://www.nlm.nih.gov/mesh/

[15] http://www.getty.edu/research/conducting_research/vocabularies/aat/

and is also available in XML files. This method takes into consideration the internal data model of the ATT thesaurus.

Since the method transforms a specific non-ontological resource, the ATT thesaurus, it provides neither methodological guidelines nor tools for the reuse process. Moreover, the method does not consider the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

Basically, the method consists in: (1) converting the full ATT hierarchy into a hierarchy of concepts; (2) augmenting a number of concepts with additional attributes; and (3) adding knowledge about the relation between possible values of fields and nodes in the knowledge base.

The method performs a semi-automatic TBox transformation, taking into account the syntactic and semantic levels. It makes explicit the semantics of the relations of the thesaurus terms. It considers only the *subClassOf* relations and provides methodological guidelines for performing the transformation of the thesaurus. However, it provides neither information about the employed techniques nor a tool to support the method.

Additionally, it produces a lightweight ontology. The ontology components generated are classes, attributes, and relations and they are implemented in RDF(S).

- **Hyvönen et al.** [HVTS08] present a method for transforming thesauri into ontologies. This method has been applied to the YSA thesaurus[16]. It provides neither information about the format of the thesaurus nor the internal data model of the thesaurus.

  This method assumes that the non-ontological resource, YSA, is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it manage the resource provenance information; therefore, the resultant ontology does not keep the reference to the non-ontological resource.

  On the other hand, it performs a semi-automatic TBox transformation, regarding the syntactic and semantic levels and makes explicit the *subClassOf* and *partOf* relations by using DOLCE [17] ontology. It also provides methodological guidelines for performing the transformation of the resource and relies on an *ad-hoc* tool; however, it does not provide information about the techniques employed.

  The resultant heavyweight ontology, based on the YSA thesaurus, is the General Finnish Ontology YSO[18]. The ontology components generated are classes, attributes, and relations, all expressed in RDF(S).

[16] http://vesa.lib.helsinki.fi/

[17] http://www.loa-cnr.it/DOLCE.html

[18] http://www.yso.fi/onto/yso

- **Soergel et al.** [SLL$^+$04], and **Lauser et al.** [LS06] present a method for re-engineering the traditional thesaurus, AGROVOC[19], which is stored in a database, into an ontology. This method considers the internal data model of the thesaurus.

  Moreover, it assumes that the AGROVOC thesaurus is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it provide the resource provenance information; therefore, the resultant ontology does not keep the reference to the non-ontological resource.

  On the other hand, the method of Soergel et al. explores the applicability of the *rules-as-you-go approach* to improve the re-engineering process. The steps of the transformation process are (1) to define the ontology structure; (2) to fill in values from one or more legacy KOS to the extent possible; and (3) to edit manually using an ontology editor and to make the existing information more precise by adding new information. In order to automate the process, Soerger et al. planned to build an inventory of patterns, namely, content ontology design patterns specific for the agricultural domain; however, the inventory has not yet been built.

  Lauser et al. present the basic OWL model, which was extracted manually from the analysis of AGROVOC schema, using the results of the Soergel et al.'s work; and they point out as a future line of work the conversion of the AGROVOC content into ontology instances. They plan to develop a Web based tool for maintaining the resultant ontology.

  Their method performs a manual TBox transformation, considering the syntactic and semantic levels. This method makes explicit the *subClassOf* and *ad-hoc* relations by means of a domain expert. It provides methodological guidelines for performing the transformation of the resource.

  Besides, this method produces a heavyweight ontology. The components of the ontology generated are classes, attributes, and relations, all expressed in OWL DL.

2.1.2.2 Methods centred on the ton-ontological resource implementation

In this section we present the most relevant methods we have found in the literature: research works for building ontologies from databases, XML, flat files and spreadsheets.

Methods for building ontologies from databases The two main methods for building ontologies from databases are presented in [SSV02],[BCGP04, Bar07].

[19] `http://www.fao.org/aims/ag_intro.htm`

- **Stojanovic et al.** [SSV02] present an integrated and semi-automatic approach for generating shared-understandable metadata of data-intensive Web applications. Their method deals with resources stored in relational databases, and takes into consideration the internal data model of the resources.

  It assumes that the resource is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it provide the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

  This method consists of the following steps: (1) the capture of information from relational schema through reverse engineering (it should be added that user interaction is necessary in this step); (2) the analysis of the information obtained and the maping of database entities into ontological entities with a set of mapping rules; (3) the evaluation, validation and refining of the mapping; and (4) the creation of a knowledge base, i.e. data migration.

  The method performs a semi-automatic ABox transformation, taking into consideration only the syntactic level. However, it does not make explicit the semantics of the relations of the NOR elements, though it provides methodological guidelines and employs mapping rules as a technique for the transformation. For the automation of the mapping process, it relies on KAON-REVERSE[20], a tool for connecting semi-automatically relational database to ontologies.

  In addition, it produces a lightweight ontology and generates ontology instances. The resultant ontology is expressed in F-Logic[21], and the ontology instances are expressed in RDF.

- **Barrasa et al.** [BCGP04, Bar07] present an integrated framework for the formal specification, evaluation and exploitation of the semantic cor-respondences between ontologies and relational data sources. These authors introduce a method that deals with resources stored in relational databases and that takes into consideration the internal data model of the resources.

  The framework consists of the following two main components: (1) R_2O, which is a declarative language for the description of arbitrarily complex mapping expressions between ontology elements (concepts, attributes and relations) and relational elements (relations and attributes); and (2) ODEMapster processor, which generates Semantic Web instances from relational instances based on the mapping description expressed in an R_2O document.

  The method assumes that the database is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it consider the resource provenance infor-

[20] http://kaon.semanticweb.org/alphaworld/reverse/

[21] http://flora.sourceforge.net/aboutFlogic.php

mation; therefore, the resultant ontology does not keep the reference to the non-ontological resource.

This method consists in (1) discovering semi-automatically mappings between the database and ontology elements, user interaction is necessary in some special cases; (2) expressing those mappings in a formal language, R_2O; (3) evaluating and verifying those mappings manually; and (4) exploiting those mappings for retrieving the data using ODEMapster.

The method performs an automatic population of an ontology, taking into consideration the syntactic level. However, it does not make explicit the semantics of the relations of the NOR elements, although it provides methodological guidelines for the transformation.

Finally, it generates ontology instances expressed in RDF.

Methods for building ontologies from XML files The three main methods for building ontologies from XML files are presented in [GC05, AM05, CXH04].

- **García et al.** [GC05] introduce a method to create an ontology from the XML schema and to populate it with instances created from the XML data. The method does not take into account the internal data model of the resource.

  This method assumes that the resource, stored in an XML file, is already selected for its transformation. Therefore it provides neither methodological guidelines nor tools for the reuse process. Nor does it consider the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

  The method consists of the following steps: (1) XSD2OWL Mapping, in which the semantics implicit in the schema is captured with the XSD2OWL tool; and (2) XML2RDF Mapping, in which a translation of the XML metadata instances to RDF instances is performed with the XML2RDF tool.

  This method performs a semi-automatic ABox transformation, taking into account only the syntactic level. However, it does not make explicit the semantics of the relations of the NOR elements, although it provides methodological guidelines and employs mapping rules as a technique for the transformation.

  Finally, it produces a single lightweight ontology. The ontology components generated are classes, attributes, relations, and instances, all expressed in RDF/OWL Full.

- **An et al.** [AM05] present a method for translating an XML web document into an instance of an OWL DL ontology. The method does not consider the internal data model of the resource.

The method assumes that the resource, stored in an XML file, is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Moreover, the method does not keep the resource provenance information, so the resultant ontology does not keep the reference to the non-ontological resource.

This method takes advantage of the semi-automatic mapping discovery tool [ABM05] for the relationship between XML schema and the ontology. It performs a semi-automatic population, considering only the syntactic level. It does not make explicit the semantics of the relations of the NOR elements and does not provide methodological guidelines.

Finally, it populates an ontology. The ontology components generated are and instances, all expressed in RDF.

- **Cruz et al**. [CXH04] present a method that transforms XML schema into an ontology and preserves the XML document structure, i.e., by modelling the knowledge implicit in XML schema with RDF(S). This method does not take into account the internal data model of the resource.

  The method assumes that the resource, stored in an XML file, is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it consider the resource provenance information; so the resultant ontology does not keep the reference to the non-ontological resource.

  Basically, the method consists of the following phases: (1) element-level transformation, which defines the basic classes and properties of the ontology; (2) structure-level transformation, which encodes the hierarchical structures of the XML schema into the ontology; and (3) query driven data migration, which transforms the query expressed in RDQL[22] into XQuery[23] query and creates the RDF instances that satisfies the query.

  This method performs a semi-automatic ABox transformation, taking into account only the syntactic level, but it does not make explicit the semantics of the relations of the NOR elements. Besides, it provides methodological guidelines and employs mapping rules as a technique for the transformation.

  Finally, it produces a single lightweight ontology. The ontology components generated are classes, attributes, relations, and instances, all expressed in RDF(S).

Methods for building ontologies from flat files The main method to transforming flat file is presented in [FB06].

[22] `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`

[23] `http://www.w3.org/TR/xquery/`

- **Foxvog et al.** in [FB06] present a method that transforms Electronic Data Interchange (EDI)[24] messages into ontologies. There are two major EDI standards: the EDIFACT [Ber94], which is defined as an open standard by the United Nations, and the ASC X12[25] standard, which is primarily used in the United States. This method is centred on the ASC X12 standard, whose messages are stored in flat files. However, it does not take into account the internal data model of the resource.

  The method assumes that the resource, EDI standard, is already selected for its transformation. Therefore, it provides neither methodological guidelines nor tools for the reuse process. Nor does it consider the resource provenance information; for this reason, the resultant ontology does not keep the reference to the non-ontological resource.

  This method, which transforms ASC X12 messages into ontologies, consists in creating (1) a vocabulary for specifying the formats of the messages; and (2) a set of instances.

  It also performs a semi-automatic ABox transformation, considering only the syntactic level, and provides methodological guidelines. However, it does not make explicit the semantics of the relations of the NOR elements.

  Finally, it produces several lightweight ontologies. The ontology components generated are classes, attributes, relations, and instances, all expressed in OWL Full, CycL, and WSML.

2.1.2.3 Comparison of the methods

Tables 2.1, 2.2, 2.3 and 2.4 summarize the methods presented according to the features related to the non-ontological resource, the reuse process, the transformation process, and the resultant ontology.

2.1.3 Tools for Re-engineering Non-ontological Resources

In this section, we describe the most significant non-ontological resource re-engineering tools according to the features identified in section 2.1.1. This section is organized into tools centred on the non-ontological resource type (section 2.1.3.1) and tools centred on the non-ontological resource implementation (section 2.1.3.2). Some of the tools give support to the methods presented in section 2.1.2. Also, it is worth mentioning that the tools provide (1) neither support for selecting the non-ontological resources for their subsequent transformation, (2) nor provenance information support.

[24] http://www.ifla.org/VI/5/reports/rep4/42.htm#chap2

[25] http://www.x12.org/

Table 2.1: Comparison of the methods according to the NOR features

Research work	Type of resource	Data model is known	Resource implemented in
Hepp et al. [HdB07]	Classification scheme, thesauri	No	Database
Hakkarainen et al. [HHST06]	Classification scheme	Yes	Flat file
Abbasi et al. [ASC07]	Folksonomy	No	Not mentioned
Maala et al. [MDA07]	Folksonomy	No	Not mentioned
Van Assem et al. [vAGS06]	Lexica	Yes	Prolog
Gangemi et al. [GNV03, GGMO03]	Lexica	Yes	Database
Hahn et al. [Hah03, HS03]	Thesauri	Yes	ASCII files
Van Assem et al. [vAMSW04]	Thesauri	No	proprietary text format, database, XML
Van Assem et al. [vAMMS06]	Thesauri	No	Not mentioned
Wielinga et al. [WSWS01]	Thesauri	Yes	XML
Hyvönen et al. [HVTS08]	Thesauri	No	Not mentioned
Soergel et al. [SLL$^+$04, LS06]	Thesauri	Yes	Database
Stojanovic et al. [SSV02]	Not specified	Yes	Database
Barrasa et al. [BCGP04, Bar07]	Not specified	Yes	Database
García et al. [GC05]	Not specified	No	XML
An et al. [AM05]	Not specified	No	XML
Cruz et al. [CXH04]	Not specified	No	XML
Foxvog et al. [FB06]	Not specified	No	Flat file

2.1.3.1 Tools centred on the non-ontological resource type

This section presents some of the tools we have found in the literature related to the building of ontologies by re-engineering non-ontological resources. Such tools transform classification schemes, folksonomies, lexica and thesauri into ontologies.

A tool for transforming classification schemes into ontologies

SKOS2GenTax[26] is an online tool that converts hierarchical classifications, avail-

[26] http://www.heppnetz.de/projects/skos2gentax/

Table 2.2: Comparison of the methods according to the reuse process features

Research work	Methodological Guidelines	Tool Support	Provenance
Hepp et al. [HdB07]	No	No	No
Hakkarainen et al. [HHST06]	No	No	No
Abbasi et al. [ASC07]	No	No	No
Maala et al. [MDA07]	No	No	No
Van Assem et al. [vAGS06]	No	No	Yes
Gangemi et al. [GNV03, GGMO03]	No	No	No
Hahn et al. [Hah03, HS03]	No	No	No
Van Assem et al. [vAMSW04]	No	No	No
Van Assem et al. [vAMMS06]	No	No	No
Wielinga et al. [WSWS01]	No	No	No
Hyvönen et al. [HVTS08]	No	No	No
Soergel et al. [SLL$^+$04, LS06]	No	No	No
Stojanovic et al. [SSV02]	No	No	No
Barrasa et al. [BCGP04, Bar07]	No	No	No
García et al. [GC05]	No	No	No
An et al. [AM05]	No	No	No
Cruz et al. [CXH04]	No	No	No
Foxvog et al. [FB06]	No	No	No

able in the W3C SKOS[27] format, into RDF(S) or OWL DL ontologies. This tool uses the GenTax algorithm described in [HdB07]. The input resource can be specified by its URL, or it can be uploaded directly to the Web site. This resource should be available in SKOS RDF format.

A tool for transforming folksonomies into ontologies

Abbasi et al. [ASC07] present **T-ORG**, a system that organizes folksonomies by classifying the tags attached to them into predefined categories. The input resource is a flat folksonomy tagspace. T-ORG gives technological support to the method described in [ASC07].

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

Table 2.3: Comparison of the methods according to the transformation process features

Research work	Transformation Approach	Transformation Aspects	Semantics of NOR relations	Additional Resources / Domain Expert	Automatic / Semi-automatic / Manual	Methodological Guidelines	Technique	Tool support
Hepp et al. [HdB07]	TBox	syntactic, semantic	*subClassOf*, *ad-hoc* relation	No	Semi-automatic	Yes	Not mentioned	SKOS2GenTax
Hakkarainen et al. [HHST06]	ABox	syntactic, semantic	*subClassOf*, *ad-hoc* relation	No	Semi-automatic	Yes	Transformation rules	Not mentioned
Abbasi et al. [ASC07]	Population	syntactic	No	Swoogle Google	Automatic	Yes	Lexico Syntactic Patterns	T-ORG
Maala et al. [MDA07]	Population	syntactic	No	WordNet, Geographical locations, Ontology of things	Automatic	Yes	Not mentioned	Not mentioned
van Assem et al. [vAGS06]	ABox	syntactic	No	No	Semi-automatic	Yes	Not mentioned	Swi-Prolog
Gangemi et al. [GNV03, GGMO03]	TBox	syntactic, semantic	*ad-hoc* relations	DOLCE	Semi-automatic	Yes	NLP Techniques	Not mentioned
Hahn et al. [Hah03, HS03]	TBox	syntactic, semantic	*subClassOf*, *partOf*, *ad-hoc* relation	No	Semi-automatic	Yes	Ontology Design Patterns	*Ad-hoc* tool
van Assem et al. [vAMSW04]	TBox	syntactic, semantic	*subClassOf*, *ad-hoc* relation	No	Semi-automatic	Yes	Not mentioned	*Ad-hoc* tool
van Assem et al. [vAMMS06]	Population	syntactic	Not mentioned	No	Automatic	Yes	Not mentioned	Swi-Prolog
Wielinga et al. [WSWS01]	TBox	syntactic, semantic	*subClassOf*	Not mentioned	Semi-automatic	Yes	Not mentioned	*Ad-hoc* tool
Hyvönen et al. [HVTS08]	TBox	syntactic, semantic	*subClassOf*, *partOf*	DOLCE	Semi-automatic	Yes	Not mentioned	*Ad-hoc* tool
Soergel et al. [SLL^{+}04, LS06]	TBox	syntactic, semantic	*subClassOf*, *ad-hoc* relation	No	Manual	Yes	Not mentioned	Not mentioned
Stojanovic et al. [SSV02]	Population	syntactic	*ad-hoc* relation	No	Semi-automatic	Yes	Mapping rules	KAON-REVERSE
Barrasa et al. [BCGP04, Bar07]	Population	syntactic	*subClassOf*, *ad-hoc* relation	No	Semi-automatic	Yes	Mapping rules	ODEMapster
García et al. [GC05]	ABox	syntactic	*ad-hoc* relation	No	Semi-automatic	Yes	Mapping rules	XSD2OWL XML2RDF
An et al. [AM05]	ABox	syntactic	*ad-hoc* relation	No	Semi-automatic	No	Not mentioned	Discovery tool
Cruz et al. [CXH04]	ABox	syntactic	Not mentioned	No	Semi-automatic	Yes	Mapping rules	*Ad-hoc* tool
Foxvog et al. [FB06]	ABox	syntactic	Not mentioned	No	Semi-automatic	Yes	Not mentioned	*Ad-hoc* tool

Table 2.4: Comparison of the methods according to the ontology features

Research Work	**Lightweight/ Heavyweight**	**Components**	**Implementation language**	**Single/ Several**
Hepp et al. [HdB07]	Lightweight	classes, relations	RDF(S) / OWL DLP	Single
Hakkarainen et al. [HHST06]	Lightweight	classes, attributes, relations	OWL DL	Single
Abbasi et al. [ASC07]	Lightweight	instances	Not mentioned	Several
Maala et al. [MDA07]	Lightweight	instances	RDF	Single
Van Assem et al. [vAGS06]	Lightweight	classes, attributes, relations, instances	RDF(S) / OWL Full	Single
Gangemi et al. [GNV03, GGMO03]	Lightweight	classes, attributes, relations	DAML+OIL	Single
Hahn et al. [Hah03, HS03]	Heavyweight	classes, relations	LOOM / ALC	Single
Van Assem et al. [vAMSW04]	Lightweight	classes, attributes, relations	RDF(S) / OWL Full	Single
Van Assem et al. [vAMMS06]	Lightweight	instances	SKOS RDF	Single
Wielinga et al. [WSWS01]	Lightweight	classes, attributes, relations	RDF(S)	Single
Hyvönen et al. [HVTS08]	Heavyweight	classes, attributes, relations	RDF(S)	Single
Soergel et al. [SLL$^+$04, LS06]	Heavyweight	classes, attributes, relations	OWL DL	Single
Stojanovic et al. [SSV02]	Lightweight	instances	F-Logic / RDF	Single
Barrasa et al. [BCGP04, Bar07]	Lightweight	instances	RDF	Single
García et al. [GC05]	Lightweight	classes, attributes, relations, instances	OWL Full/ RDF	Single
An et al. [AM05]	Lightweight	instances	RDF	Single
Cruz et al. [CXH04]	Lightweight	classes, attributes, relations, instances	RDF(S)	Single
Foxvog et al. [FB06]	Lightweight	classes, attributes, relations, instances	CycL / OWL Full / WSML	Several

2.1.3.2 Tools centred on the non-ontological resource implementation

In this section we present some of the tools we found in the literature related to the re-engineering of non-ontological resources and centred on their implementation. We first introduce some research works to transform databases, XML files, spreadsheet files and flat files into ontologies.

Tools for transforming databases into ontologies The four main tools for transforming databases are KAON-REVERSE, ODEMapster, D2R Server and TopBraid Composer. Next we describe each one of them.

- **KAON-REVERSE**[28] is a tool that supports the reverse engineering method presented in [SSV02] for transforming databases into ontologies. This tool performs an ABox transformation of the databases, generating ontologies in F-Logic and instances in RDF.

- **ODEMapster**[29] is the processor in charge of carrying out the exploitation of the mappings defined with R_2O [Bar07]. This tool is intended to create instances of an available ontology on demand or in a batch processing. The ontologies have to be expressed in OWL or RDF(S), and the instances generated are expressed in RDF.

- **D2R Server**[30] is a tool for publishing the content of relational databases on the Semantic Web. This tool does not consider the data model of the resource stored in the database.

  It is intended to create instances of an ontology on demand or in a batch processing, that is, to populate ontologies. D2R Server performs a semi-automatic conversion and does not consider the internal data model of the resource. Nor does it provide the resource provenance information, so the resultant ontology does not keep the reference to the database. D2R Server consists of: (1) a D2RQ mapping language, a declarative language for describing the relation between an ontology and a relational model; and (2) a D2RQ engine, that is, a plug-in for the Jena and Sesame Semantic Web toolkits. This engine uses the mappings to rewrite Jena and Sesame API calls into SQL queries against the database and passes query results up to the higher layers of the frameworks.

  This tool populates a single ontology. The resultant ontology instances are expressed in RDF.

- **TopBraid Composer**[31] is an enterprise-class modelling environment for developing Semantic Web Ontologies. TopBraid Composer can convert databa-

[28] http://kaon.semanticweb.org/alphaworld/reverse/

[29] www.oeg-upm.net/index.php/en/downloads/9-r2o-odempaster

[30] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server

[31] http://www.topbraidcomposer.com/

ses into ontologies, but does not consider the internal data model of the resource. This tool has a relational database importer, D2RQ[32] in its platform. TopBraid performs an ABox transformation, though it does not provide the resource provenance information; therefore, the resultant ontology does not keep the reference to the database.

TopBraid Composer performs the following tasks for converting databases into ontologies: (1) static import of schema, where tables become classes, columns become properties and link tables become object properties; and (2) dynamic import of actual data, where rows become instances on the fly, i.e., data can stay where it is.

The tool produces a single ontology whose components are classes, attributes, relations, and instances. The resultant ontology is expressed in RDF/OWL (Full, DL or Lite).

Tools for transforming XML files into ontologies The main tools are XSD2-OWL, XML2RDF, and TopBraid Composer.

- **XSD2OWL** and **XML2RDF**[33] are tools that support the method for transforming XML files into ontologies [GC05]. The input files are (1) an XML schema definition (XSD) file, which describes the XML schema; and (2) an XML file, which contains the XML instances. This tool does not consider the data model of the resource stored in the XML.

  It produces a single lightweight ontology whose components are classes, attributes, relations, and instances, all expressed in RDF/OWL Full.

- **TopBraid Composer** also can convert XML files into ontologies. This tool does not consider the data model of the resource stored in the XML.

  It performs a semi-automatic ABox transformation and a Population, taking into consideration the syntactic level. Besides, it relies on mapping rules as a technique for performing the transformation.

  The ontology components generated are classes, attributes, relations, and instances. The resultant ontology is expressed in RDF/OWL (Full, DL or Lite) and the tool generates a single lightweight ontology.

Tools for transforming flat files into ontologies The four main tools for transforming flat files are TopBraid Composer, ConvertToRdf, flat2rdf and Java BibTeX-To-RDF converter.

- **TopBraid Composer** can also convert flat files into ontologies. This tool performs a semi-automatic ABox transformation, taking into account the syntactic level.

[32] `http://www4.wiwiss.fu-berlin.de/bizer/d2rq/`

[33] `http://rhizomik.net/redefer/`

The ontology components generated are classes, attributes, relations, and instances. The resultant ontology is expressed in RDF/OWL (Full, DL or Lite), and the tool generates a single lightweight ontology.

- **ConvertToRdf**[34] is a tool for automatically converting delimited text data into RDF via a simple mapping mechanism. The input resources are delimited text files. This tool does not consider the data model of the resource stored in the XML.

  It performs a semi-automatic population of an available ontology from the data stored in the flat file. And it produces a single lightweight ontology. The resultant ontology instances are expressed in RDF.

- **flat2rdf**[35] is a simple *Perl script* that converts classic Unix text database files into RDF.

  The input resources are classic Unix text files (e.g., `/etc/passwd`). This tool populates semi-automatically an available ontology.

  The tool generates a single lightweight ontology. The resultant ontology instances are expressed in RDF.

- **Java BibTeX-To-RDF Converter**[36] allows converting BibTeX files into an RDF format according the SWRC ontology[37].

  The input resources are plain BiBTex files (i.e. text files). This tool populates automatically an ontology from the information stored in the BiBTex files

  The resultant ontology instances are expressed in RDF.

Tools for transforming spreadsheet files into ontologies The four main tools for transforming spreadsheet files are TopBraid Composer, Excel2rdf, RDF123, and XLWrap.

- **TopBraid Composer** can also convert spreadsheets into ontologies. The tool receives Excel spreadsheets as input, but it does not consider the internal resource data model. Besides, it performs semi-automatically an ABox transformation.

  The ontology components generated are classes, attributes, relations, and instances. The resultant ontology is expressed in RDF/OWL (Full, DL or Lite) and the tool deals with a single lightweight ontology.

- **Excel2rdf**[38] is a Microsoft Windows program that converts Excel files into valid RDF. It populates semi-automatically an ontology, but it does not consider the data model of the resource for the transformation.

[34] `http://www.mindswap.org/~mhgrove/convert/`

[35] `http://simile.mit.edu/repository/RDFizers/flat2rdf/`

[36] `http://www.aifb.uni-karlsruhe.de/WBS/pha/bib/index.html`

[37] `http://ontoware.org/projects/swrc/`

[38] `http://www.mindswap.org/~Erreck/excel2rdf.shtml`

The resultant ontology instances are expressed in RDF.

- **RDF123** [HFP$^+$06] is a highly flexible open source tool for transforming semi-automatically spreadsheet data into RDF. It works on CSV files and also Google spreadsheets.

  This tool populates semi-automatically an available ontology. Every row of a spreadsheet will generate a row graph, and the RDF graph produced for the whole spreadsheet is the merge of all row graphs, eliminating duplicated resources and triples.

  RDF123 consists of the following two components: (1) an RDF123 application, that is, a component whose main purpose is to give users an interactive and easy-to-use graphical interface for creating the map graph and outputting the map graph in RDF syntax; and (2) a RDF123 Web Service, which aims to provide a public service that translates online spreadsheets into RDF.

  This tool populates more than one ontology and the resultant ontology instances are expressed in RDF.

- **XLWrap** [LW09] is a spreadsheet-to-RDF wrapper that is capable of transforming spreadsheets into arbitrary RDF graphs based on a mapping specification. The tool supports Microsoft Excel and OpenDocument spreadsheets such as comma (and tab) separated value (CSV) files and it can load local files or download remote files via HTTP.

  This tool populates semi-automatically ontologies. Every row of a spreadsheet will generate a row graph, and the RDF graph produced for the whole spreadsheet is the merge of all row graphs, eliminating duplicated resources and triples.

  This tool populates more than one ontology; thus, the resultant ontology instances are expressed in RDF.

2.1.3.3 Comparison of the tools

Tables 2.5, 2.6 and 2.7 present the tools according to the features related to the non-ontological resource, the transformation process and the resultant ontology.

2.1.4 Results and Conclusions

After having analysed the state of the art of the methods and tools for re-engineering non-ontological resources, we present the results of applying the evaluation framework described in section 2.1.1. The results are provided according to the features of the groups identified, namely, non-ontological resource, reuse process, transformation process, and resultant ontology.

Table 2.5: Comparison of the tools according to the NOR features

Tool	**Type of resource**	**Data model model is known**	**Resource implemented in**
SKOS2GenTax	Classification schemes, thesauri	No	SKOS RDF
T-ORG	Folksonomy	No	Not mentioned
KAON-REVERSE	Not specified	Yes	Database
ODEMapster	Not specified	Yes	Database
D2R Server	Not specified	No	Database
TopBraid Composer	Not specified	No	Database, XML, Flat file, Spreadsheet
XSD2OWL and XML2RDF	Not specified	No	XML
ConvertToRdf	Not specified	No	Delimited text data file
flat2rdf	Not specified	No	Flat file
Java BibTeX-To-RDF Converter	Not specified	No	Flat file
Excel2rdf	Not specified	No	Spreadsheet
RDF123	Not specified	No	Spreadsheet
XLWrap	Not specified	No	Spreadsheet

2.1.4.1 Results according to non-ontological resource

Table 2.1 and table 2.5 summarize the methods and tools presented according to the characteristics of the non-ontological resource: type of resource, knowledge about the data model, and resource implementation.

Methods

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

- 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 the other half does.

Table 2.6: Comparison of the tools according to the transformation process features

Tool	**Transformation Approach**	**Transformation Aspects**	**Semantics of NOR relations**	**Additional Resources / Domain Expert**	**Automatic / Semi-automatic / Manual**	**Technique**
SKOS2GenTax	TBox	syntactic, semantic	*subClassOf*, *ad-hoc* relation	No	Semi-automatic	Not mentioned
T-ORG	Population	syntactic	No	Swoogle	Automatic	Lexico Syntactic Patterns
KAON-REVERSE	ABox	syntactic	No	No	Semi-automatic	Mapping rules
ODEMapster	Population	syntactic	No	No	Semi-automatic	Mapping rules
D2R Server	Population	syntactic	No	No	Semi-automatic	Mapping rules
TopBraid Composer	ABox	syntactic	No	No	Semi-automatic	Mapping rules
XSD2OWL and XML2RDF	ABox	syntactic	No	No	Semi-automatic	Mapping rules
ConvertToRdf	Population	syntactic	No	No	Semi-automatic	Not mentioned
flat2rdf	Population	syntactic	No	No	Semi-automatic	Not mentioned
Java BibTeX-To-RDF Converter	Population	syntactic	No	No	Automatic	Not mentioned
Excel2rdf	Population	syntactic	No	No	Semi-automatic	Not mentioned
RDF123	Population	syntactic	No	No	Semi-automatic	Not mentioned
XLWrap	Population	syntactic	No	No	Semi-automatic	Mapping rules

Table 2.7: Comparison of the tools according to the ontology features

Tool	Lightweight/ Heavyweight	Components	Implementation language	Single/ Several
SKOS2GenTax	Lightweight	classes, attributes, relations	OWL DLP/ RDF(S)	Single
T-ORG	Lightweight	instances	Not mentioned	Several
KAON-REVERSE	Lightweight	classes, attributes, relations, instances	F-Logic / RDF	Single
ODEMapster	Lightweight	instances	RDF	Single
D2R Server	Lightweight	instances	RDF	Single
TopBraid Composer	Lightweight	classes, attributes, relations, instances	RDF/OWL (Full, DL or Lite)	Single
XSD2OWL and XML2RDF	Lightweight	classes, attributes, relations, instances	OWL Full/ RDF	Single
ConvertToRdf	Lightweight	instances	RDF	Single
flat2rdf	Lightweight	instances	RDF	Single
Java BibTeX-To-RDF Converter	Lightweight	instances	RDF	Single
Excel2rdf	Lightweight	instances	RDF	Single
RDF123	Lightweight	instances	RDF	Several
XLWrap	Lightweight	instances	RDF	Several

- With regard 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 the resource implementation. In addition, one method is focused on resources implemented in Prolog, whereas another method includes resources implemented in proprietary format, relational database, and XML.

Tools

- In relation to the type of non-ontological resource, we can observe that most of the tools do not consider the type of the resource, since they are focused on the resource implementation. In addition, one tool considers classification schemes and thesauri, whereas another considers folksonomies.

- As for the data model, most of the tools do not consider the data model for the transformation.

- With regard to the implementation, almost all the tools are focused on the resource implementation, many of them on databases, and some on spreadsheets, XML and flat files. Only one tool provides an integrated environment that considers resources implemented in databases, XML, flat files, and spreadsheets.

To sum up we can affirm that most of the methods and tools 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 artefact in the re-engineering process [GGPSFVT08]. There is not integrated framework, method or corresponding tool that considers the resource types, data models and implementations identified in a unified way. Thus, we can conclude that there is a clear need for some sort of re-engineering methods and tools 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.
- Deal with non-ontological resources implemented in databases, XML files, flat files, or spreadsheets.

2.1.4.2 Results according to the reuse process

Table 2.2 summarizes the methods described above regarding the features of the reuse process: methodological guidelines for the selection of the resource, tool support and provenance information. We can conclude that the whole set of methods assumes that the non-ontological resources are already selected for their transformation; therefore, they do not provide methodological guidelines for the selection of the resource. Consequently, there is not tool support for this process. It should be noted that only one method keeps the reference to the non-ontological resource.

In conclusion, we can say that there is a clear need for some sort of methods and tools that

- Provide guidelines for the selection of the most appropriate resources for building an ontology.
- Consider the provenance information of the resource.

2.1.4.3 Results according to transformation process

Table 2.3 and table 2.6 summarize the methods and tools presented above regarding the features of the transformation process, namely, the transformation approach; the transformation performed on the syntactic and semantic level; the explicitation of the hidden semantics in the relations of the resource terms; the use of additional resources or a domain expert for making explicit the hidden semantics of the relations; the degree of automation; the provision of methodological guidelines; and the list of the techniques employed.

Methods

- With regard to the transformation approach, 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 the transformation at the syntactic and semantic levels, we can observe that this feature is closely related to the transformation approach performed by the methods. As mentioned in section 2.1.1.3, the ABox transformation disregards the informal semantics of the transformed resources, so the transformation is performed only at the syntactic level; and this also happens to the Population. On the contrary, the TBox transformation tries to enforce a formal semantics on the resources, so the transformation is performed at the syntactic and semantic levels.

- As for the explicitation of the hidden semantics of the relations of the resource components, we can state that the methods performing 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 respect to how the methods make explicit the hidden semantics in the relations of the resource components, 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 though not for making explicit the hidden semantics, but for finding out a proper ontology and populating it.

- Regarding the degree of automation, almost all the methods perform a semi-automatic transformation of the resource, three are performed automatically, and one is done manually.

- As for 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 task/activity, nor when that task/activity 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 establish techniques as transformation rules, lexico-syntactic patterns, mapping rules and natural language techniques.

- As for the tool support, most of the methods rely on *ad-hoc* tools for the transformation, but only a few integrate a public available tool, such as, KAON-REVERSE, ODEMapster, XSD2OWL, or XML2RDF.

Tools

- Regarding the transformation approach, most of the tools perform a population, some perform an ABox transformation, and one performs a TBox transformation. However, no tool includes the possibility to perform the three transformation approaches.
- With respect to the transformation at the syntactic and semantic levels, as we observed, before this feature is closely related to the transformation approach performed by the tools. The ABox transformation is performed only at the syntactic level, and this also happens to the Population. On the contrary, the TBox transformation is performed at the syntactic and semantic level. However, almost of all the tools perform a population, three perform an ABox and a TBox transformation.
- With regard to the explicitation of the hidden semantics in the relations of the resource terms, we can state that the tool performing the TBox transformation is the only one that makes explicit the semantics hidden in the relations of the NOR terms.
- As for how the tools make explicit the hidden semantics in the relations of the resource terms, the only tool that makes this explicitation it does it by setting both *ad-hoc* and taxonomic relations among the NOR terms, though it does not state which relation is the correct. Moreover, there is one tool that relies on an external resource, though it does not do it for expliciting the hidden semantics, but for finding out a proper ontology for populating it.
- With respect to the degree of automation, almost all the tools perform a semi-automatic transformation of the resource, and only two tools perform an automatic transformation.
- Regarding the techniques employed, the majority of the tools do not mention them at all. Only a few methods specify techniques as mapping rules and lexico syntactic patterns.

In summary, after having analysed the features related to the transformation process, we can conclude that (1) methods are mostly focused on the TBox transformation approach, whereas tools are focused on the population; (2) only a few methods and tools make explicit the hidden semantics in the relations of the NOR components, and most of them rely on the domain expert for doing it; (3) almost all the methods provide a methodological guidelines for the transformation, but they are not finely detailed; (4) only a few methods and tools specify the techniques employed for the transformation, and (5) there is not any integrated framework, method or corresponding tool 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 and tools 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 an 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 task/activity and when this task/activity has to be carried out.
- Integrate in a single framework the method and its corresponding tool supporting for the transformation.
- Employ techniques that improve the efficiency of the re-engineering process.

2.1.4.4 Results according to the ontology

Tables 2.4 and 2.7 summarize the methods and tools presented regarding the features of the resultant ontology, namely, whether the ontology is lightweight or heavyweight; the ontology components; the ontology implementation language; and whether one or more ontologies are generated.

Methods

- With respect to whether the ontology is lightweight or heavyweight, most of the methods generate lightweight ontologies; only three rely on domain experts to generate heavyweight ontologies.
- Regarding 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.
- With regard to the ontology implementation language, even though there is a large variety of languages, the ontology languages mostly used are OWL for the ontology and RDF for the instances.
- As for whether the methods generate one or several ontologies, almost all the methods generate a single ontology.

Tools

- Regarding whether the ontology generated is lightweight or heavyweight, we can state that almost all generate lightweight ontologies.
- Concerning the ontology components, we can observe that this feature is closely related to the transformation approach performed by the tools, just like in the case of the methods.
- With respect to the ontology implementation language, and taking into account that almost all the tools generate ontology instances, the language most used is RDF.
- As for whether the tools generate one or several ontologies, we can state that almost all the tools generate a single ontology.

After having analysed the features related to the resultant ontology, we can confirm the lack both of re-engineering methods and tools supporting several ontologies, and of the generation of ontologies with classes, attributes, relations and instances.

2.2 Patterns for Re-engineering

In this section we analyse the role patterns play in software and ontology engineering, emphasizing specifically the re-engineering patterns.

2.2.1 Software Re-engineering

This section is based on the landmark work of Byrne [Byr92], one of the most prominent specialists on software re-engineering. In a nutshell, software re-engineering takes a legacy software that is expensive to maintain or whose system architecture or implementation is obsolete and remakes it with current software and/or hardware technology. The difficulty of this process lies in understanding the existing system. Very often requirements, design and code documentation are no longer available, or are very out of date, so it is unclear what functions have to be moved.

Several definitions have been given for software re-engineering, but the one most widely accepted comes from Chikofsky [CCI90]. He defines *software re-engineering as the examination and alteration of a software system to reconstitute it in a new form and the subsequent implementation.*

2.2.1.1 Levels of abstraction

Understanding how software is developed is useful for understanding how software can be re-engineered. The concept of *levels of abstraction* that underlies the development process also underlies the re-engineering process. This concept is

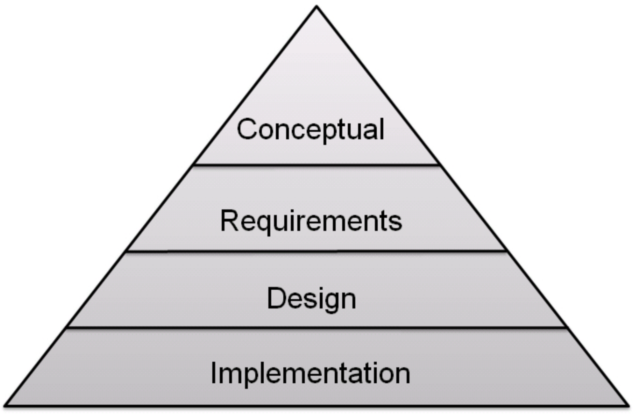


Figure 2.3: Software levels of abstraction [Byr92]

used to model software development as a sequence of phases, in which each phase corresponds to a particular **level of abstraction** [Byr92], as shown in Figure 2.3.

Next, we describe briefly the software levels of abstraction proposed by [Byr92].

- The **conceptual level** describes, in general terms, the functional characteristics of a system, i.e., the concept of the system (its reason for existence).
- The **requirements level** depicts in detail the functional characteristics of a system, though it does not provide details of the internal system.
- The **design level** describes the system characteristics, such as architectural structure, system components, interfaces between components, algorithmic procedures, and data structures. Here, we face two degrees of abstraction levels: (1) the high-level design, which expresses the architectural structure of a system, and (2) the detailed-design, which expresses the internal structure of system components.
- The **implementation level** focuses on the description of the implementation characteristics, and is represented in a language understood by the computer.

Byrne [Byr92] also has made a set of assumptions for software re-engineering based on the software levels of abstraction. Next, we describe briefly each of them.

Assumption 1. *The re-engineering of a software system produces a new form of the system that is better, in some way, than the original form.* This assumption answers the question of why software is re-engineered. There are many different reasons for software re-engineering, and most of them assume that the available software needs to be improved.

Assumption 2. *Software re-engineering begins with an existing system representation expressed at some level of abstraction.* The concept of levels of abstraction contributes to the understanding of software re-engineering. Thus, we can say that,on the one hand, software development starts with an idea for a system and creates a system representation for each abstraction level. On the other hand, software re-engineering starts with an existing system representation.

Assumption 3. *To alter a system characteristic, we have to work at the level of abstraction at which information about that characteristic is explicitly expressed.* This is related to the issue of how to identify the abstraction level at which the re-engineering work should be carried out. Re-engineering changes the characteristics of a software system. To alter information about a system characteristic we have to work at either abstraction level where the characteristic is introduced, or at any level below that.

Assumption 4. *A system characteristic can be altered by working within a level of abstraction below the level at which information about the characteristic is explicitly expressed. However, the best re-engineering result might not be achieved.* “Best possible result” means that the target system has the desired system characteristics and properties.

Assumption 5. *A system characteristic cannot be altered by working at a level of abstraction above the level where information on the characteristic is introduced.* A characteristic cannot be altered, in the sense of manipulating information about that characteristic, by working at a higher abstraction level containing no information on that characteristic.

2.2.1.2 Software re-engineering principles

According to [Byr92] there are three principles underlying a re-engineering method, which are Refinement, Abstraction and Alteration.

Refinement. This principle states that *the gradual decrease in the abstraction level of a system representation is caused by successively replacing available system information with more detailed information.*

Abstraction. This principle establishes that *if a system representation at a particular abstraction level is missing or not up-to-date, then it is possible to reconstruct that representation.*

Alteration. *Alteration is the making of one or more changes to a system representation without changing the degree of abstraction. Alteration includes the addition, deletion, and modification of information.*

2.2.1.3 General model for software re-engineering

Here we present the general software re-engineering model proposed by Byrne [Byr92]. Assumption 3 and the *Principle of Alteration* both show the level of abstraction at which certain types of change can be made. The *Principle of Refinement* is the basis for forward engineering, which creates a target system implementation. This model suggests that re-engineering begins with the available system implementation and produces a target system implementation.

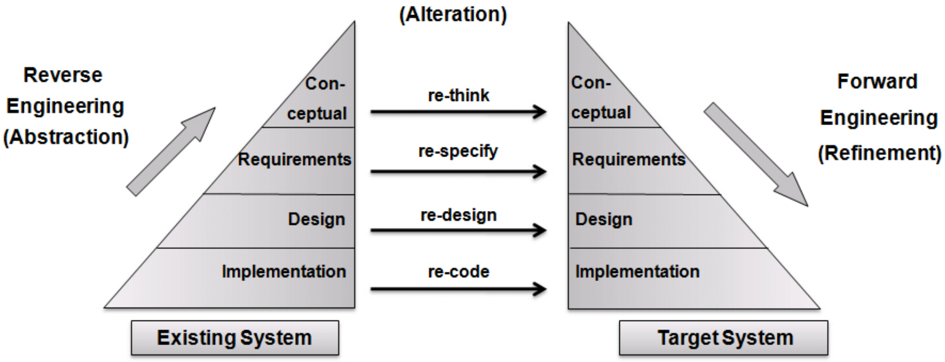


Figure 2.4: General model for software re-engineering [Byr92]

The sequence of reverse engineering, re-designing (or re-coding, re-specifying, re-thinking) and forward re-engineering rests on the three re-engineering principles of Abstraction, Alteration, and Refinement.

Reverse Engineering. Reverse Engineering is the process of analysing a subject system to identify the system's components and their relationships and to create representations of the system in another form or at a higher level of abstraction. In reverse engineering, the requirements and essential design, structure and content of the legacy system must be recaptured. Reverse engineering does not involve changes in the system or creating a new system; it is the process of examining the system without changing its overall functionality.

Alteration or Transformation. To alter a system characteristic, the work is done at the level of abstraction at which information on that characteristic is explicitly expressed. To translate the existing code to a target language no reverse engineering is needed because the alteration (re-coding) is done at the implementation level. As the level of abstraction increases, the alteration tasks change and the amount and tasks of reverse engineering also change. As for re-design changes, they might include restructuring a design architecture, altering a system's data model or a database, etc. To re-specify requirements, reverse engineering techniques must be applied to the implementation and design in order to obtain the functional characteristics. Regarding the re-thinking changes, they can result in drastic changes to a system, i.e., manipulating the concepts embodied in an existing system to create a new system that operates in a different problem domain.

Forward Engineering. The new target system is created by moving downward through the levels of abstraction, that is, a gradual decrease in the abstraction level of the system representation, by successively replacing system information with more detailed information. This downward movement is actually a forward movement through the standard software development, i.e., forward engineering.

2.2.2 Software Patterns

In this section we introduce briefly the role that patterns play in software and ontology engineering, focusing specially on software re-engineering patterns.

Patterns were introduced by Christopher Alexander [Ale79] to encode knowledge and experience when 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 [Ale79]. 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 the design considerations are used to decide whether the pattern is useful and how it could be best implemented [EPJ06].

2.2.2.1 Software design patterns

Software design patterns [Tic97] describe proven solutions to recurring software design problems. A software design pattern consists of one or several software design elements (such as interfaces, classes, objects, methods, functions, processes, threads), relationships among the elements (for example, association, inheritance, delegation, invocation, and creation), and a behavioural description. Examples of design patterns are the Layered System and the Model-View-Controller.

The purpose of design patterns is to capture "design know-how" and to make it reusable. Design patterns can improve the structure of software, speed up implementation, simplify maintenance, and help to avoid architectural drift. Design patterns also improve communication among software developers and can empower less experienced developers to produce high-quality designs [Tic97].

There are several classifications of software design patterns. The classification proposed by Gamma et al. [GHJV95] uses two orthogonal dimensions: (1) purpose, in which they define three categories, namely, creational, structural and behavioural patterns, and (2) scope, in which they distinguish whether a pattern applies primarily to classes or to objects. Buschmann et al. [BMR$^+$96] propose three categories, namely, architectural patterns, design patterns, and idioms. In addition to these, there are several online catalogues of software design patterns, such as the

Design Pattern Library [39], the Portland Pattern Repository[40], the Design Patterns Study Group of New York City[41], and the Architecture & Design Patterns[42].

2.2.2.2 Software re-engineering patterns

A kind of software patterns are the re-engineering software patterns [PS98, LDP]. They 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; they can sometimes suggest which type of tool to use [LDP].

The structure of a re-engineering pattern consists of some essential elements, which are described next.

- *Pattern name*. The name should be short, clear, and descriptive.
- *Intent*. It includes the description of the re-engineering process, the results and why it is desirable.
- *Applicability*. It describes when a particular pattern is applicable and when it is not. It also comprises symptoms, reengineering goals and related patterns.
- *Motivation*. It includes the descriptions of the legacy system and its structure as well as the refactored system and the relation between them. This is done through the use of a concrete example.
- *Structure*. It describes the structure before and after re-engineering.
- *Process*. It includes the description of how to perform the re-engineering and possible variants.

2.2.3 Ontology Patterns

The idea of applying patterns to modelling ontologies was proposed by Clark et al. [CTP00]. They introduce "knowledge patterns" as a technique for helping construct axiom-rich, formal ontologies, based first on identifying and explicitly representing recurring patterns of knowledge in the ontology, and then on mapping those patterns onto domain-specific concepts in the ontology.

Since then, the ontology community has adopted the pattern idea. Relevant works on patterns are

- Semantic Web Best Practices and Deployment Working Group[43], which aims to provide hands-on support for developers of Semantic Web applications.

[39] http://hillside.net/patterns

[40] http://c2.com/ppr/index.html

[41] http://industriallogic.com/patterns/

[42] http://www.cetus-links.org/oo_patterns.html

[43] http://www.w3.org/2001/sw/BestPractices/OEP/

- The Ontology Design Patterns Public Catalog[44], which focuses on the biological knowledge domain.
- The Ontology Design Patterns (ODP) Portal[45], a Semantic Web portal dedicated to ontology design patterns.
- Linked Data Patterns[46], a catalogue of Linked Data [Biz09] patterns.

According to [GP08] Ontology Design Patterns are modelling solutions to solve a recurrent ontology design problem. Gangemi et al. distinguish different types of Ontology Design Patterns by grouping them into six families (see first level in Figure 2.5). Each family addresses different kind of problems and can be represented with different levels of formality. Next we briefly describe each family of patterns.

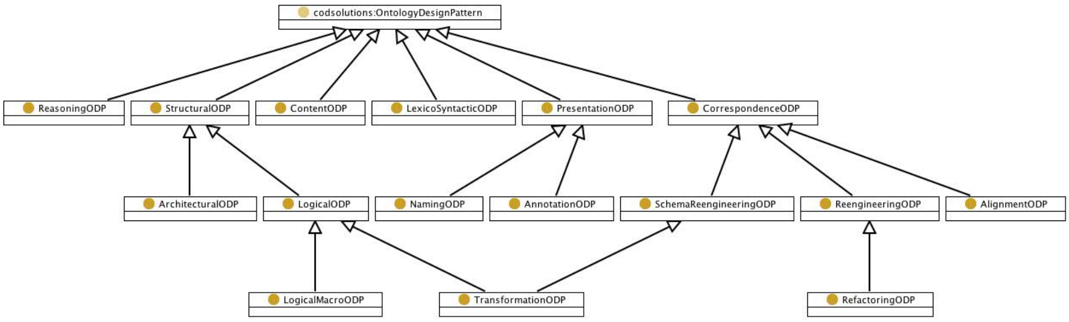


Figure 2.5: Ontology Design Pattern categorization [GP08]

- *Content ODPs* propose, on the one hand, patterns for solving design problems for the domain classes and, on the other hand, properties that populate an ontology, thus addressing content problems.
- *Structural OPs* include Logical OPs and Architectural OPs. Logical OPs are compositions of logical constructs that solve the problem of expressivity, while Architectural OPs are defined in terms of composition of Logical OPs used to affect the overall shape of the ontology.
- *Lexico-Syntactic ODPs* can be defined as linguistic structures or schemas that consist of certain types of words following a specific order, and that permit generalizing and extracting some conclusions about the meaning they express.
- *Reasoning ODPs* are applications of Logical OPs oriented to obtain certain reasoning results and based on the behaviour implemented in a reasoning engine.

[44] http://www.gong.manchester.ac.uk/odp/html/index.html

[45] http://ontologydesignpatterns.org

[46] http://patterns.dataincubator.org/book/

- *Presentation ODPs* deal with the usability and readability of ontologies from a user perspective. They are meant to be used as good practices that support the reuse of patterns by facilitating their evaluation and selection.

- *Correspondence ODPs* are templates to represent alignments between models. They include Schema Re-engineering ODPs, Re-engineering ODPs and Alignment ODPs. *Re-engineering ODPs* are transformation rules applied to create a new ontology starting from elements of a source model; *Refactoring ODPs* provide designers with rules for transforming an existing OWL-DL "source" ontology into a new OWL-DL "target" ontology.

2.3 Summary and Discussion

This chapter has presented an exhaustive analysis of the state of the art of the various topics dealt within this thesis, and discussed their limitations.

In this section we provide an overall summary of the open research problems identified and we have focused on such problems to provide methods and tools for reusing and re-engineering non-ontological resources with the aim of speeding up the ontology development.

The first step is to identify the resources that the methods and tools proposed are going to deal with. For this purpose we introduce the notions of **non-ontological resources** and **ontology** (see sections 1.1 and 1.1.2). Then, we put forward our categorization of non-ontological resources according to the three different features presented in Figure 5.1: (1) the type of non-ontological resource, which refers to the type of inner organization of the information; (2) the data model, that is, the design data model used to represent the knowledge encoded by the resource; and (3) the resource implementation.

The second step is the **selection of the most appropriate non-ontological resources** for ontology development. The analysis of the State of the Art reveals that there are not detailed guidelines on how to find the most suitable non-ontological resources for the development of ontologies. Most of the research studies assume that there are already suitable resources to use in the conversion. In conclusion, we can state that there is a clear need for some sort of methods, techniques and tools that help in the selection of the resources and that keep the provenance information of these resources.

The third step is the **transformation of the resources selected into ontologies**. In this step we can state that there is a clear need for some sort of re-engineering methods that (1) cope with the overall set of NORs (i.e., classification schemes, thesauri, and lexica) in an uniform way, independently of how it has been implemented; (2) include the three transformation approches (TBox, ABox and Population); (3) make explicit the hidden semantics in the relations of the NOR terms, by means of external resources in an semi-automatic way; and (4) provide finely detailed guidelines for the transformation, including information on who is in charge of performing a particular task/activity and when such a task/activity has to be carried out.

Additionally, we have reviewed the state of the art on software re-engineering, software re-engineering patterns, and ontology patterns. All along this thesis we intend to demonstrate that the application of re-engineering patterns for transforming non-ontological resources into ontologies has several advantages. The most representative are

- Improvement of the efficiency of the re-engineering process.
- Ease of the transformation process for both ontology engineers and domain experts.
- Improvement of the reusability of non-ontological resources.

Finally, a very important matter that we would like to emphasize is the lack of an integrated method (and technological support) that addresses all the previous issues.

Chapter 3

OBJECTIVES AND CONTRIBUTIONS

The goal of this thesis is to investigate methods and tools for reusing and re-engineering non-ontological resources when building ontologies, as opposed to custom-building new ontologies from scratch. With the thesis we have contributed to the NeOn Methodology Framework since it lies on this new paradigm. It presents a re-engineering model as well as a method and a technology for reusing and re-engineering non-ontological resources when building ontologies by means of re-engineering patterns. Figure 3.1 depicts a general overview of our contributions and the relationships between them. Next, we present the main contributions.

i) The definition of methodological guidelines for reusing non-ontological resources when building ontologies. These methodological guidelines provide support (1) for selecting the most appropriate non-ontological resources for ontology development; and (2) for describing and providing the provenance information of the ontology generated.

ii) The definition of methodological guidelines for re-engineering non-ontological resources into ontologies. The methodological guidelines (1) cope with the classification schemes, thesauri, and lexica, in an uniform way, independently of how those resources have been implemented; (2) are based on re-engineering patterns; (3) include the three transformation approches (TBox, ABox and Population); (4) make explicit the hidden semantics in the relations of the NOR terms by means of external resources in a semi-automatic way; and (5) provide support for the transformation, including information about who is in charge of performing a particular activity and when such an activity has to be carried out.

iii) The development of a library of patterns for re-engineering non-ontological resources into ontologies. These patterns cover classification schemes, thesauri, and lexica.

iv) The development of a software library, NOR_2O, that implements the transformations suggested by the re-engineering patterns.

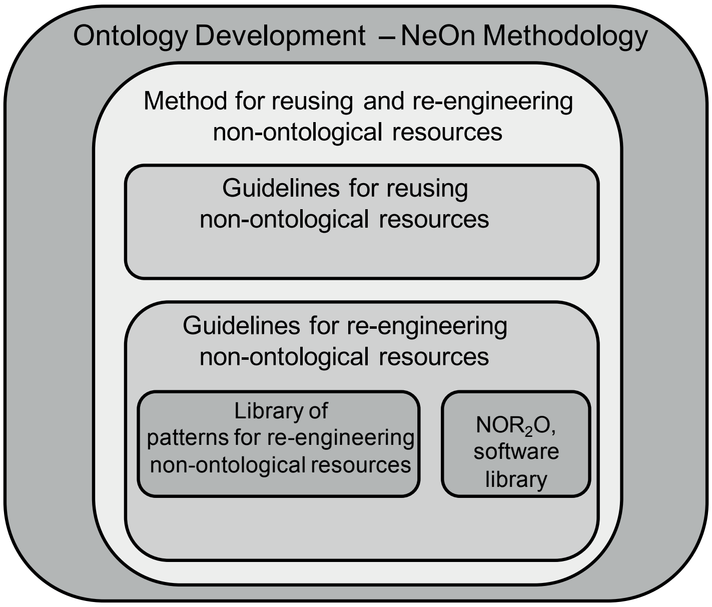

Ontology Development – NeOn Methodology
Method for reusing and re-engineering non-ontological resources
Guidelines for reusing non-ontological resources
Guidelines for re-engineering non-ontological resources
Library of patterns for re-engineering non-ontological resources
NOR₂O, software library

Figure 3.1: Thesis main contributions

3.1 Objectives

The general objective of the thesis is to provide domain independent, and resource independent methods and tools for speeding up the ontology development process and is achieved by reusing and re-engineering as much as possible available non-ontological resources. To fulfil this overall goal, we have decomposed it into the following methodological and technological objectives:

Methodological Objectives

O1. The definition of methodological aspects related to the **reuse of non-ontological resources for building ontologies**. We propose a method that describes a set of activities and serves as a guide for selecting the most suitable non-ontological resources to develop ontologies.

O2. The definition of methodological aspects related to the **re-engineering of non-ontological resources for building ontologies**. We propose a method that guides users through the transformation of a non-ontological resource into an ontology.

Technological Objectives

O3. The creation of **a library of patterns for re-engineering non-ontological resources into ontologies**. These patterns transform classification schemes, thesauri, and lexica. The re-engineering patterns follow the best practices of ontology engineering and use Logical and Content Ontology Design Patterns for generating OWL Lite ontologies or RDF instances. Moreover, the patterns rely on external resources for discovering the relationships among the non-ontological resource terms.

O4. The development of **a software library, NOR$_2$O, that implements the suggestions given by the re-engineering patterns**. The patterns have an associated software library, that performs the suggested transformations automatically.

3.2 Contributions to the State of the Art

We have tried to provide solutions to some of the open research and technological problems (see Chapter 2) identified in the scope of this thesis.

I. Regarding the **methodological guidelines for carrying out the non-ontological resource reuse process**, this thesis presents new advances in the state of the art in the following aspects:

C1. A definition of non-ontological resources. There is a wealth of non-ontological resources that embodies knowledge about some particular domains and that represents some degree of consensus for a user community. These resources present the form of free texts, textual corpora, web pages, standards, catalogues, web directories, classification schemes, thesauri, lexica and folksonomies, among others. The definition of non-ontological resource is provided in section 5.1.

C2. A categorization of non-ontological resources according to three different features: type of non-ontological resource, data model and implementation. The categorization is described in section 5.1. It should be noted that an accepted and agreed upon typology of non-ontological resources does not exist yet. This contribution is the result of the thorough analysis of the structures that NORs usually have.

C3. A metadata vocabulary for non-ontological resources, NoRMV. This vocabulary allows (1) describing the available non-ontological resources, and (2) including the provenance information in the ontology generated. The vocabulary is described in Section 5.2.

C4. A method for reusing non-ontological resources when building ontologies. The description of the method is included in Section 5.3.

II. Regarding the **methodological guidelines for carrying out the non-ontological resource re-engineering process**, the new advances in the state of the art are

C5. A re-engineering model for non-ontological resources. This model is based on the software re-engineering model presented in [Byr92] and depicted in Chapter 6.

C6. A method for re-engineering non-ontological resources when building ontologies with re-engineering patterns. This method is described in Chapter 6.

III. Problems in ontology engineering can be solved by applying common solutions (as experienced in software engineering); on the other hand, Ontology Design Patterns (ODPs) can support reusability on the design side. Our third objective belongs to the Ontology Design Patterns field. We propose a **library of re-engineering patterns (PR-NOR)**, which is included in the `ontologydesignpatterns.org` portal[1]. In this thesis we provide

C7. A set of patterns for re-engineering classification schemes into ontologies. These patterns take advantage of the classification scheme underlying data models. The data models identified for classification schemes are described in Chapter 7.

C8. A set of patterns for re-engineering thesauri into ontologies. These patterns take advantage of the thesaurus underlying data models. The data models identified for thesauri are described in Chapter 8.

C9. A set of patterns for re-engineering lexica into ontologies. These patterns take advantage of the lexicon underlying data model. The data models identified for lexica are described in Chapter 9.

IV. Finally, our fourth objective is to provide **technological support to the patterns for re-engineering non-ontological resources**. We present the following advance in the state of the art:

C10. A software library, NOR$_2$O, that implements the transformation suggested by the patterns. In this way, the software library covers the transformations of classification schemes, thesauri and lexica. These resources can be implemented in databases, XML files, flat files or spreadsheets. The description of this software library is included in Section 10.1.

The contributions are presented in the document as follows: first, Chapter 5 presents the contributions: (C1) a definition of non-ontological resource; (C2) a categorization of non-ontological resources; (C3) a metadata vocabulary for describing non-ontological resources; and (C4) a method for reusing non-ontological

[1] `http://ontologydesignpatterns.org/wiki/Submissions:ReengineeringODPs`

resources for building ontologies. Then, Chapter 6 presents (C5) our model for re-engineering non-ontological resources and (C6) a method for re-engineering non-ontological resources when building ontologies. Then, Chapters 7, 8, and 9 present the patterns for re-engineering classification schemes, thesauri, and lexica respectively (C7, C8, and C9). These contributions are the result of establishing which *semantic additions* (enrichments) have to be made after an initial transformation. Finally, Chapter 10 describes the technological support we provide for the model and method proposed, including the implementation of a software library, NOR$_2$O, that carries out the transformation process suggested by the patterns (C10).

3.3 Assumptions

The work described in this thesis is based on a set of assumptions described next.

A1. Some claims valid in software engineering and software re-engineering are also valid in ontology engineering.

A2. The Ontology Specification Activity was carried out previously, i.e. we started from a correctly created Ontology Requirements Specification Document (ORSD).

A3. The non-ontological resources to be reused and transformed are freely available and with no restriction of use.

A4. The non-ontological resources to be transformed are well designed and implemented.

A5. The quality of the ontologies generated can be measured as the similarity value of the ontologies generated against a gold standard ontology. This gold standard is created by human domain experts.

3.4 Hypotheses

Once the assumptions have been identified, the hypotheses of our work are described. These hypotheses cover the main features of the solutions proposed.

H1. The reuse and re-engineering of non-ontological resources, which have reached some degree of consensus in the community, will allow the development of ontologies in an easier and faster way.

H2. It is possible to define a unified method for transforming non-ontological resources into ontologies independently (1) of the type, data model or implementation of the resource, and (2) of the target ontology to be generated, i.e., ontology schema (TBox), ontology (TBox+ABox), or ontology instances (ABox).

H3. The method for re-engineering non-ontological resources is extensible and adaptable to other types of resources. It can be applied to any kind of non-ontological resource independently of its type, data model or implementation.

H4. It is possible to create re-engineering patterns that allow generating ontologies from available non-ontological resources, namely, classification schemes, thesauri, and lexica, in an uniform way, independently (1) of how they have been implemented; (2) of the target ontology to be generated, i.e., ontology schema (TBox), ontology (TBox+ABox), or ontology instances (ABox); (3) of the domain of the resources, that is, the patterns can be used to build ontologies in different domains.

H5. The re-engineering patterns proposed can be implemented in a software library that facilitates the work of ontology engineers when developing ontologies.

3.5 Restrictions

Finally, there is a set of restrictions that defines the limits of our contribution and establishes future research objectives. These restrictions are the following:

R1. The categorization of non-ontological resources covers semi-structured resources, but it does not cover unstructured resources, e.g., free text.

R2. NoRMV covers the description of the non-ontological resources according to the proposed categorization, but this thesis only includes the identified types, data models and implementations.

R3. The method for reusing non-ontological resources considers only semi-structured resources, but does not cover unstructured resources, e.g., free text.

R4. The method for re-engineering non-ontological resources covers the transformation of the whole resource, but does not cover the transformation of excerpts of the resource.

R5. The method for re-engineering non-ontological resources covers the transformation of one resource per time, but does not consider the integration of several resources simultaneously or one after the other.

R6. The patterns for re-engineering non-ontological resources do not generate ontologies with disjoint knowledge.

R7. The software library supports the following non-ontological resource implementations: database, XML, spreadsheets and flat files.

R8. The software library generates ontologies implemented in OWL Lite, and ontology instances in RDF.

R9. The discovery of the semantics of relations among the non-ontological resource terms occurs only in the English language.

R10. The whole set of techniques, with the exception of the discovery of relations, is independent of the language.

R11. The evaluation of our work is restricted to the use of the results in real cases providing specific feedback.

Table 3.1 summarizes the mapping between the objectives identified in Section 3.1 and the specific contributions. The table also summarizes for each contribution, the associated assumptions (c.f. Section 3.3), hypotheses (c.f. Section 3.4) and restrictions (c.f. Section 3.5) of the thesis.

Table 3.1: Mapping between objectives and contributions with associated assumptions, hypotheses and restrictions

Objective	**Contribution**	**([Assumptions],[Hypotheses],[Restrictions])**
O1. Methodological aspects related to the reuse of non-ontological resources for building ontologies.	**C1.** Definition of non-ontological resources.	([**A3**,**A4**], [], [**R1**])
	C2. A three level categorization of non-ontological resources.	([**A3**,**A4**], [], [**R1**])
	C3. NoRMV, a metadata vocabulary for non-ontological resources.	([**A3**], [**H1**], [**R1**,**R2**])
	C4. Method for reusing non-ontological resources when building ontologies.	([**A2**,**A3**,**A4**], [**H1**], [**R3**,**R11**])
O2. Methodological aspects related to the re-engineering of non-ontological resources for building ontologies.	**C5.** Re-engineering model for non-ontological resources.	([**A3**,**A4**], [**H3**,**H4**], [])
	C6. Method for re-engineering non-ontological resources when building ontologies.	([**A1**,**A3**,**A4**], [**H1**,**H2**,**H3**], [**R4**,**R5**,**R11**])
O3. Set of patterns for re-engineering non-ontological resources into ontologies.	**C7.** Set of patterns for re-engineering classification schemes into ontologies.	([**A1**,**A3**,**A4**], [**H4**], [**R6**,**R9**,**R10**,**R11**])
	C8. Set of patterns for re-engineering thesauri into ontologies.	([**A1**,**A3**,**A4**], [**H4**], [**R6**,**R9**,**R10**,**R11**])
	C9. Set of patterns for re-engineering lexica into ontologies.	([**A1**,**A3**,**A4**], [**H4**], [**R6**,**R9**,**R10**,**R11**])
O4. Software library that implements the suggestions made by the re-engineering patterns.	**C10.** A software library, NOR$_2$O, that implements the transformations made by the re-engineering patterns.	([**A2**,**A4**], [**H5**], [**R7**,**R9**,**R11**])

Chapter 4

RESEARCH METHODOLOGY

This chapter presents the research methodology we have used when designing the method for reusing and re-engineering non-ontological resources into ontologies, as well as the main requirements that guide its development.

Since the work presented in this thesis is a subset of the NeOn Methodology [SF10], we follow the same research methodology (see Chapter 4 [SF10]) used for the creation of such methodology, which we have tried to specialize in our method.

4.1 General Framework for Describing the Method

For designing the method we have followed the "divide and conquer" strategy, that is, the general problem to be solved is decomposed into different subproblems. Then to solve each subproblem different strategies and alternatives are provided. Finally, to obtain the solution to the general problem, i.e., speeding up the ontology development process by reusing and re-engineering as much as possible available non-ontological resources, the solutions to the different subproblems are combined.

The subproblems identified are (1) the selection of the most appropriate non-ontological resources for building ontologies; (2) the transformation of the non-ontological resources selected into ontologies; (3) the techniques used for such transformation; and (4) the technological support for the method.

We introduce prescriptive methodological guidelines for reusing non-ontological resources when building ontologies, as described in Chapter 5, as a solution to subproblem (1). We provide methodological guidelines for re-engineering non-ontological resources into ontologies, as described in Chapter 6, as a solution to subproblem (2). We present a set of Patterns for Re-engineering Non-Ontological Resources in Chapters 7, 8, and 9, as solution to subproblem (3). We also present a software library that implements the transformation process suggested by the patterns, and a pattern library as solution to subproblem (4). The pattern library is available at the ODP portal[1],

[1] http://ontologydesignpatterns.org

In order to obtain the **methodological guidelines for reusing non-ontological resources**, we were grounded in the following approaches, as presented graphically in Figure 4.1.

- *Existing categorization of resources*. In this case, we analysed from [MS01, SAd$^+$07, GPS98, Hod00] different *ad-hoc* categorization of resources.

- *Previous practices and experiences*. Here, we used our previous experiences in the development of ontologies within several European and National funded projects, such as the REIMDOC Project[2] (FIT340100-2004-022), and the Knowledge Web Project[3] (FP6-507482). We made a retrospective analysis of the processes and activities performed within these projects to get a preliminary set of informal steps, which were refined, improved and completed to provide full methodological guidelines for each process, activity, and task.

- *Available ad-hoc methods*. In this case, we used the *ad-hoc* methods [MPBS06, BA05, BHM$^+$05] that provide guidelines for reusing existing resources when building ontologies.

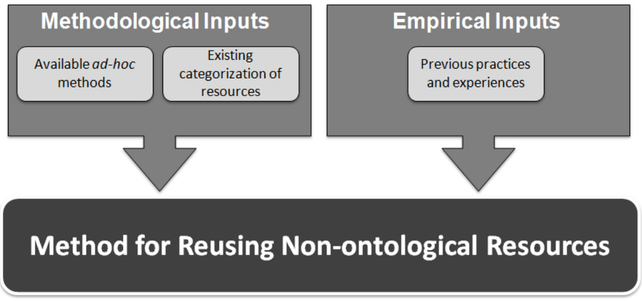

Methodological Inputs
Available ad-hoc methods
Existing categorization of resources
Empirical Inputs
Previous practices and experiences
Method for Reusing Non-ontological Resources

Figure 4.1: Inputs considered when developing the method for reusing non-ontological resources

To obtain the **methodological guidelines for re-engineering non-ontological resources**, we were grounded in the following approaches, as presented graphically in Figure 4.2.

- *Available ad-hoc methods*. In this case, we used the *ad-hoc* methods, described in Section 2.1, that provide guidelines for re-engineering resources. Some of the most representative are Hepp et al. [HdB07], van Assem et al. [vAGS06], Gangemi et al. [GNV03, GGMO03], van Assem et al. [vAMMS06], and Soergel et al. [SLL$^+$04]. We performed an analysis of the guidelines they propose to extract and improve a preliminary set of guidelines.

[2] http://reimdoc.atosorigin.es/

[3] http://knowledgeweb.semanticweb.org/

- *Previous practices and experiences*. Here, we used our previous experiences in the development of ontologies within several European and National funded projects, such as, the REIMDOC Project[4] (FIT340100-2004-022), the Knowledge Web Project[5] (FP6-507482), and the NeOn Project[6] (FP6-027595). We made a retrospective analysis of the processes or activities performed within such projects to get a preliminary set of informal steps, which were refined, improved and completed to provide full methodological guidelines for each process, activity, and task.

- *Available Software Re-engineering practices*. In this case, we based our work on the re-engineering model proposed by Byrne [Byr92].

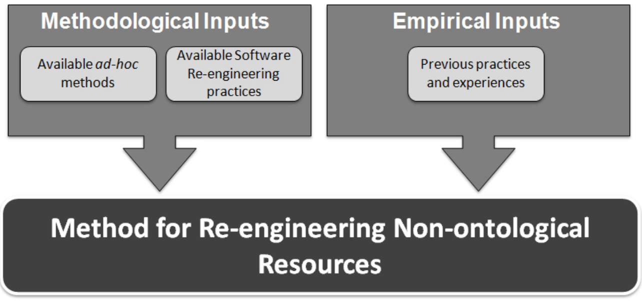

Methodological Inputs
Available *ad-hoc* methods
Available Software Re-engineering practices
Empirical Inputs
Previous practices and experiences
Method for Re-engineering Non-ontological Resources

Figure 4.2: Inputs considered when developing the method that allows re-engineering non-ontological resources

To obtain the set of **patterns required for re-engineering non-ontological resources**, we were grounded in the following best practices and patterns, as presented graphically in Figure 4.3.

- *Previous practices and experiences*. Here we used the practices we describe in Section 2.1. Additionally, within the SEEMP project[7] (FP6-027347) we built some *ad-hoc* wrappers for transforming existing non-ontological resources into ontologies. Then, we transformed ten non-ontological resources and identified some common data structures for storing and organizing the resources. These common data structures, also known as data models [Car02], are abstract models that describe how data is represented and accessed. For every data model we can define a process with a well-defined sequence of activities to extract the NORs terms and then map them to the conceptual model of an ontology. Each process can be expressed as a pattern for re-engineering NORs.

[4] http://reimdoc.atosorigin.es/

[5] http://knowledgeweb.semanticweb.org/

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

[7] http://seemp.org

- *Previous practices in the Ontology Engineering community*. As we mentioned in Section 2.2.3, the ontology community is adopting the use of design patterns for modelling ontologies. We extend the current patterns in the Ontology Engineering field with patterns for re-engineering non-ontological resources into ontologies. These re-engineering patterns make use of the logical and content patterns, from the ODP portal, for generating the ontologies; therefore, the re-engineering patterns follow the best practices of the community.

- *Available Software Re-engineering patterns*. In the software engineering community, it is well known that the reuse of resources helps to reduce costs and to disseminate good practices. This also holds for ontology engineering, where the reuse of existing knowledge can be done either by directly reusing resources as they are, or after performing a reengineering process. The underlying principle is that reuse allows saving time and money, and promotes the application of good practices. Therefore, for consolidating our patterns we have applied the concept of software re-engineering pattern.

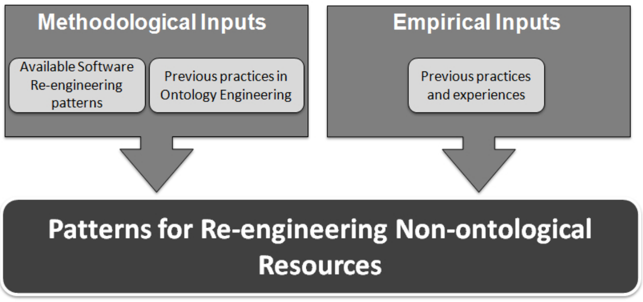

Methodological Inputs
Available Software Re-engineering patterns
Previous practices in Ontology Engineering
Empirical Inputs
Previous practices and experiences
Patterns for Re-engineering Non-ontological Resources

Figure 4.3: Inputs considered to obtain the patterns for re-engineering non-ontological resources

4.2 Description of the Processes

As mentioned before, our method consists of a set of methodological guidelines for the processes, activities and tasks involved in Scenario 2 of the NeOn Methodology [SF10, GPSF09]. For each process included in this method we provide a *filling card* [SF10, GPSF09] describing

- Definition, which is based on the NeOn Glossary of Activities [SF10, SFGP08].

- Goal, which explains the main objective intended to achieve by the process or the activity.

- Input, which includes the resources needed to carry out the process or the activity.

- Output, which includes the results obtained after carrying out the process or the activity.
- Who, which identifies the people or teams involved in the process or the activity.
- When, which explains in which moment the process or the activity should be carried out.
- How, which includes details for carrying out the process or the activity in a prescriptive manner. A graphical workflow on how the process or the activity should be carried out is also included, with the inputs, outputs and actors involved.

4.3 Requirements for the Proposed Method

The method presented in this thesis must fulfil a set of requirements that can be grouped into two main types: generic and specific requirements. The generic requirements are those that any method must fulfil, while the specific requirements of a given method are determined by factors such as the domain where the method is applied as well as cases, situations or problems it deals with. The following sections present the requirements we have considered for the design of our method. The requirements are based on those presented by Paradela [PG01]. It is worth mentioning that the requirements identified by Paradela are specific to methodologies, and that Suarez Figueroa has already demonstrated those requirements for the NeOn Methodology [SF10]. In this thesis we refine the requirements for our particular method.

4.3.1 Generic Requirements

- **Generality**. A method should be general enough and not be driven to solve *ad-hoc* cases or problems. Thus our method tackles the development of ontologies by reusing and re-engineering non-ontological resources.
- **Completeness**. A method must consider all the cases presented and propose solutions to all of them. In this sense, the method here proposed considers classification schemes, thesauri and lexica; and presents methodological guidelines and patterns for these non-ontological resources.
- **Effectiveness**. A method should solve adequately the cases proposed, independently of the person using such a method. Therefore, it should be more prescriptive than descriptive. Thus our goal is to describe the method in a simple way, and any person (a software developer or an ontology practitioner) should be able to understand and follow it without any special effort.

- **Efficiency**. A method must be efficient, that is, be able to achieve its objective. This means that the method should allow the construction of ontologies with fewer resources (time, money, etc.) and better quality. We will describe and carry out the necessary experiments using the method for validating this requirement.

- **Consistency**. A method must produce the same set of results for the same problem, independently of who employs it. Thus, our method identifies which the outputs of the different processes, activities and tasks should be, or who the different users involved in the development of ontologies are. We will validate that the same set of outputs is obtained after applying the method in several cases.

- **Finiteness**. The number of the elements that compose a method and the number of activities must be finite, i.e., consuming a reasonable period of time. Our method consists of a finite set of processes, activities and tasks. The number of elements used to describe them is also finite.

- **Discernment**. A method must be composed of a small set of structural, functional and representational components. Thus, the method here proposed includes

  - A categorization of non-ontological resources (structural component).
  - Processes, activities, tasks, inputs, outputs and restrictions (funcional components).
  - A set of patterns for re-engineering non-ontological resources into ontologies (representational components).

- **Environment**. Methods can be classified into scientific and technological. In scientific methods ideas are validated, whereas in technological methods artefacts are built. In our case, since the main result after applying the method is a technical product, i.e., an ontology, our method can be considered as a technological one.

- **Transparency**. A method must be like a white box, so that we can know in every moment the active processes or activities being performed, who is performing them, etc. The method here presented explicitly defines the actors, inputs, and outputs of each process, as well as activities and tasks.

4.3.2 Specific Requirements

- The method should allow performing the transformation approaches identified in Section 2.1.1.3, namely, TBox, ABox and population.

- The method should cover any kind of non-ontological resource. Thus, we can apply the method to other types of resources not contemplated in this thesis.

- The method should be automatable, because the size of the non-ontological resources can be huge. It should have technological support that permits the automatization of the activities and tasks involved, thus saving time and effort when dealing with very large non-ontological resources.

Chapter 5

REUSING NON-ONTOLOGICAL RESOURCES

As stated in the introduction of this thesis, our goal is to speed up the ontology development process by reusing available non-ontological resources that have been agreed upon by a particular community. In this sense, we have to identify first which non-ontological resources we are going to work with. However, as we discussed in Section 1.1.2 an accepted and agreed upon typology of non-ontological resources does not exist yet. Therefore, in this section we start by describing our categorization of non-ontological resources. Then, we present our Non-ontological Resource Metadata Vocabulary (NoRMV) for depicting the available non-ontological resources. This vocabulary will be included later on in the ontology generated as provenance information. Finally, we set forth the methodological guidelines devised for reusing non-ontological resources.

5.1 Non-ontological Resources

Non-ontological Resources[1] (NORs) are knowledge resources whose semantics has not yet been formalized by an ontology. There is a considerable 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, thesaurus, lexicon, etc. NORs have related 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 have been agreed on for building ontologies can have several benefits, e.g. interoperability in terms of the

[1] Along this thesis we use either *NOR* or *Non-ontological resource* without distinction.

vocabulary used, information browse/search, decrease of the knowledge acquisition bottleneck, and reuse, among others.

As already stated in Section 1.1.1, an accepted and agreed upon typology of NORs does not exist. Therefore, one of the contributions of this thesis is the categorization of NORs, according to the following three features presented in Figures 5.1: (1) type of NOR, which refers to the type of inner organization of the information; (2) data model, that is, the design data model used to represent the knowledge encoded by the resource; and (3) resource implementation.

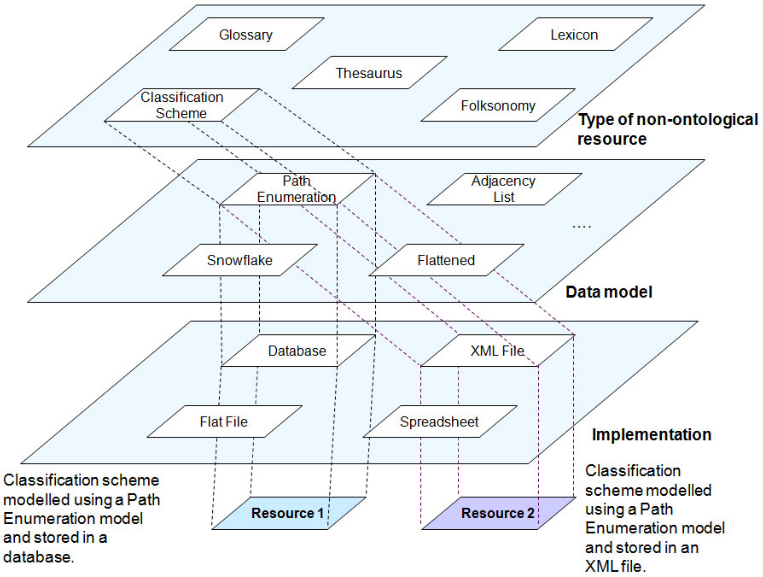

Glossary
Lexicon
Thesaurus
Classification Scheme
Folksonomy
Type of non-ontological resource
Path Enumeration
Adjacency List
....
Snowflake
Flattened
Data model
Database
XML File
Flat File
Spreadsheet
Implementation
Classification scheme modelled using a Path Enumeration model and stored in a database.
Resource 1
Resource 2
Classification scheme modelled using a Path Enumeration model and stored in an XML file.

Figure 5.1: Non-ontological resource categorization

According to the **type of NORs** we classify them into

- *Glossaries*: A glossary is an alphabetical list of terms or words found in or related to a specific topic or text. It may or may not include explanations, and its vocabulary may be monolingual, bilingual or multilingual [WB97]. An example of glossary is the FAO Fisheries Glossary[2].

- *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 [Hir04]. A fine example is WordNet[3], the best known computational lexicon of English.

- *Classification schemes*: A classification scheme is the descriptive information of an arrangement or division of objects into groups according to the characteristics that the objects have in common [ISO04]. A good example is

[2] http://www.fao.org/fi/glossary/default.asp

[3] http://wordnet.princeton.edu/

the Fishery International Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP)[4].

- *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 retrieving articles in large databases [ISO86]. An example of thesaurus is the AGROVOC[5] thesaurus.

- *Folksonomies*: Folksonomies are Web 2.0 systems that users employ to upload and annotate their content effortlessly and without requiring any expert knowledge[6]. This simplicity has made folksonomies widely successful, and this success, in its turn, has resulted in a massive amount of user-generated and user-annotated web content. The main advantage of folksonomies is the implicit knowledge they contain. When users tag resources with one or more tags, they assign these resources the meaning of the tag. Furthermore, the co-occurrence of tags implies a semantic correlation among them. An example of how folksonomies are used can be seen in the *del.icio.us*[7] website.

The knowledge encoded by the resource can be represented in different ways, known as data models. A data model [Car02] is an abstract model that describes how data is represented and accessed. There are three types: (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 thesis we will use the term data model when referring to the logical data model. With regard 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***, shown in Figure 5.2. These data models will be described in detail in Chapter 7.

- *Path Enumeration* [Bra05]: A path enumeration model (see Figure 5.2-b)) is a recursive structure for hierarchy representations and is 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 node code in the path from the root to the node.

- *Adjacency List* [Bra05]: An adjacency list model is a recursive structure for hierarchy representations comprising a list of nodes with a linking column to their parent nodes. Figure 5.2-c) shows this model.

[4] http://www.fao.org/figis/servlet/RefServlet

[5] http://www.fao.org/agrovoc/

[6] http://www.vanderwal.net/folksonomy.html

[7] http://del.icio.us/

- *Snowflake* [MZ06]: 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 column linked to its parent node. Figure 5.2-d) shows this model.
- *Flattened* [MZ06]: A flattened model is a denormalized structure for hierarchy representations. The hierarchy is represented by a table where each hierarchy level is stored in a different column. Figure 5.2-e) shows this model.

Next we present two ***data models for thesauri***. These data models are described in detail in Chapter 8.

- *Record-based model* [Soe95]: A record-based model is a denormalized structure that for every term it uses a record with information about the term, such as synonyms, broader, narrower and related terms. This model looks like the flattened model for classification scheme.
- *Relation-based model* [Soe95]: 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.

Next we present a ***data model for lexica***. These data models are described in detail in Chapter 9.

- *Record-based model* [Soe95]: This model can also be used for lexicons, because the use of a record for every lexical resource and information about that lexical resource is possible.
- *Relation-based model* [Soe95]: It can also be used for lexicons, because the storage of information about the lexicon in individual pieces is possible.

According to the **implementation** we classify NORs into

- *Databases* : A database is a structured collection of records or data stored in a computer system.
- *Spreadsheets* : An electronic spreadsheet consists of a matrix of cells where a user can enter formulas and values.
- *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 where the user defines the names of the nodes.

- *Flat file* : A flat file is a file usually read or written sequentially. In general, a flat file is a file containing records with no structured inter-relationships.

In summary, Figure 5.1 shows how a given type of NOR can be modelled following one or more data models, each of which implemented in different ways at the implementation layer. Figure 5.1 shows, as an example, a classification scheme modelled 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 presented with a real life classification scheme, we use an excerpt from the FAO water area classification presented in Figure 5.2-a). This classification schema is modelled following a path enumeration model (Figure 5.2-b)), an adjacency list model (Figure 5.2-c)), a snow-flake model (Figure 5.2-d)), and a flattened model (Figure 5.2-e)). Figure 5.2-f) presents an XML implementation of the adjacency list model and Figure 5.2-g) 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[8] and Electronic Data Interchange, EDI[9].

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

- The United Nations Standard Products and Services Code, UNSPSC[10], is a classification scheme, modelled with the path enumeration data model and stored in a relational database.
- WordNet[11], a lexical database for English, is a lexicon, modelled with the relation-based data model and stored in several implementations; a particular implementation of it is a relational database.
- UMLS[12]is a very large, multi-purpose, multilingual thesaurus that contains information about biomedical and health related concepts. It is modelled with the record-based model and stored in a flat file.
- MeSh[13], the Medical Subject Headings, is a classification scheme, modelled with the path enumeration data model.
- The Art and Architecture Thesaurus[14] is modelled with the record-based data model and implemented in XML.
- The ISCO-08 International Standard Classification of Occupations[15] is a classification scheme modelled with the path enumeration data model and implemented in a database and spreadsheet.

[8] http://www.rosettanet.org/

[9] http://www.edibasics.co.uk/

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

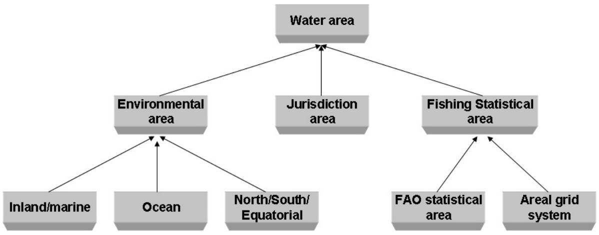
[11] http://wordnet.princeton.edu/

[12] http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html

[13] http://www.nlm.nih.gov/mesh/

[14] http://www.getty.edu/research/tools/vocabularies/aat/index.html

[15] http://www.ilo.org/public/english/bureau/stat/isco/index.htm



a) Excerpt of the Water Area classification scheme.

ID	CSI_Name
20000	Water area
20000.21000	Environmental area
20000.24020	Jurisdiction area
20000.22000	Fishing Statistical area
20000.21000.21001	Inland/marine
20000.21000.21002	Ocean
20000.21000.21003	North/South/Equatorial
20000.22000.22001	FAO statistical area
20000.22000.22002	Areal grid system

b) Path Enumeration data model

ID	CSI_Name	Parent
20000	Water area	
21000	Environmental area	20000
24020	Jurisdiction area	20000
22000	Fishing Statistical area	20000
21001	Inland/marine	21000
21002	Ocean	21000
21003	North/South/Equatorial	21000
22001	FAO statistical area	22000
22002	Areal grid system	22000

c) Adjacency List data model

First Level	
ID	CSI_Name
20000	Water area

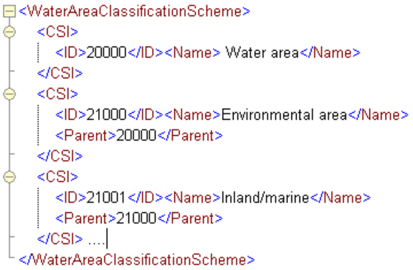
Second Level		
ID	First Level ID	CSI_Name
21000	20000	Environmental area
24020	20000	Jurisdiction area
22000	20000	Fishing Statistical area

Third Level		
ID	Second Level ID	CSI_Name
21001	21000	Inland/marine
21002	21000	Ocean
21003	21000	North/South/Equatorial
22001	22000	FAO statistical area
22002	22000	Areal grid system

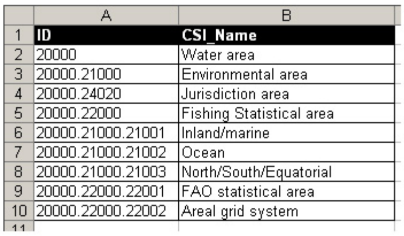
First Level		Second Level		Third Level	
ID	CSI_Name	ID	CSI_Name	ID	CSI_Name
20000	Water area	21000	Environmental area	21001	Inland/marine
20000	Water area	21000	Environmental area	21002	Ocean
20000	Water area	21000	Environmental area	21003	North/South/Equatorial
20000	Water area	22000	Fishing Statistical area	22001	FAO statistical area
20000	Water area	22000	Fishing Statistical area	22002	Areal grid system
20000	Water area	24020	Jurisdiction area		

d) Snowflake data model

e) Flattened data model



```
<WaterAreaClassificationScheme>
  <CSI>
    <ID>20000</ID><Name> Water area</Name>
  </CSI>
  <CSI>
    <ID>21000</ID><Name>Environmental area</Name>
    <Parent>20000</Parent>
  </CSI>
  <CSI>
    <ID>21001</ID><Name>Inland/marine</Name>
    <Parent>21000</Parent>
  </CSI> ....
</WaterAreaClassificationScheme>
```



	A	B
1	ID	CSI_Name
2	20000	Water area
3	20000.21000	Environmental area
4	20000.24020	Jurisdiction area
5	20000.22000	Fishing Statistical area
6	20000.21000.21001	Inland/marine
7	20000.21000.21002	Ocean
8	20000.21000.21003	North/South/Equatorial
9	20000.22000.22001	FAO statistical area
10	20000.22000.22002	Areal grid system
11		

f) XML implementation for the Adjacency List data model.

g) Spreadsheet implementation for the Path Enumeration data model.

Figure 5.2: Example of classification scheme

- The European Training Thesaurus, ETT[16], is modelled with the record-based datamodel and implemented in XML.
- The Classification of Fields of Education and Training, FOET[17], is a classification scheme modelled with path enumeration data model and implemented in XML and spreadsheet.
- The Aquatic Sciences and Fisheries Abstracts thesaurus, ASFA[18], is modelled with the record-based data model and implemented in XML.
- The AGROVOC thesaurus[19] is modelled with the relation-based data model and implemented in a database.
- The Fisheries Global Information System, FIGIS[20], 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[21], is a classification scheme modelled with the flattened data model and implemented in a spreadsheet.

5.2 Non-ontological Resource Metadata Vocabulary

As stated before there is a large amount of NORs that embody knowledge on some particular domains and that represent some degree of consensus. Currently, most of these NORs are in its pure form without any additional information, e.g., domain of interest or authorship information, like the one provided by Dublin Core[22] for text documents, or OMV[23] for ontologies[24]. This burden makes it difficult for academia and industry to identify, find and reuse NORs effectively and efficiently. As a consequence, the reuse of NORs for building ontologies is nowadays a hard task, if not impossible.

We argue that metadata, when meaning machine processable information for the Web, helps to improve NORs accessibility and reusability. Besides, it can provide other useful resource information to support maintenance. Thus, we claim here that metadata not only helps when applied (or, attached) to documents or ontologies, but also when applied to NORs themselves. There is a great demand in the field for a NOR metadata standard, a standard that would permit, amongst other things, the access and reuse of NORs.

In this thesis we propose a metadata standard reflecting the most relevant properties of NORs for supporting their reuse, which is the so called Non-Ontological

[16] http://libserver.cedefop.europa.eu/ett/en/

[17] http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=DSP_GEN_DESC_VIEW_NOHDR&StrNom=EDU_TRAINI&StrLanguageCode=EN

[18] http://www.fao.org/fishery/asfa/8/en

[19] http://aims.fao.org/website/AGROVOC-Thesaurus/sub

[20] http://www.fao.org/figis/servlet/RefServlet

[21] http://en.istat.it/

[22] http://dublincore.org/

[23] http://omv2.sourceforge.net/

[24] http://dublincore.org/

Resource Metadata Vocabulary (NoRMV). This vocabulary allows (1) describing the non-ontological resources available, and (2) including in the ontology generated the provenance information by extending the Ontology Metadata Vocabulary (OMV) [HPS05].

5.2.1 NoRMV Core Metadata Entities

The main classes and properties of the NoRMV are illustrated in Figure 5.3[25]

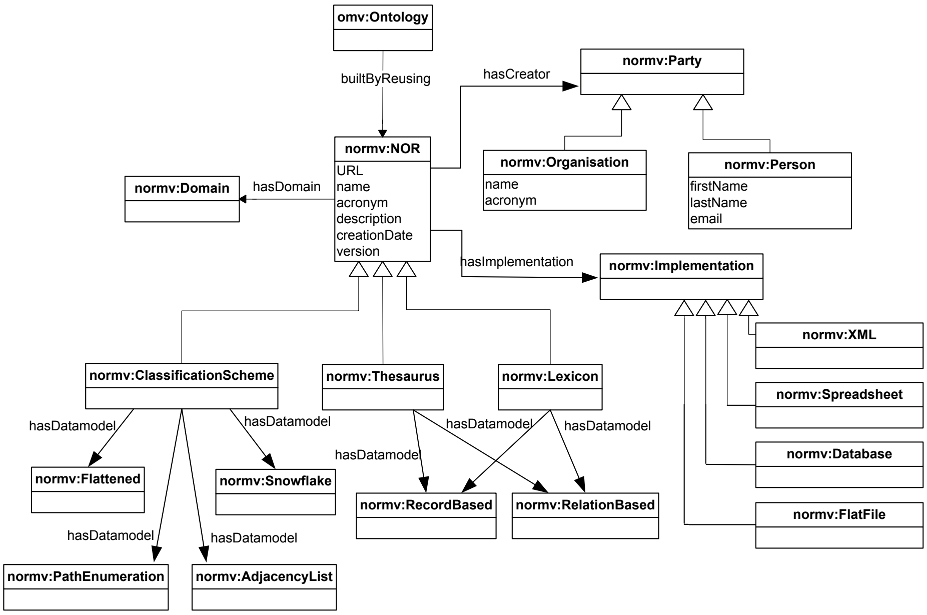


Figure 5.3: NoRMV: A metadata vocabulary for non-ontological resources

Besides the main NOR class, the metadata model contains elements describing various aspects related to the creation, management and usage of a NOR. We briefly discuss these in the following section. The NOR class includes as datatype properties, the URL, name, acronym, description, creation date and version of the non-ontological resource. As already described in section 5.1, we classify NOR into Classification Scheme, Thesaurus, and Lexicon, among others. Regarding the datamodel, a ClassificationScheme may have a Flattened, a PathEnumeration, an AdjacencyList, or a Snowflake data model. On the other hand, a Thesaurus may have RecordBased or RelationBased data model. And the same occurs to a Lexicon, that is, it may have RecordBased or RelationBased data model. Regarding the Implementation, it may be classified into XML, Spreadsheet, Database, and FlatFile. In addition, a NOR has a Domain, and a creator, Person(s) or Organization(s).

[25] Please note that not all classes and properties are included. The ontology is available for downloading at http://mccarthy.dia.fi.upm.es/normv

We group these two classes (Person and Organization) under the generic class Party by a subClassOf relation. Finally, in order to include the provenance information in the ontology generated, we relate the Ontology class, taken from OMV, to the NOR class by means of the builtByReusing relation.

Next, as an illustrative example we present an excerpt of the NoRMV metadata from the Water Area classification scheme (see Figure 5.2), modelled with the Path Enumeration data model, and implemented in a database.

```
<normv:ClassificationScheme rdf:about="#WaterAreas">
        <normv:name>Water Areas Classification Scheme</normv:name>
        <normv:URL>http://www.fao.org/figis/servlet/RefServlet</normv:URL>
        <normv:acronym>FAO WAC</normv:acronym>
        <normv:hasDatamodel rdf:resource="&normv;PathEnumeration"/>
        <normv:hasImplementation rdf:resource="&normv;Database"/>
        <normv:hasCreator rdf:resource="#FAO"/>
</normv:ClassificationScheme>
<normv:Organisation rdf:about="#FAO">
        <normv:name>
                Food and Agriculture Organization of the United Nations
        </normv:name>
        <normv:acronym>FAO</acronym>
</normv:Organisation>
```

5.3 Method for Reusing Non-ontological Resources

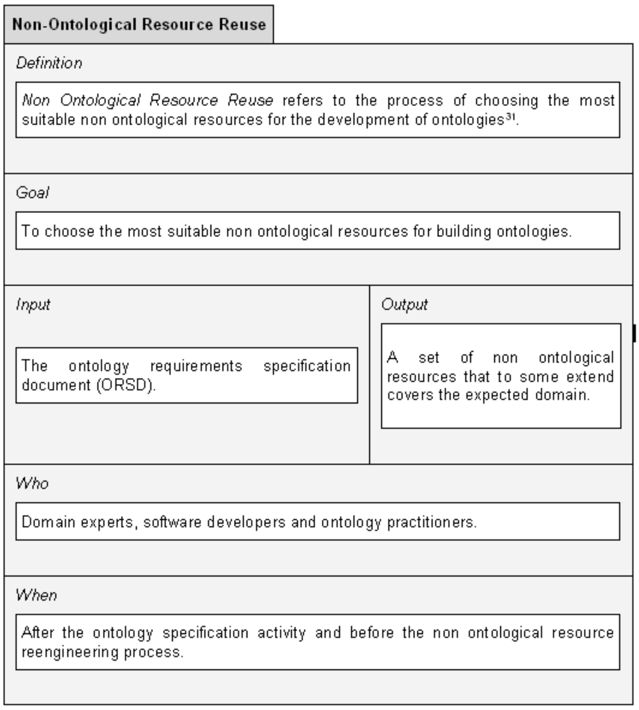
Once we have defined and categorized the non-ontological resources to be dealt with, we present the methodological guidelines for reusing them. The goal of the Non-Ontological Resource Reuse process is to choose the most suitable non-ontological resource for building ontologies. Domain experts, software developers and ontology practitioners carry out this process by taking as input the ontology requirements specification document (ORSD)[26] to find the most suitable non-ontological resources for the development of ontologies. The output of the process is a set of non-ontological resources that, to some extent, covers the expected domain. Figure 5.4 shows the filling card used in the process of reusing non-ontological resources, which includes the definition, goal, input, output, performer of the process and period of execution.

This process includes the activities and tasks presented in Figure 5.5 and is explained next.

5.3.1 Activity 1. Search Non-ontological Resources

The goal of the activity is to search non-ontological resources from highly reliable Web sites, domain-related sites and resources within organizations. Domain experts, software developers and ontology practitioners carry out this activity taking as input the ORSD. They use the terms that have the highest frequency in the ORSD to search for the candidate non-ontological resources that cover the desired

[26] This document is the outcome of the Ontology Specification Activity [SFGPVT09]



Non-Ontological Resource Reuse

Definition

Non Ontological Resource Reuse refers to the process of choosing the most suitable non ontological resources for the development of ontologies[31].

Goal

To choose the most suitable non ontological resources for building ontologies.

Input

The ontology requirements specification document (ORSD).

Output

A set of non ontological resources that to some extend covers the expected domain.

Who

Domain experts, software developers and ontology practitioners.

When

After the ontology specification activity and before the non ontological resource reengineering process.

Figure 5.4: Non-ontological resource reuse filling card

terminology. The activity output is a set of candidate non-ontological resources that might present any of the identified typologies described in Section 5.1.

5.3.2 Activity 2. Assess the Set of Candidate Non-ontological Resources.

The goal of the activity is to assess the set of candidate non-ontological resources. Domain experts, software developers and ontology practitioners carry out this activity, taking as input the set of candidate non-ontological resources. We propose to consider the following measurable criteria: (1) coverage, (2) precision plus two subjective criteria (3) quality[27] and (4) consensus. These criteria are inspired on the work proposed in [GCCL06].

5.3.2.1 Task 2.1 Extract lexical entries

The goal of this task is to extract the lexical entries of the non-ontological resources. The task is carried out by software developers and ontology practitioners by taking as input the non-ontological resources for extracting their lexical entries with terminology extraction tools.

[27] A deep analysis of the quality of the resource is out of the scope of this thesis

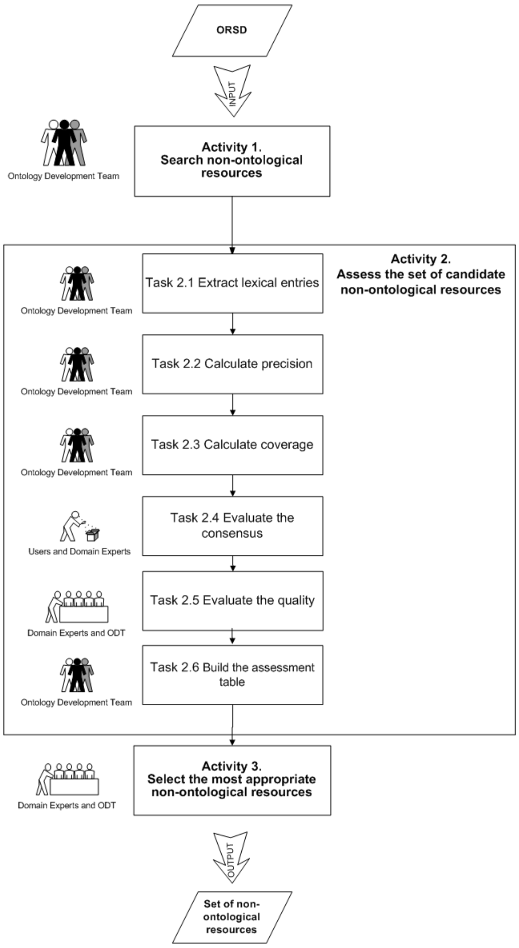

ORSD
INPUT
Ontology Development Team
Activity 1.
Search non-ontological resources
Activity 2.
Assess the set of candidate non-ontological resources
Ontology Development Team
Task 2.1 Extract lexical entries
Ontology Development Team
Task 2.2 Calculate precision
Ontology Development Team
Task 2.3 Calculate coverage
Users and Domain Experts
Task 2.4 Evaluate the consensus
Domain Experts and ODT
Task 2.5 Evaluate the quality
Ontology Development Team
Task 2.6 Build the assessment table
Domain Experts and ODT
Activity 3.
Select the most appropriate non-ontological resources
OUTPUT
Set of non-ontological resources

Figure 5.5: Activities for the non-ontological resource reuse process

5.3.2.2 Task 2.2 Calculate precision

The goal of this task is to calculate the precision of the candidate non-ontological resources. Precision is a measure widely used in information retrieval [BYRN99] and is defined as the proportion of retrieved material that is actually relevant. This task is carried out by software developers and ontology practitioners by taking as input the lexical entries extracted for the non-ontological resources and the terminology gathered in the ORSD. To adapt this precision measure into our context we need to define

- *NORLexicalEntries* as the set of lexical entries extracted from the non-ontological resource.
- *ORSDTerminology* as the set of identified terms included in the ORSD.

Now we can define the precision, in our context, as the proportion of the lexical entries of the non-ontological resource that are included in the identified terms of the ORSD over the lexical entries of the non-ontological resource. This is expressed as follows:

$$Precision = \frac{|\{NORLexicalEntries\} \cap \{ORSDTerminology\}|}{|\{NORLexicalEntries\}|}$$

5.3.2.3 Task 2.3 Calculate coverage

The goal of this task is to calculate the coverage of the non-ontological resources. Coverage is based on the recall measure used in information retrieval [BYRN99]. Recall is defined as the proportion of relevant material actually retrieved in answer to a search request. This task is carried out by software developers and ontology practitioners by taking as input both the lexical entries extracted from the non-ontological resources and the terminology gathered in the ORSD. To adapt this measure into our context, we use the aforementioned definitions of *NORLexicalEntries* and *ORSDTerminology*. In this context, coverage is the proportion of the identified terms of the ORSD that are included in the lexical entries of the non-ontological resource over the identified terms of the ORSD. This is expressed as follows:

$$Coverage = \frac{|\{NORLexicalEntries\} \cap \{ORSDTerminology\}|}{|\{ORSDTerminology\}|}$$

5.3.2.4 Task 2.4 Evaluate the consensus

The goal of this task is to evaluate the consensus of the non-ontological resources. Consensus is a subjective and not quantifiable criterion. This task is carried out by domain experts, taking as input the non-ontological resources for stating whether the non-ontological resources contain terminology agreed upon by the community or not. We propose a preliminary starting point for this evaluation. Domain experts have to check whether the resource is coming from

- A standardization body or any entity whose primary activity is to develop, coordinate, promulgate, revise, amend, reissue, or otherwise maintain standards. For example: the International Organization for Standardization (ISO), the American National Standards Institue (ANSI), the World Wide Web Consortium (W3C).

- Large organizations across national governments, such as the Food and Agriculture Organization of the United Nations (FAO), the World Health Organization (WHO), the United Nations Educational, the Scientific and Cultural Organization (UNESCO), the International Olympic Commitee (IOC).
- A large enough user community to make it profitable for developers to use it as a means of general interoperability.

If the resource is coming from any of the aforementioned parties, then domain experts may state that the resource has reached some degree of consensus.

5.3.2.5 Task 2.5 Evaluate the quality

The goal of this task is to evaluate the quality of the resource. We do not intend to provide a deep analysis of the quality of the resource but to offer some preliminary considerations about it. In this thesis, we propose to check the following quality attributes:

- Good documentation of the resource.
- Lack of anomalies of the non-ontological resource, such redundancies or inconsistencies.
- Reliability of the non-ontological resource. This means analysing whether we can trust in the resource or not.

5.3.2.6 Task 2.6 Build the assessment table

The goal of this task is to create an assessment table of the non-ontological resources. Software developers and ontology practitioners carry out this task, taking as input the non-ontological resources with their respective values for precision, coverage, consensus and quality criteria, for the construction of the assessment table. This table is shown in Table 11.3. The first column shows the non-ontological resources found. The precision column shows the precision value calculated for each non-ontological resource. Then, the coverage column shows the coverage value calculated for each non-ontological resource. Next, the consensus column depicts the domain experts' judgment about whether the non-ontological resource has been agreed on by the community or not (Yes/No). Finally, the quality column illustrates the domain experts, software developers and ontology practitioners' judgment about whether the resource has an acceptable level of quality or not (Yes/No).

5.3.3 Activity 3. Select the Most Appropriate Non-ontological Resources

The goal of this activity is to select the most appropriate non-ontological resources to be transformed into an ontology. This activity is carried out by domain experts,

Table 5.1: Assessment table for the NORs

NOR	Precision	Coverage	Consensus	Quality
NOR 1	NOR 1 Precision value	NOR 1 Coverage value	(Yes/No)	(Yes/No)
NOR 2	NOR 2 Precision value	NOR 2 Coverage value	(Yes/No)	(Yes/No)
NOR 3	NOR 3 Precision value	NOR 3 Coverage value	(Yes/No)	(Yes/No)

software developers and ontology practitioners, taking as input the non-ontological resource assessment table. The selection is performed manually and we recommend looking for resources with

- Consensus. This criterion is taken into account in the first place because, if the resource to be reused contains terminology agreed upon by the community, the effort and time spent in finding out the right labels for the ontology terms will decrease considerably.
- Quality. This criterion is taken into account in the second place because, if the resource to be reused has an acceptable level of quality, then the resultant ontology should also have it.
- High value of coverage. This criterion is taken into account in the third place because our third concern is to consider most of the ORSD terms identified.
- High value of precision. This criterion is taken into account in the fourth place because our fourth concern is the proportion of non-ontological lexical entries over the identified terms of the ORSD.

The activity output is a ranked list of non-ontological resources that, to some extent, covers the expected domain. These resources will be ready for the re-engineering process.

5.4 Summary

This chapter presents our solution to those aspects related to the reuse of non-ontological resources for building ontologies. It addresses some of the limitations identified in the state of art in this area.

First, it provides a formal definition of non-ontological resources and a categorization of them according to three dimensions: type of resource, data model, and implementation. Second, it introduces the Non-ontological Resource Metadata Vocabulary (NoRMV). NoRMV allows describing the non-ontological resources available, which can be used later on for generating provenance information in the ontology. Finally, it presents a method for reusing non-ontological resources for building ontologies. This method provides detailed guidelines for selecting the most suitable non-ontological resources for ontology development.

The solutions presented in this chapter cover contributions **C1**, **C2**, **C3** and **C4**, which address objective **O1** (see Chapter 3).

Chapter 6

PATTERN BASED RE-ENGINEERING METHOD

This chapter presents the method for re-engineering non-ontological resources into ontologies, which is based on a model for re-engineering non-ontological resources. First, it provides a description of this re-engineering model for NORs, and then it introduces the notion of patterns for re-engineering NORs. Next, it presents a discussion about the hidden semantics in the relations of the NORs and the formal definitions of the ontologies generated. Finally, it depicts the prescriptive methodological guidelines for re-engineering NORs into ontologies.

6.1 Re-engineering Model for Non-ontological Resources

This section describes our model for re-engineering non-ontological resources. The model is based on the software re-engineering model presented in Section 2.2.1. It is worth mentioning that we consider non-ontological resources as software resources because a software system consists of one or more programs, data files, databases, and job control scripts.

The model for non-ontological resource re-engineering is depicted in Figure 6.1. The figure also shows the following activities: NOR reverse engineering, NOR transformation, and ontology forward engineering. Next, we describe the activities defined in the Glossary of Activities in Ontology Engineering [SFGP08]:

- NOR reverse engineering is defined as the activity of analysing a non-ontological resource to identify its underlying components and creating 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.

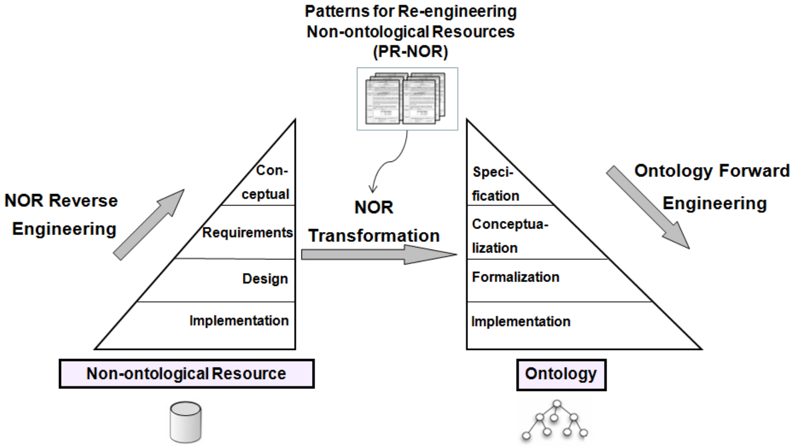


Figure 6.1: Re-engineering model for non-ontological resources

As mentioned before, we consider non-ontological resources as software resources and, therefore, we use the software abstraction levels shown in Figure 6.1 to depict the reverse engineering of the non-ontological resource. Understanding how a non-ontological resource is created is useful for also understanding how non-ontological resource can be reverse engineered. The idea of levels of abstraction that underlies the development process also underlies the reverse engineering process. This idea is used to model software development as a sequence of phases, in which each phase corresponds to a particular level of abstraction.

In the left triangle of Figure 6.1 we can distinguish the four different abstraction levels that define each activity in software development:

1. *The conceptual abstraction level*, which describes in general terms, the system functional characteristics;
2. *The requirements level*, which is the specification of the problem being solved;
3. *The design level*, which is the specification of the solution;
4. *The implementation level*, which refers to the coding, testing and delivery of the operational system.

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

In the right triangle of Figure 6.1 we can distinguish the four different abstraction levels that define each activity in ontology engineering:

1. *The specification level*, which describes the collection of requirements that the ontology should fulfil;
2. *The conceptualization level*, which information from the acquisition process is organized into meaningful conceptual models;

3. *The formalization level*, which represents the transformation of the conceptual model into a formal or semi-computable model according to a knowledge representation paradigm;
4. *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 6.1 suggests the path from the available non-ontolo-gical resource to the target ontology. This transformation is guided by a set of Patterns for Re-engineering Non-Ontological Resources (PR-NOR), and goes from the non-ontological resource requirements/design level to the conceptualization level of the ontology.

6.2 Requirements for the Transformation

In this section we describe the requirements identified for the transformation. The requirements are listed according to the three transformation approaches identified in Section 2.1.1 (see Figure 2.2).

- *TBox transformation* [CHPG09], which 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 also be possible on the ontology generated.
  - Conversion on the semantic level, which implies that the schema translation interprets the semantics of the data. In other words, the conversion should not avoid possible interpretations, e.g., relations among the NOR entities.

- *ABox transformation* [CHPG09], which 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 for the TBox transformation. Again, this implies that all queries that are possible on the original source should also be possible on the ontological version.
  - Structure preserving translation, which is the opposite of the second requirement of the TBox transformation. The translation should reflect as much as possible the original structure of the resource; in other words, the conversion should avoid possible interpretations.

- *Population*, which transforms the resource content into instances of an ontology. The requirements of the transformation are

- Full conversion, the same requirement for 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 to the already existing ontology schema.

6.3 Patterns for Re-engineering Non-ontological Resources

In this section we introduce the sixteen patterns, developed in this thesis, that perform the transformations of NORs into ontologies. Patterns for re-engineering NORs (PR-NOR) define a procedure that transforms the NOR terms into ontology representational primitives.

Next, we present the template proposed that describes the patterns for re-engineering non-ontological resources (PR-NOR). We have modified the tabular template used in [VTAGS$^+$08] for describing the PR-NORs. The meaning of each field is shown in Table 6.1.

According to the NOR categorization presented in section 5.1, in this thesis we propose patterns for re-engineering classification schemes, thesauri, and lexicons (see Table 6.2). Since 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 in order to extract the NORs terms and then to map these terms to a conceptual model of an ontology. This process is expressed as an algorithm. Moreover, it is worth mentioning that we refer to *ontology schema* as TBox, and just *ontology* as TBox and ABox. These patterns are included in the ODP Portal[1].

The re-engineering patterns take advantage of the use of the Ontology Design Patterns[2] for creating the ontology code. So, most of the code generated follows the best practices already identified by the community (*Process section* Table 6.1).

Although we have identified five types of NORs, here we just list patterns for re-engineering classification schemes, thesauri, and lexica (see Table 6.2).

The identifier of a PR-NOR follows a naming convention. Next, we illustrate the naming convention for identifying the patterns. We have the pattern identifier

```
PR-NOR-&&%%-##
```

where

- `PR-NOR` is the prefix
- `&&` represents the type of resource: `CL` for classification scheme, `TS` for thesaurus, and `LX` for lexicon.

[1] http://ontologydesignpatterns.org

[2] Ontology Design Patterns are included in the ODP portal. The ODP portal is a Semantic Web portal dedicated to ontology design best practices for the Semantic Web, emphasizing particulary ontology design patterns (OPs)

- `%%` represents the transformation approach: `TX` for TBox, `AX` for ABox.
- `##` represents a non-negative integer for numerating the patterns. It starts with 1 for TBox transformation and 10 for ABox transformation.

Table 6.1: Template of pattern for re-engineering non-ontological resource

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 non-ontological resources.
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 non-ontological resource.
Example	Description in natural language of an example of the non-ontological resource.
Graphical Representation	
General	Graphical representation of the non-ontological resource.
Example	Graphical representation of the example of non-ontological resource.
OUTPUT: Designed Ontology	
General	Description in natural language of the ontology created after applying the pattern for re-engineering the non-ontological resource.
Graphical Representation	
(UML) General Solution Ontology	Graphical representation, using the UML profile [BH06], of the ontology created for the non-ontological resource being re-engineered.
(UML) Example Solution Ontology	A graphical representation example, which uses the UML profile [BH06], of the ontology created for the non-ontological resource being used.
PROCESS: How to Re-engineer	
General	Algorithm for the re-engineering process.
Example	Application of the algorithm to the non-ontological resource example.
Time Complexity	The time complexity of the algorithm.
Additional Notes	Additional notes of the algorithm.
Formal Transformation	
General	Formal description of the transformation made with the formal definitions of the resources.
Relationships (Optional)	
Relations to other modelling components	Description of any relation to other PR-NOR patterns or other ontology design patterns.

Table 6.2: Set of patterns for re-engineering NORs

N	Identifier	Type of NOR	NOR Data Model	Target
1	PR-NOR-CLTX-01	Classification Scheme	Path Enumeration	Ontology Schema (TBox)
2	PR-NOR-CLTX-02	Classification Scheme	Adjacency List	Ontology Schema (TBox)
3	PR-NOR-CLTX-03	Classification Scheme	Snowflake	Ontology Schema (TBox)
4	PR-NOR-CLTX-04	Classification Scheme	Flattened	Ontology Schema (TBox)
5	PR-NOR-CLAX-10	Classification Scheme	Path Enumeration	Ontology (TBox+ABox)
6	PR-NOR-CLAX-11	Classification Scheme	Adjacency List	Ontology (TBox+ABox)
7	PR-NOR-CLAX-12	Classification Scheme	Snowflake	Ontology (TBox+ABox)
8	PR-NOR-CLAX-13	Classification Scheme	Flattened	Ontology (TBox+ABox)
9	PR-NOR-TSTX-01	Thesaurus	Record-based	Ontology Schema (TBox)
10	PR-NOR-TSTX-02	Thesaurus	Relation-based	Ontology Schema (TBox)
11	PR-NOR-TSAX-10	Thesaurus	Record-based	Ontology (TBox+ABox)
12	PR-NOR-TSAX-11	Thesaurus	Relation-based	Ontology (TBox+ABox)
13	PR-NOR-LXTX-01	Lexicon	Record-based	Ontology Schema (TBox)
14	PR-NOR-LXTX-02	Lexicon	Relation-based	Ontology Schema (TBox)
15	PR-NOR-LXAX-10	Lexicon	Record-based	Ontology (TBox+ABox)
16	PR-NOR-LXAX-11	Lexicon	Relation-based	Ontology (TBox+ABox)

6.4 Semantics of the Relations among the NOR Terms

The TBox transformation approach converts the resource content into an ontology schema. TBox transformation tries to impose a formal semantics on 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 thesis, we want to prove that we can rely on an external resource for making explicit the relations. For this purpose, first we rely on WordNet, and, then as a future line of this work, we may rely on other information resources, such as DBpedia[3].

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

Algorithm 1 Discovering the semantics of the relations - *getRelation*

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

The main parts of algorithm 1 are explained next

- (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

[3] http://www.dbpedia.org/

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) from the BT/NT relation, we recommend using the *subClassOf* relation by default, and (2) from the RT relation, we recommend using the *relatedTerm* relation by default.

- (Lines 3-6) Check if it is possible to get the *subClassOf* relation by identifying attribute adjetives[4] 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*
    - (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).

It is worth mentioning that the algorithm takes advantage of the use of the `PartOf content pattern`[5] for asserting the *partOf* relation.

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

6.5 Formal Definition of the Ontologies Generated

In this section we provide a formal definition of the ontologies generated which are dealt with by the patterns. This formal definition is used in the Formal Transformation section of the patterns (see Table 6.1).

Based on the definition provided in [ES07], we can define a lightweight ontology O as the following tuple:

$$O = \langle OS, KB \rangle$$

Where OS represents the ontology schema, and KB represents the knowledge base.

An ontology schema OS is defined through the following tuple:

$$OS = \langle C, A, R, S \rangle$$

[4] 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.

[5] `http://ontologydesignpatterns.org/wiki/Submissions:PartOf`

where:

- $C = \{c_1, ..., c_n\}$, a finite set of classes.
- $A = \{a_1, ..., a_n\}$, a finite set of attributes, where every $a_i \subseteq C$ x *Literal*.
- $R = \{r_1, ..., r_n\}$, a finite set of binary relations, where every $r_i \subseteq C$ x C.
- $S : C \to C$, a *subClassOf* relation.

A knowledge base is a structure:

$$KB = \langle C, A, R, I, t_C, t_A, t_R \rangle$$

consisting of:

- three sets C, A, and R as defined before.
- a set $I = \{i_1, ..., i_n\}$ whose elements are called instance identifiers
- a function $t_C : C \to I$ called class instantiation
- a function $t_A : A \to I$ called attribute instantiation
- a function $t_R : R \to I^2$ called relation instantiation

6.6 Method for Re-engineering Non-ontological Resources

In this section we depict the prescriptive methodological guidelines for re-engineering NORs. The goal of the Method for Re-engineering Non-Ontological Resources is to transform a non-ontological resource into an ontology. The output of the process is an ontology. Figure 6.2 shows the filling card of the non-ontological resource re-engineering process, which includes the definition, goal, input, output, performer of the process and time execution.

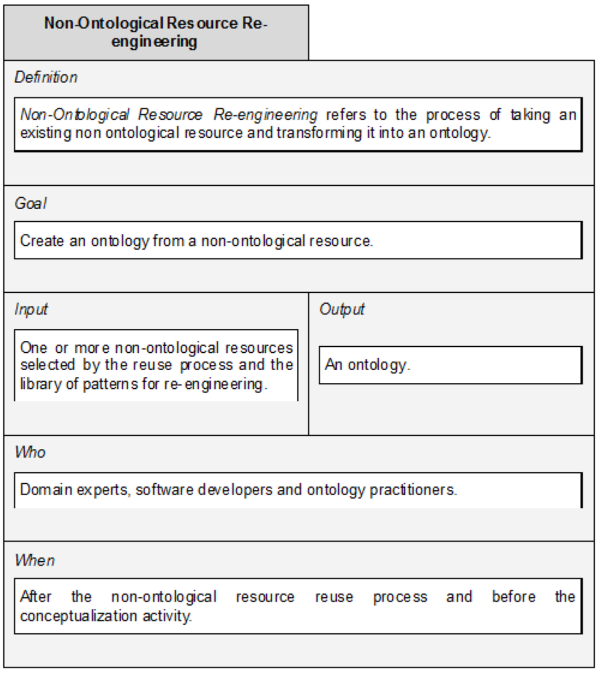
The NOR re-engineering process consists of the three activities depicted in Figure 6.3.

6.6.1 Activity 1. Non-ontological Resource Reverse Engineering.

The goal of this activity is to analyse a non-ontological resource, to identify its underlying terms, and to create representations of the resource at the different levels of abstraction (design, requirements and conceptual).

6.6.1.1 Task 1.1 Data gathering.

The goal of this task is to search and compile all the available data and documentation about the non-ontological resource, including purpose, components, data model and implementation details.



Non-Ontological Resource Re-engineering

Definition

Non-Ontological Resource Re-engineering refers to the process of taking an existing non ontological resource and transforming it into an ontology.

Goal

Create an ontology from a non-ontological resource.

Input

One or more non-ontological resources selected by the reuse process and the library of patterns for re-engineering.

Output

An ontology.

Who

Domain experts, software developers and ontology practitioners.

When

After the non-ontological resource reuse process and before the conceptualization activity.

Figure 6.2: Non-ontological resource re-engineering filling card

6.6.1.2 Task 1.2 Conceptual abstraction.

The goal of this task is to identify the schema of the non-ontological resource 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.

6.6.1.3 Task 1.3 Information exploration.

The goal of this task is to find out how the conceptual schema of the non-ontological resource and its content are represented in the data model. If the non-ontological resource data model is not available in the documentation, the data model should be reconstructed manually or with a data modelling tool.

6.6.2 Activity 2. Non-ontological Resource Transformation.

This activity has as a goal to generate a conceptual model from the non-ontological resource. We propose the use of Patterns for Re-engineering Non-Ontological Resources (PR-NOR) to guide the transformation process.

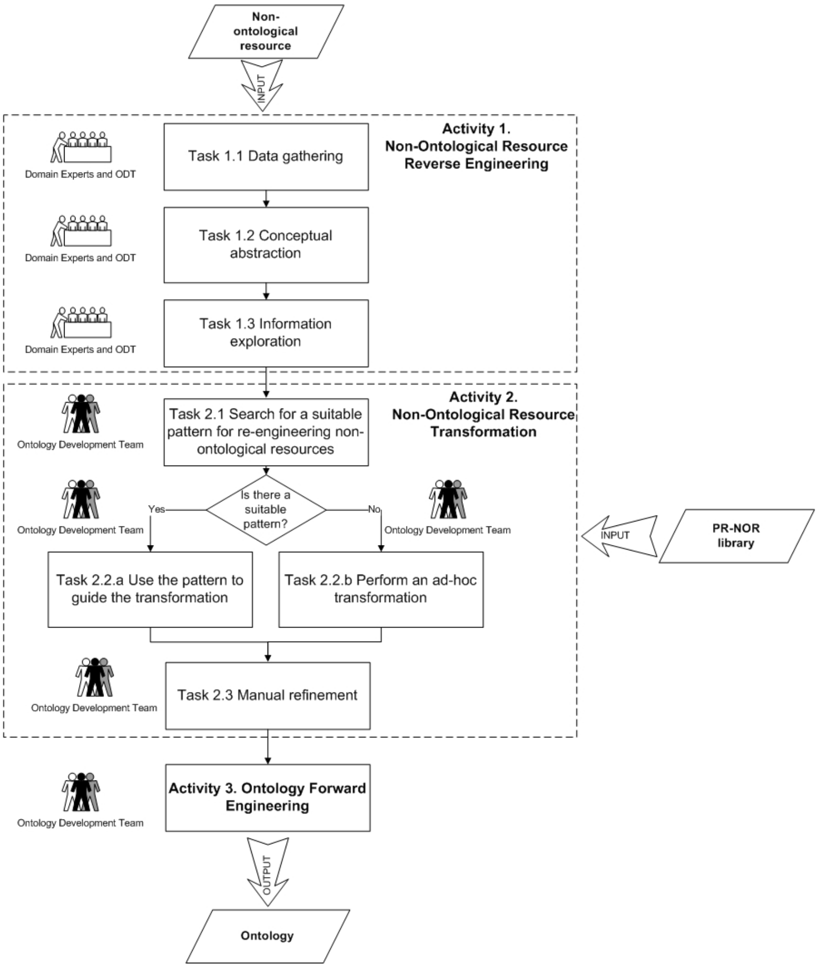

Non-ontological resource
INPUT
Activity 1. Non-Ontological Resource Reverse Engineering
Domain Experts and ODT
Task 1.1 Data gathering
Domain Experts and ODT
Task 1.2 Conceptual abstraction
Domain Experts and ODT
Task 1.3 Information exploration
Activity 2. Non-Ontological Resource Transformation
Ontology Development Team
Task 2.1 Search for a suitable pattern for re-engineering non-ontological resources
Ontology Development Team
Yes
Is there a suitable pattern?
No
Ontology Development Team
INPUT
PR-NOR library
Task 2.2.a Use the pattern to guide the transformation
Task 2.2.b Perform an ad-hoc transformation
Ontology Development Team
Task 2.3 Manual refinement
Ontology Development Team
Activity 3. Ontology Forward Engineering
OUTPUT
Ontology

Figure 6.3: Re-engineering process for non-ontological resources

6.6.2.1 Task 2.1 Search for a suitable pattern for re-engineering non-ontological resource.

The goal of this task is to find out if there is any applicable re-engineering pattern that transforms the non-ontological resource into a conceptual model. The search is performed in the ODP Portal[6], which includes the PR-NOR library, and with the following criteria: (1) non-ontological resource type, (2) internal data model of the resource, and (3) the transformation approach.

6.6.2.2 Task 2.2.a Use re-engineering patterns to guide the transformation.

The goal of this task is to apply the re-engineering pattern obtained in task 2.1 to transform the non-ontological resource into a conceptual model. If a suitable

[6] http://ontologydesignpatterns.org

pattern for re-engineering non-ontological resource is found, then the conceptual model is created from the non-ontological resource following the procedure established in the pattern for re-engineering. Alternatively, the software library, described in Chapter 10, can be used for generating the ontology automatically.

6.6.2.3 Task 2.2.b Perform an ad-hoc transformation.

The goal of this task is to set up an *ad-hoc* procedure that transforms the non-ontological resource into a conceptual model when a suitable pattern for re-engineering cannot be found. This *ad-hoc* procedure may be generalized to create a new pattern for re-engineering non-ontological resource.

6.6.2.4 Task 2.3 Manual refinement.

The goal of this task is to check whether any inconsistency appears after the transformation. Software developers and ontology practitioners, with the help of domain experts, can fix manually any inconsistencies appearing after the transformation.

6.6.3 Activity 3. Ontology Forward Engineering

The goal of this activity 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. The conceptual model obtained in task 2.2.a or 2.2.b is transformed into a formalized model, according to a knowledge representation paradigm such as description logics and first order logic. Then, the formalized model is implemented in an ontology language.

6.7 Summary

This chapter has presented our solution for the aspects related to the re-engineering of non-ontological resources for building ontologies. It also addresses some of the limitations identified in the state of art in this area.

First, it presents our Re-engineering Model for Non-ontological resources, which is based on the software re-engineering model presented in Section 2.2.1. Then, it describes the requirements for the transformation process. Next, it briefly describes the Patterns for Re-engineering Non-ontological resources into Ontologies (PR-NOR). Then, it shows how the patterns make explicit the hidden semantics in the relations of the non-ontological resource. After that, it introduces the formal definitions of the ontologies generated. Finally, it presents our method for re-engineering non-ontological resources for building ontologies. This method provides detailed guidelines for transforming a non-ontological resource into an ontology.

The solutions presented in this chapter cover contributions **C5** and **C6**, which address objective **O2** (see Chapter 3).

Chapter 7

PATTERNS FOR RE-ENGINEERING CLASSIFICATION SCHEMES

Classification schemes [KBH$^+$97] play an important role when retrieving information in a network environment, especially because they provide browsing structures for subject-based information gateways on the Web. The advantages of using classification schemes include improved subject browsing facilities and interoperability with other services. Classification schemes are probably the most valuable input for creating, at a reasonable cost, ontologies in many domains. They contain, readily available, a wealth of category definitions plus a hierarchy and reflect some degree of community consensus [HdB07].

In this chapter we present a definition of classification schemes, the data models for representing classification schemes and our main contribution: the set of patterns for re-engineering classification schemes into ontologies.

7.1 Classification Scheme

A classification scheme [ISO04] is the descriptive information of an arrangement or division of objects into groups based on the characteristics the objects have in common. A good example is the Fishery International Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP)[1] from FAO[2].

7.1.1 Components of a Classification Scheme

The ISO/IEC 11179-2 [ISO04] provides a conceptual model for managing classification schemes and identifies the classification scheme components, presented in Figure 7.1.

[1] `http://www.fao.org/figis/servlet/RefServlet`

[2] `http://www.fao.org`

- A *classification scheme*, which represents the classification scheme itself. It has the *cs_name* element, that is, the name of the classification scheme.
- A *classification scheme item*, which represents the individual item within a classification scheme. It has the following elements:
  - *csi_name*, which is the name of the classification scheme item.
  - One or more *csi_attributes*
- A *classification scheme item relationship*. It is the relationship among items within a classification scheme. Such relation serves to assist in navigating through a large number of classification scheme items. This relationship bears the *csir_name* element, which is the name of the classification scheme item relationship.

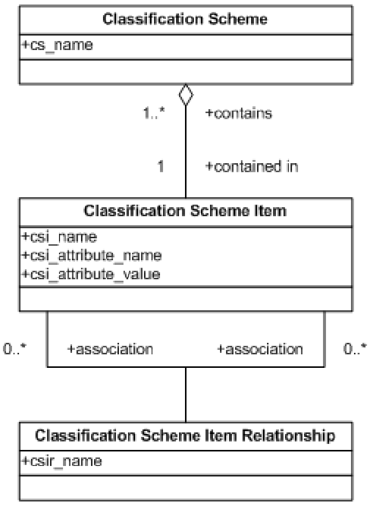


Figure 7.1: Main components of the UML representation of the classification scheme [ISO04]

7.1.2 Classification Scheme Formal Definition

We formally define a classification scheme as the following tuple:

$$C = \langle CS, CC \rangle$$

Where CS represents the schema of the classification scheme, and CC represents the content of the classification scheme.

The schema of the classification scheme, CS, is defined as:

$$CS = \langle CG, CA, CR \rangle$$

where:

- $CG = \{c_1\}$, a set of one category.

- $CA = \{a_1, ..., a_n\}$, a finite set of attributes, where every $a_i \subseteq CG$ x *Literal*.
- $CR = \{r_1\}$, a set of one binary relation, where $r_1 \subseteq CG$ x CG.

The content of the classification scheme, CC, is defined as

$$CC = \langle CG, CA, CR, CI, Ct_C, Ct_A, Ct_R \rangle$$

which consists of

- The three CG, CA and CR sets, as were defined before.
- A $CI = \{csi_1, ..., csi_n\}$ set, whose elements are called classification scheme item identifiers
- A $Ct_G : CG \to CI$ function, called classification scheme item instantiation
- A $Ct_A : CA \to CI$ function, called classification scheme attribute instantiation
- A $Ct_R : CR \to CI^2$ function, called classification scheme relation instantiation

7.1.3 Classification Scheme Data Models

As we mentioned in Section 5.1 there are different ways of representing the knowledge encoded by a particular resource. In this section we review the existing data models for classification schemes already presented in Section 5.1. In order to exemplify the data models for classification schemes, we use an excerpt from the FAO classification scheme of water areas[3] shown in Figure 5.2-a). These data models are the following:

- A path enumeration data model [Bra05] 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, see Figure 5.2-b).
- An adjacency list [Bra05] data model is a recursive structure for hierarchy representations comprising a list of nodes with a linking column to their parent nodes. In this case, every classification scheme item has the parent code, see Figure 5.2-c).
- A snowflake data model [MZ06] is a normalized structure for hierarchy representations. In this case, the classification scheme items are grouped by levels or entities. There are as many groups as levels the classification scheme has. In this model every classification scheme item has the parent code (i.e., parent key value), just like the adjacency list data model has; see Figure 5.2-d).

[3] `http://www.fao.org/figis/servlet/RefServlet`

- A flattened data model [MZ06] is a denormalized structure for hierarchy representations. In this case, each hierarchy level is represented on a different column. There are as many columns as levels the classification scheme has. The hierarchy is represented with one single entity where each hierarchy level is stored in a different column, see Figure 5.2-e).

7.1.4 Classification Scheme Implementations

These data models can be implemented as databases, XML files, flat files, spreadsheets, etc. Figure 5.2-f) presents an XML implementation of the adjacency list model of the water area classification, and Figure 5.2-g) presents a spreadsheet implementation of the path enumeration model of the same classification scheme.

Figure 7.2 shows a classification scheme modelled following a path enumeration model. In this case, the classification scheme is implemented in a database and in an XML file. Figure 7.2 depicts the resource in our three level categorization of NORs.

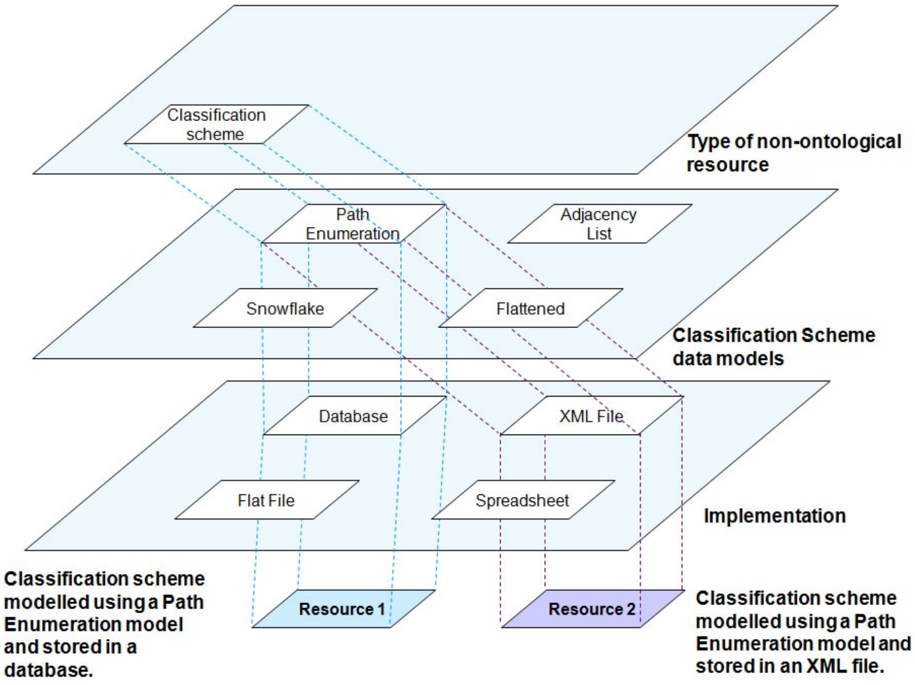

Classification scheme
Type of non-ontological resource
Path Enumeration
Adjacency List
Snowflake
Flattened
Classification Scheme data models
Database
XML File
Flat File
Spreadsheet
Implementation
Classification scheme modelled using a Path Enumeration model and stored in a database.
Resource 1
Resource 2
Classification scheme modelled using a Path Enumeration model and stored in an XML file.

Figure 7.2: Classification scheme categorization

7.2 Patterns for Re-engineering Classification Schemes into Ontologies

This section presents re-engineering patterns (PR-NOR) for re-engineering classification schemes into ontologies. The patterns follow the naming convention defined in Section 6.3. The patterns are

- Patterns for the TBox transformation
  - PR-NOR-CLTX-01. The pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema.
  - PR-NOR-CLTX-02. The pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema.
  - PR-NOR-CLTX-03. The pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema.
  - PR-NOR-CLTX-04. The pattern for re-engineering a classification scheme following the flattened data model, into an ontology schema.
- Patterns for the ABox transformation
  - PR-NOR-CLAX-10. The pattern for re-engineering a classification scheme following the path enumeration data model into an ontology.
  - PR-NOR-CLAX-11. The pattern for re-engineering a classification scheme following the adjacency list data model into an ontology.
  - PR-NOR-CLAX-12. The pattern for re-engineering a classification scheme following the snowflake data model into an ontology.
  - PR-NOR-CLAX-13. The pattern for re-engineering a classification scheme following the flattened data model into an ontology.

7.2.1 Patterns for the TBox Transformation

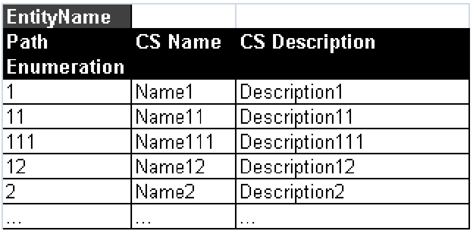
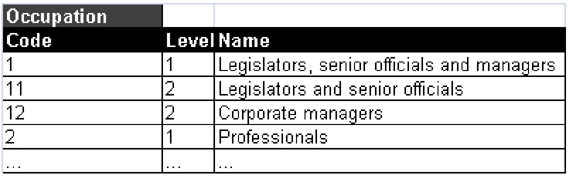
These patterns transform the resource content into an ontology schema. The TBox transformation approach tries to impose a formal semantics to the re-engineered resources, even at the cost of changing their structure [SAd$^+$07]. For making explicit the semantics of the relations among the NOR terms, the patterns rely on an external resource, WordNet, as we described in Section 6.4.

The time complexity of the algorithms described in the section *PROCESS: How to Re-engineering* is polynomial $O(n^2)$.

7.2.1.1 Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema

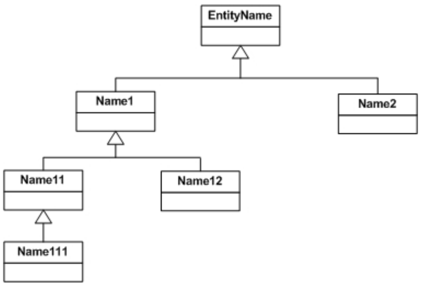
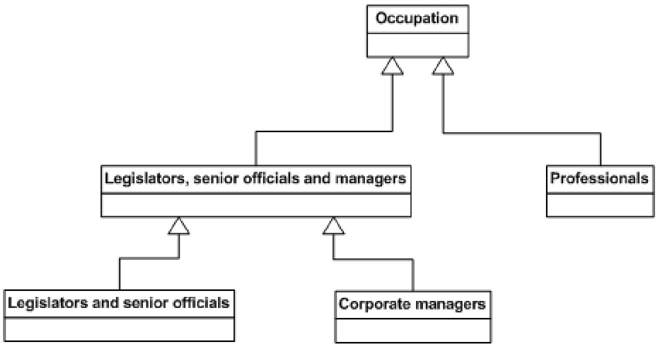
The pattern for re-engineering non-ontological resource, shown in Table 7.1, provides a guide to transform a classification scheme into an ontology schema. The classification scheme is modelled with a path enumeration data model.

Table 7.1: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema.

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td colspan="2"><b>General Information</b></td></tr>
<tr><td><b>Name</b></td><td>Pattern for Re-engineering a Classification Scheme following the Path Enumeration data model into an Ontology Schema.</td></tr>
<tr><td><b>Identifier</b></td><td>PR-NOR-CLTX-01</td></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resource (PR-NOR)</td></tr>
<tr><td colspan="2"><b>Use Case</b></td></tr>
<tr><td><b>General</b></td><td>Re-engineering a classification scheme following the path enumeration model, to design an ontology schema.</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology based on the International Standard Classification of Occupations (for European Union purposes) ISCO-88 (COM). This classification scheme follows the path enumeration data model.</td></tr>
<tr><td colspan="2"><b>Pattern for Re-engineering Non-ontological Resource</b></td></tr>
<tr><td colspan="2"><b>INPUT: Resource to be Re-engineered</b></td></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a classification scheme that follows the path enumeration model.
A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children terms may vary depending on the context.
The path enumeration data model [Bra05], for classification schemes, takes advantage of the fact that there is one and only one path from the root to every item in the classification. The path enumeration model stores that path as a string by concatenating either the edges or the keys of the classification scheme items in the path.</td></tr>
<tr><td><b>Example</b></td><td>The International Standard Classification of Occupations (for European Union purposes), 1988 version: ISCO-88 (COM) published by Eurostat is modelled with the path enumeration data model. This classification scheme is available at <code>http://ec.europa.eu/eurostat/ramon/</code></td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>EntityName</th><th></th><th></th></tr>
<tr><th>Path Enumeration</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>Name1</td><td>Description1</td></tr>
<tr><td>11</td><td>Name11</td><td>Description11</td></tr>
<tr><td>111</td><td>Name111</td><td>Description111</td></tr>
<tr><td>12</td><td>Name12</td><td>Description12</td></tr>
<tr><td>2</td><td>Name2</td><td>Description2</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th>Occupation</th><th></th><th></th></tr>
<tr><th>Code</th><th>Level</th><th>Name</th></tr>
<tr><td>1</td><td>1</td><td>Legislators, senior officials and managers</td></tr>
<tr><td>11</td><td>2</td><td>Legislators and senior officials</td></tr>
<tr><td>12</td><td>2</td><td>Corporate managers</td></tr>
<tr><td>2</td><td>1</td><td>Professionals</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
</table>

Continued on next page

Table 7.1: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema (continued).

Slot	Value
OUTPUT: Designed Ontology	
General	The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. Each term in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent terms are made explicit by means of an external resource.
Graphical Representation	
(UML) General Solution Ontology	
(UML) Example Solution Ontology	 Occupation Legislators, senior officials and managers Professionals Legislators and senior officials Corporate managers

Continued on next page

Table 7.1: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema (continued).

Slot	**Value**
PROCESS: How to Re-engineer	

General	**Require:** Identification of the parent/child by using the *path enumeration* model
	$noParentTerms \leftarrow$ classification scheme terms without parent
	if $noParentTerms.length > 1$ **then**
	  $entityName \leftarrow$ name of the entity that contains the classification scheme terms
	  $rootClass \leftarrow$ createClass($entityName$)
	  for $ri \in noParentTerms$ **do**
	    $Ri \leftarrow$ createClass(ri)
	    $relation \leftarrow$ ExternalResource.getRelation($rootClass$,Ri)
	    relate($relation$,$rootClass$,Ri)
	  end for
	end if
	repeat
	  for $cei \in noParentTerms$ **do**
	    if not alreadyCreatedClassFor(cei) **then**
	      $Ci \leftarrow$ createClass(cei)
	    end if
	    $children \leftarrow$ childrenOf(cei)
	    for $cej \in children$ **do**
	      if not alreadyCreatedClassFor(cej) **then**
	        $Cj \leftarrow$ createClass(cej)
	      end if
	      $relation \leftarrow$ ExternalResource.getRelation(cei,cej)
	      relate($relation$,cei,cej)
	    end for
	    add($allChildren$,$children$)
	  end for
	  $noParentTerms \leftarrow allChildren$
	  removeAllTerms($allchildren$)
	until isEmpty($noParentTerms$)

Continued on next page

Table 7.1: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema (continued).

Slot	Value
Example	**Require:** Identification of the parent/child by using the *path enumeration* model 1: $noParentTerms$ ← [Legislators, senior officials and managers;Professionals] 2: // $noParentTerms.length$=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: $children$ ← $\emptyset$ ← childrenOf(Professionals) 26: $noParentTerms$ ← [Legislators and senior officials;Corporate managers] 27: removeAllTerms($allChildren$) 13: // Legislators and senior officials, $C1$, already created 16: $children$ ← $\emptyset$ ← childrenOf(Legislators and senior officials) 13: // Corporate managers, $C2$, already created 16: $children$ ← $\emptyset$ ← childrenOf(Corporate managers) 24: $allChildren$ ← $\emptyset$ 26: $noParentTerms$ ← $allChildren$ ← $\emptyset$
Time Complexity	$O(n^2)$
Additional Notes	• $noParentTerms$, $children$, $allChildren$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *childrenOf* is a function that returns the children of a given term. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list.
Formal Transformation	
General	Classification Scheme: $C = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CC \longrightarrow OS$: $Ct_G \longrightarrow C$ $Ct_R \longrightarrow R \cup S$

Continued on next page

Table 7.1: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema (continued).

Slot	Value
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-TX-01 [SFBG+07]

7.2.1.2 Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema

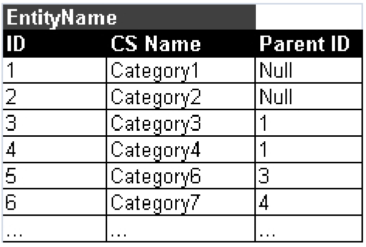
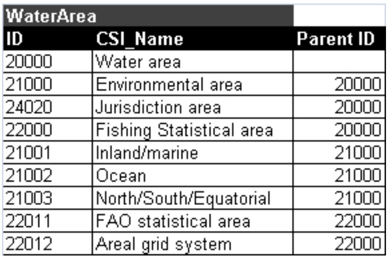
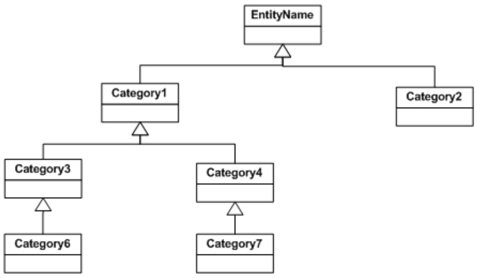
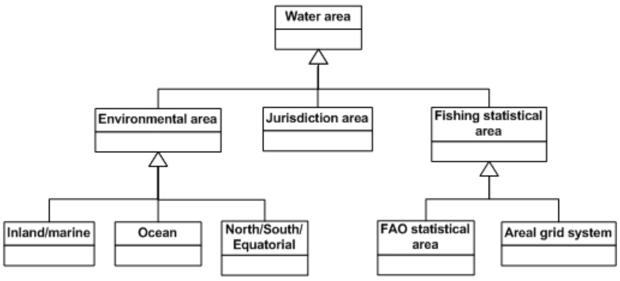
The pattern for re-engineering non-ontological resource, shown in Table 7.2, provides a guide to transform a classification scheme into an ontology schema. The classification scheme is modelled with an adjacency list data model.

Table 7.2: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Adjacency List data model into an ontology Schema
Identifier	PR-NOR-CLTX-02
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the adjacency list model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on the water areas classification published by FAO. This classification scheme follows the adjacency list data model.
Pattern for Re-engineering Non-ontological Resource	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a classification scheme that follows the adjacency list model. A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parent and children terms may vary depending on the context. The adjacency list data model [Bra05] for hierarchical classifications proposes to create an entity which holds a list of items with a linking column associated to their parent items.
Example	The FAO classification for water areas groups them according to some different criteria, such as environment, statistics, and jurisdiction, among others. This classification scheme is available at `http://www.fao.org/figis/servlet/RefServlet`

Continued on next page

Table 7.2: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="3">EntityName</th></tr>
<tr><th>ID</th><th>CS Name</th><th>Parent ID</th></tr>
<tr><td>1</td><td>Category1</td><td>Null</td></tr>
<tr><td>2</td><td>Category2</td><td>Null</td></tr>
<tr><td>3</td><td>Category3</td><td>1</td></tr>
<tr><td>4</td><td>Category4</td><td>1</td></tr>
<tr><td>5</td><td>Category6</td><td>3</td></tr>
<tr><td>6</td><td>Category7</td><td>4</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="3">WaterArea</th></tr>
<tr><th>ID</th><th>CSI_Name</th><th>Parent ID</th></tr>
<tr><td>20000</td><td>Water area</td><td></td></tr>
<tr><td>21000</td><td>Environmental area</td><td>20000</td></tr>
<tr><td>24020</td><td>Jurisdiction area</td><td>20000</td></tr>
<tr><td>22000</td><td>Fishing Statistical area</td><td>20000</td></tr>
<tr><td>21001</td><td>Inland/marine</td><td>21000</td></tr>
<tr><td>21002</td><td>Ocean</td><td>21000</td></tr>
<tr><td>21003</td><td>North/South/Equatorial</td><td>21000</td></tr>
<tr><td>22011</td><td>FAO statistical area</td><td>22000</td></tr>
<tr><td>22012</td><td>Areal grid system</td><td>22000</td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. Each term in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent terms are made explicit by using an external resource.</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
</td></tr>
<tr><td><b>(UML) Example Solution Ontology</b></td><td>
</td></tr>
</table>

Continued on next page

Table 7.2: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema (continued).

Slot	Value
PROCESS: How to Re-engineer	

General	**Require:** Identification of the parent/child by using the *adjacency list* model
	$noParentTerms \leftarrow$ classification scheme terms without parent
	if $noParentTerms.length > 1$ **then**
	  $entityName \leftarrow$ name of the entity that contains the classification scheme terms
	  $rootClass \leftarrow$ createClass($entityName$)
	  for $ri \in noParentTerms$ **do**
	    $Ri \leftarrow$ createClass(ri)
	    $relation \leftarrow$ ExternalResource.getRelation($rootClass$,Ri)
	    relate($relation$,$rootClass$,Ri)
	  end for
	end if
	repeat
	  for $cei \in noParentTerms$ **do**
	    if not alreadyCreatedClassFor(cei) **then**
	      $Ci \leftarrow$ createClass(cei)
	    end if
	    $children \leftarrow$ childrenOf(cei)
	    for $cej \in children$ **do**
	      if not alreadyCreatedClassFor(cej) **then**
	        $Cj \leftarrow$ createClass(cej)
	      end if
	      $relation \leftarrow$ ExternalResource.getRelation(cei,cej)
	      relate($relation$,cei,cej)
	    end for
	    add($allChildren$,$children$)
	  end for
	  $noParentTerms \leftarrow allChildren$
	  removeAllTerms($allchildren$)
	until isEmpty($noParentTerms$)

Continued on next page

Table 7.2: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema (continued).

Slot	Value

Example	**Require:** Identification of the parent/child by using the *adjacency list* model
	1: $noParentTerms \leftarrow$ [Water area]
	2: // $noParentTerms.length$=1 > 1
	14: $C1 \leftarrow$ createClass(Water area)
	16: $children \leftarrow$ childrenOf(Water area) // using the *adjacency list* model
	16: $children \leftarrow$ [Environmental area; Jurisdiction area; Fishing statistical area]
	19: $C2 \leftarrow$ createClass(Environmental area)
	21: $rel1 \leftarrow$ ExternalResource.getRelation($C1$,$C2$)
	19: $C3 \leftarrow$ createClass(Jurisdiction area)
	21: $rel2 \leftarrow$ ExternalResource.getRelation($C1$,$C3$)
	19: $C4 \leftarrow$ createClass(Fishing statistical area)
	21: $rel3 \leftarrow$ ExternalResource.getRelation($C1$,$C4$)
	26: $noParentTerms \leftarrow$ [Environmental area; Jurisdiction area; Fishing statistical area]
	16: $children \leftarrow$ childrenOf(Environmental area) // using the *adjacency list* model
	16: $children \leftarrow$ [Inland/Marine;Ocean;North/South/Equatorial]
	19: $C5 \leftarrow$ createClass(Inland/Marine)
	21: $rel4 \leftarrow$ ExternalResource.getRelation($C2$,$C5$)
	19: $C6 \leftarrow$ createClass(Ocean)
	21: $rel5 \leftarrow$ ExternalResource.getRelation($C2$,$C6$)
	19: $C7 \leftarrow$ createClass(North/South/Equatorial)
	21: $rel6 \leftarrow$ ExternalResource.getRelation($C2$,$C7$)
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Jurisdiction area)
	16: $children \leftarrow$ childrenOf(Fishing statistical area) // using the *adjacency list* model
	16: $children \leftarrow$ [FAO Statistical area;Areal grid system]
	19: $C8 \leftarrow$ createClass(FAO Statistical area)
	21: $rel7 \leftarrow$ ExternalResource.getRelation($C4$,$C8$)
	19: $C9 \leftarrow$ createClass(Areal grid system)
	21: $rel8 \leftarrow$ ExternalResource.getRelation($C4$,$C9$)
	26: $noParentTerms \leftarrow$ [Inland/Marine;Ocean;North/South/Equ.;FAO Statistical;Areal grid system]
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Inland/Marine)
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Ocean)
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(North/South/Equatorial)
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(FAO Statistical area)
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Areal grid system)
	26: $noParentTerms \leftarrow \emptyset$

Time Complexity	$O(n^2)$
Additional Notes	• $noParentTerms$, $children$, $allChildren$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *childrenOf* is a function that returns the children of a given term. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list.

Continued on next page

Table 7.2: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $C = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CC \longrightarrow OS$: $Ct_G \longrightarrow C$ $Ct_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-TX-01 [SFBG+07]

7.2.1.3 Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema

The pattern for re-engineering non-ontological resource, shown in Table 7.3, provides a guide to transform a classification scheme into an ontology schema. The classification scheme is modelled with a snowflake data model.

Table 7.3: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Snowflake data model into an ontology Schema
Identifier	PR-NOR-CLTX-03
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the snowflake model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on an occupation hierarchical classification following the snowflake data model.
Pattern for Re-engineering Non-ontological Resource	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a classification scheme that follows the snowflake model. A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parent and children terms may vary depending on the context. The snowflake data model [MZ06] is a normalized structure for hierarchy representations. In this case, the classification scheme items are grouped by levels or entities. There are as many groups as levels the classification scheme has.

Continued on next page

Table 7.3: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema (continued).

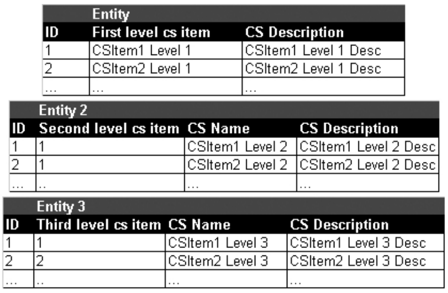
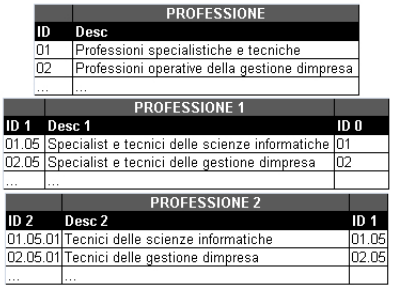
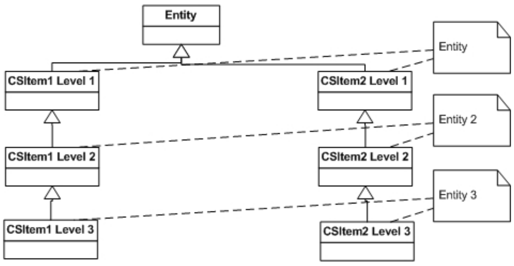
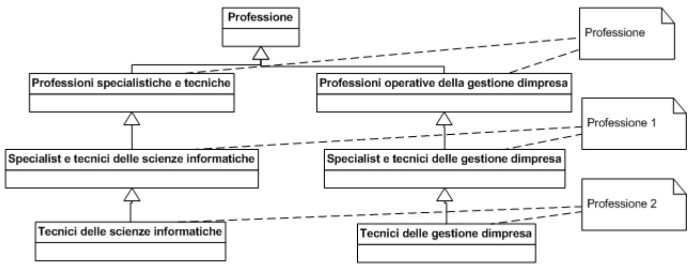
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>Snowflake models are widely used on data warehouses to build hierarchical classifications on structures known as dimensions. Some examples of dimension are Time, Product Category, Geography, Occupations, etc.
In this pattern the example is an occupation hierarchical classification hold on four different tables, one for each level (PROFESSIONI_0, PROFESSIONI_1, PROFESSIONI_2, PROFESSIONI_3).</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th></th><th colspan="2">Entity</th></tr>
<tr><th>ID</th><th>First level cs item</th><th>CS Description</th></tr>
<tr><td>1</td><td>CSItem1 Level 1</td><td>CSItem1 Level 1 Desc</td></tr>
<tr><td>2</td><td>CSItem2 Level 1</td><td>CSItem2 Level 1 Desc</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th></th><th colspan="3">Entity 2</th></tr>
<tr><th>ID</th><th>Second level cs item</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>1</td><td>CSItem1 Level 2</td><td>CSItem1 Level 2 Desc</td></tr>
<tr><td>2</td><td>1</td><td>CSItem2 Level 2</td><td>CSItem2 Level 2 Desc</td></tr>
<tr><td>...</td><td>..</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th></th><th colspan="3">Entity 3</th></tr>
<tr><th>ID</th><th>Third level cs item</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>1</td><td>CSItem1 Level 3</td><td>CSItem1 Level 3 Desc</td></tr>
<tr><td>2</td><td>2</td><td>CSItem2 Level 3</td><td>CSItem2 Level 3 Desc</td></tr>
<tr><td>...</td><td>..</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th></th><th>PROFESSIONE</th></tr>
<tr><th>ID</th><th>Desc</th></tr>
<tr><td>01</td><td>Professioni specialistiche e tecniche</td></tr>
<tr><td>02</td><td>Professioni operative della gestione dimpresa</td></tr>
<tr><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th></th><th>PROFESSIONE 1</th><th></th></tr>
<tr><th>ID 1</th><th>Desc 1</th><th>ID 0</th></tr>
<tr><td>01.05</td><td>Specialist e tecnici delle scienze informatiche</td><td>01</td></tr>
<tr><td>02.05</td><td>Specialist e tecnici delle gestione dimpresa</td><td>02</td></tr>
<tr><td>...</td><td>...</td><td></td></tr>
</table>
<table>
<tr><th></th><th>PROFESSIONE 2</th><th></th></tr>
<tr><th>ID 2</th><th>Desc 2</th><th>ID 1</th></tr>
<tr><td>01.05.01</td><td>Tecnici delle scienze informatiche</td><td>01.05</td></tr>
<tr><td>02.05.01</td><td>Tecnici delle gestione dimpresa</td><td>02.05</td></tr>
<tr><td>...</td><td>...</td><td></td></tr>
</table></td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. Each term in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent terms are made explicit by using an external resource.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 7.3: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema (continued).

Slot	Value
(UML) Example Solution Ontology	 Professione Professioni specialistiche e tecniche Professioni operative della gestione dimpresa Specialist e tecnici delle scienze informatiche Specialist e tecnici delle gestione dimpresa Tecnici delle scienze informatiche Tecnici delle gestione dimpresa Professione Professione 1 Professione 2
PROCESS: How to Re-engineer	

General	**Require:** Identification of the parent/child by using the *snowflake* model
	$noParentTerms$ ← classification scheme terms without parent
	if $noParentTerms.length > 1$ **then**
	  $entityName$ ← name of the entity that contains the classification scheme terms
	  $rootClass$ ← createClass($entityName$)
	  for $ri \in noParentTerms$ **do**
	    Ri ← createClass(ri)
	    $relation$ ← ExternalResource.getRelation($rootClass$,Ri)
	    relate($relation$,$rootClass$,Ri)
	  end for
	end if
	repeat
	  for $cei \in noParentTerms$ **do**
	    if not alreadyCreatedClassFor(cei) **then**
	      Ci ← createClass(cei)
	    end if
	    $children$ ← childrenOf(cei)
	    for $cej \in children$ **do**
	      if not alreadyCreatedClassFor(cej) **then**
	        Cj ← createClass(cej)
	      end if
	      $relation$ ← ExternalResource.getRelation(cei,cej)
	      relate($relation$,cei,cej)
	    end for
	    add($allChildren$,$children$)
	  end for
	  $noParentTerms$ ← $allChildren$
	  removeAllTerms($allchildren$)
	until isEmpty($noParentTerms$)

Continued on next page

Table 7.3: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema (continued).

Slot	Value

Example	**Require:** Identification of the parent/child by using the *snowflake* model
	1: $noParentTerms \leftarrow$ [Professioni specialistiche e tecniche;Professioni operative della gestione dimpresa]
	2: // $noParentTerms.length$=2 $>$ 1
	3: $entityName \leftarrow$ Professione
	4: $rootClass \leftarrow$ createClass($entityName$)
	6: $R1 \leftarrow$ createClass(Professioni specialistiche e tecniche)
	7: $relation1 \leftarrow$ ExternalResource.getRelation($rootClass$,$R1$)
	6: $R2 \leftarrow$ createClass(Professioni operative della gestione dimpresa)
	7: $relation2 \leftarrow$ ExternalResource.getRelation($rootClass$,$R2$)
	16: $children \leftarrow$ childrenOf(Professioni specialistiche e tecniche) // using the *snowflake* model
	16: $children \leftarrow$ [Specialist e tecnici delle scienze informatiche]
	19: $C1 \leftarrow$ createClass(Specialist e tecnici delle scienze informatiche)
	21: $rel1 \leftarrow$ ExternalResource.getRelation($R1$,$C1$)
	16: $children \leftarrow$ childrenOf(Professioni operative della gestione dimpresa) // using the *snowflake* model
	16: $children \leftarrow$ [Specialist e tecnici delle gestione dimpresa]
	19: $C2 \leftarrow$ createClass(Specialist e tecnici delle gestione dimpresa)
	21: $rel2 \leftarrow$ ExternalResource.getRelation($R2$,$C2$)
	26: $noParentTerms \leftarrow$ [Specialist e tecnici delle scienze informatiche;Specialist e tecnici delle gestione dimpresa]
	16: $children \leftarrow$ childrenOf(Specialist e tecnici delle scienze informatiche) // using the *snowflake* model
	16: $children \leftarrow$ [Tecnici delle scienze informatiche]
	19: $C3 \leftarrow$ createClass(Tecnici delle scienze informatiche)
	21: $rel3 \leftarrow$ ExternalResource.getRelation($C1$,$C3$)
	16: $children \leftarrow$ childrenOf(Specialist e tecnici delle gestione dimpresa) // using the *snowflake* model
	16: $children \leftarrow$ [Tecnici delle gestione dimpresa]
	19: $C4 \leftarrow$ createClass(Tecnici delle gestione dimpresa)
	21: $rel4 \leftarrow$ ExternalResource.getRelation($C2$,$C4$)
	26: $noParentTerms \leftarrow$ [Tecnici delle scienze informatiche;Tecnici delle gestione dimpresa]
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Tecnici delle scienze informatiche) // using the *snowflake* model
	16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Tecnici delle gestione dimpresa)
	26: $noParentTerms \leftarrow \emptyset$

Time Complexity	$O(n^2)$
Additional Notes	• $noParentTerms$, $children$, $allChildren$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *childrenOf* is a function that returns the children of a given term. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list.

Continued on next page

Table 7.3: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology schema (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $C = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CC \longrightarrow OS:$ $Ct_G \longrightarrow C$ $Ct_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-TX-01 [SFBG+07]

7.2.1.4 Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema

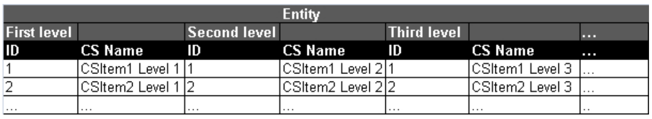
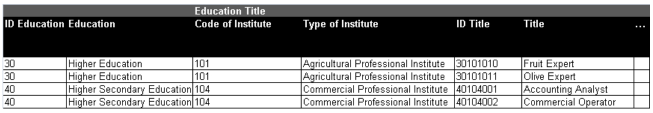
The pattern for re-engineering non-ontological resource, shown in Table 7.4, provides a guide to transform a classification scheme into an ontology schema. The classification scheme is modelled with a flattened data model.

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Flattened data model into an ontology Schema
Identifier	PR-NOR-CLTX-04
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the flattened model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on a classification published as one table with a column for each classification level.

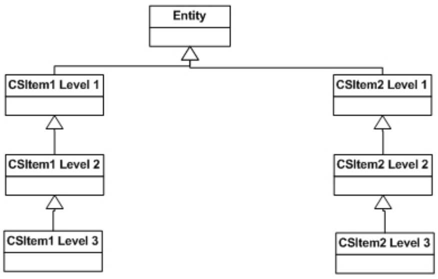
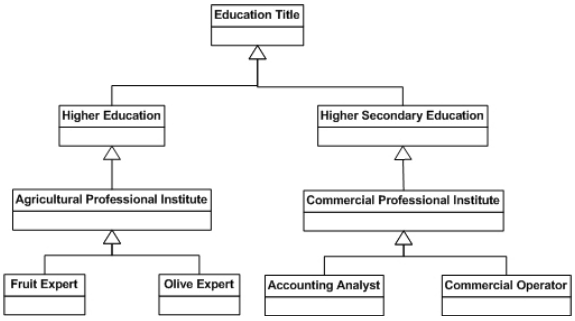
Continued on next page

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">Pattern for Re-engineering Non-ontological Resource</th></tr>
<tr><th colspan="2">INPUT: Resource to be Re-engineered</th></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a classification scheme that follows the flattened data model.
A classification scheme is a rooted tree of terms in which each terms groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parents and children terms may vary depending of the context.
The flattened data model [MZ06] is a denormalized structure for hierarchy representations. In this case, each hierarchy level is represented on a different column. There are as many columns as levels the classification scheme has. Therefore each row has the complete path from the root to a leaf node.</td></tr>
<tr><td><b>Example</b></td><td>The Classification of Italian Education Titles published by the National Institute of Statistics (ISTAT) is represented following a flattened model. The first level of the classification (level code) is related to the education title level, which comprises values as elementary, media, university, master, etc. The second level of the classification is the type of school or institute that offers the education title. The last level is the education title itself; it has a specific specialization code and also a code that is the concatenation of the previous code levels.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="7">Entity</th></tr>
<tr><th>First level</th><th></th><th>Second level</th><th></th><th>Third level</th><th></th><th>...</th></tr>
<tr><th>ID</th><th>CS Name</th><th>ID</th><th>CS Name</th><th>ID</th><th>CS Name</th><th>...</th></tr>
<tr><td>1</td><td>CSItem1 Level 1</td><td>1</td><td>CSItem1 Level 2</td><td>1</td><td>CSItem1 Level 3</td><td>...</td></tr>
<tr><td>2</td><td>CSItem2 Level 1</td><td>2</td><td>CSItem2 Level 2</td><td>2</td><td>CSItem2 Level 3</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>..</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="7">Education Title</th></tr>
<tr><th>ID Education</th><th>Education</th><th>Code of Institute</th><th>Type of Institute</th><th>ID Title</th><th>Title</th><th>...</th></tr>
<tr><td>30</td><td>Higher Education</td><td>101</td><td>Agricultural Professional Institute</td><td>30101010</td><td>Fruit Expert</td><td></td></tr>
<tr><td>30</td><td>Higher Education</td><td>101</td><td>Agricultural Professional Institute</td><td>30101011</td><td>Olive Expert</td><td></td></tr>
<tr><td>40</td><td>Higher Secondary Education</td><td>104</td><td>Commercial Professional Institute</td><td>40104001</td><td>Accounting Analyst</td><td></td></tr>
<tr><td>40</td><td>Higher Secondary Education</td><td>104</td><td>Commercial Professional Institute</td><td>40104002</td><td>Commercial Operator</td><td></td></tr>
</table></td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. Each term in the classification scheme is mapped to a class, and the semantics of the relationship between children and parent terms are made explicit by using an external resource.</td></tr>
</table>

Continued on next page

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema (continued).

Slot	Value
Graphical Representation	
(UML) General Solution Ontology	 Entity CSItem1 Level 1 CSItem2 Level 1 CSItem1 Level 2 CSItem2 Level 2 CSItem1 Level 3 CSItem2 Level 3
(UML) Example Solution Ontology	 Education Title Higher Education Higher Secondary Education Agricultural Professional Institute Commercial Professional Institute Fruit Expert Olive Expert Accounting Analyst Commercial Operator

Continued on next page

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema (continued).

Slot	Value
PROCESS: How to Re-engineer	

General	**Require:** Identification of the parent/child by using the *flattened* model
	$noParentTerms$ ← classification scheme terms without parent
	if $noParentTerms.length > 1$ **then**
	  $entityName$ ← name of the entity that contains the classification scheme terms
	  $rootClass$ ← createClass($entityName$)
	  for $ri \in noParentTerms$ **do**
	    Ri ← createClass(ri)
	    $relation$ ← ExternalResource.getRelation($rootClass$,Ri)
	    relate($relation$,$rootClass$,Ri)
	  end for
	end if
	repeat
	  for $cei \in noParentTerms$ **do**
	    if not alreadyCreatedClassFor(cei) **then**
	      Ci ← createClass(cei)
	    end if
	    $children$ ← childrenOf(cei)
	    for $cej \in children$ **do**
	      if not alreadyCreatedClassFor(cej) **then**
	        Cj ← createClass(cej)
	      end if
	      $relation$ ← ExternalResource.getRelation(cei,cej)
	      relate($relation$,cei,cej)
	    end for
	    add($allChildren$,$children$)
	  end for
	  $noParentTerms$ ← $allChildren$
	  removeAllTerms($allchildren$)
	until isEmpty($noParentTerms$)

Continued on next page

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema (continued).

Slot	Value
Example	**Require:** Identification of the parent/child by using the *flattened* model 1: $noParentTerms \leftarrow$ [HigherEducation;Higher SecondaryEducation] 2: // $noParentTerms.length$=2 > 1 3: $entityName \leftarrow$ Education Title 4: $rootClass \leftarrow$ createClass($entityName$) 6: $R1 \leftarrow$ createClass(Higher Education) 7: $relation1 \leftarrow$ ExternalResource.getRelation($rootClass$,$R1$) 6: $R2 \leftarrow$ createClass(Higher Secondary Education) 7: $relation2 \leftarrow$ ExternalResource.getRelation($rootClass$,$R2$) 16: $children \leftarrow$ childrenOf(Higher Education) 16: $children \leftarrow$ [Agricultural Professional Institute] 19: $C1 \leftarrow$ createClass(Agricultural Professional Institute) 21: $rel1 \leftarrow$ ExternalResource.getRelation($R1$,$C1$) 16: $children \leftarrow$ childrenOf(Higher Secondary Education) 16: $children \leftarrow$ [Commercial Professional Institute] 19: $C2 \leftarrow$ createClass(Commercial Professional Institute) 21: $rel2 \leftarrow$ ExternalResource.getRelation($R2$,$C2$) 26: $noParentTerms \leftarrow$ [Agricultural Professional Institute;Commercial Professional Institute] 16: $children \leftarrow$ childrenOf(Agricultural Professional Institute) 16: $children \leftarrow$ [Fruit Expert;Olive Expert] 19: $C3 \leftarrow$ createClass(Fruit Expert) 21: $rel3 \leftarrow$ ExternalResource.getRelation($C1$,$C3$) 19: $C4 \leftarrow$ createClass(Olive Expert) 21: $rel4 \leftarrow$ ExternalResource.getRelation($C1$,$C4$) 16: $children \leftarrow$ childrenOf(Commercial Professional Institute) 16: $children \leftarrow$ [Accounting Analyst;Commercial Operator] 19: $C5 \leftarrow$ createClass(Accounting Analyst) 21: $rel5 \leftarrow$ ExternalResource.getRelation($C2$,$C5$) 19: $C6 \leftarrow$ createClass(Commercial Operator) 21: $rel6 \leftarrow$ ExternalResource.getRelation($C2$,$C6$) 26: $noParentTerms \leftarrow$ [Fruit Expert;Olive Expert;Accounting Analyst;Commercial Operator] 16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Fruit Expert) 16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Olive Expert) 16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Accounting Analyst) 16: $children \leftarrow \emptyset \leftarrow$ childrenOf(Commercial Operator) 26: $noParentTerms \leftarrow \emptyset$
Time Complexity	$O(n^2)$
Additional Notes	• $noParentTerms, children, allChildren$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *childrenOf* is a function that returns the children of a given term. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list.

Continued on next page

Table 7.4: Pattern for re-engineering a classification scheme following the flattened data model into an ontology schema (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $C = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CC \longrightarrow OS$: $Ct_G \longrightarrow C$ $Ct_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-TX-01 [SFBG$^+$07]

7.2.2 Patterns for the ABox Transformation

These patterns transform the resource schema into an ontology schema, and the resource content, into ontology instances. The ABox transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd$^+$07].

The patterns presented here deal with classification schemes. As described in Section 7.1, the schema of a classification scheme has the following components: (1) a classification scheme entity, which will be transformed into a class, and (2) a classification scheme item relationship, which will be transformed into a *subClassOf* relation.

The time complexity of the algorithms described in the Section *PROCESS: How to Re-engineering* is linear $O(n)$.

7.2.2.1 Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology

The pattern for re-engineering non-ontological resource, shown in Table 7.5, provides a guide to transform a classification scheme following the path enumeration data model into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 7.5: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Path Enumeration Data model into an Ontology Schema.
Identifier	PR-NOR-CLAX-10

Continued on next page

Table 7.5: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology (continued).

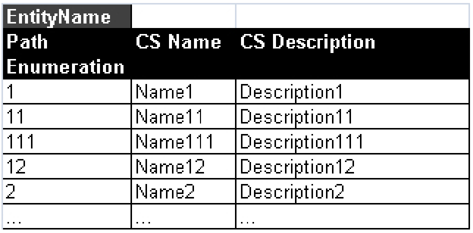
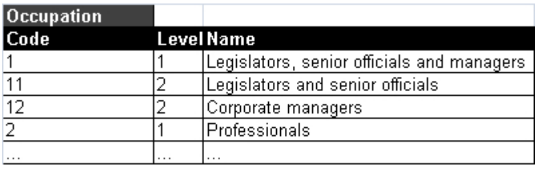
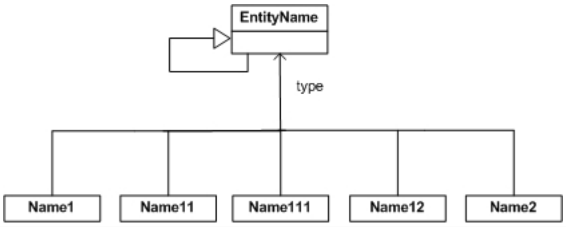
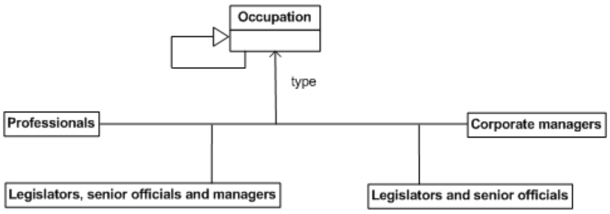
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resource (PR-NOR)</td></tr>
<tr><td colspan="2"><b>Use Case</b></td></tr>
<tr><td><b>General</b></td><td>Re-engineering a classification scheme following the path enumeration model, to design an ontology schema.</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology based on the International Standard Classification of Occupations (for European Union purposes) ISCO-88 (COM). This classification scheme follows the path enumeration data model.</td></tr>
<tr><td colspan="2"><b>Pattern for Re-engineering Non-ontological Resource</b></td></tr>
<tr><td colspan="2"><b>INPUT: Resource to be Re-engineered</b></td></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a classification scheme that follows the path enumeration model.
A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parent and children concepts may vary depending on the context.
The path enumeration data model [Bra05], for classification schemes, takes advantage of that there is one and only one path from the root to every term in the classification. The path enumeration model stores that path as string by concatenating either the edges or the keys of the classification scheme terms in the path.</td></tr>
<tr><td><b>Example</b></td><td>The International Standard Classification of Occupations (for European Union purposes), 1988 version: ISCO-88 (COM) published by Eurostat is modeled with the path enumeration data model. This classification scheme is available at <code>http://ec.europa.eu/eurostat/ramon/</code></td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>EntityName</th><th></th><th></th></tr>
<tr><th>Path Enumeration</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>Name1</td><td>Description1</td></tr>
<tr><td>11</td><td>Name11</td><td>Description11</td></tr>
<tr><td>111</td><td>Name111</td><td>Description111</td></tr>
<tr><td>12</td><td>Name12</td><td>Description12</td></tr>
<tr><td>2</td><td>Name2</td><td>Description2</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th>Occupation</th><th></th><th></th></tr>
<tr><th>Code</th><th>Level</th><th>Name</th></tr>
<tr><td>1</td><td>1</td><td>Legislators, senior officials and managers</td></tr>
<tr><td>11</td><td>2</td><td>Legislators and senior officials</td></tr>
<tr><td>12</td><td>2</td><td>Corporate managers</td></tr>
<tr><td>2</td><td>1</td><td>Professionals</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. The classification scheme entity will be transformed into a class. The classification scheme item relationship will be transformed either to a <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 7.5: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology (continued).

Slot	Value
Graphical Representation	
(UML) General Solution Ontology	 EntityName type Name1 Name11 Name111 Name12 Name2
(UML) Example Solution Ontology	 Occupation type Professionals Corporate managers Legislators, senior officials and managers Legislators and senior officials
PROCESS: How to Re-engineer	
General	1: $entityName \leftarrow$ name of the entity that contains the c.s. terms 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ classification scheme terms 6: **for** $csi \in csTerms$ **do** 7: $Ii \leftarrow$ createInstance(csi) 8: setInstanceOfClass(Ii,$mainClass$) 9: **end for**
Example	2: $mainClass \leftarrow$ createClass(Occupation) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ [Legislators, senior officials and managers;Legislators and senior officials;Corporate managers;Professionals] 7: $I1 \leftarrow$ createInstance(Legislators, senior officials and managers) 8: setInstanceOfClass($I1$,$mainClass$) 7: $I2 \leftarrow$ createInstance(Legislators and senior officials) 8: setInstanceOfClass($I2$,$mainClass$) 7: $I3 \leftarrow$ createInstance(Corporate managers) 8: setInstanceOfClass($I3$,$mainClass$) 7: $I4 \leftarrow$ createInstance(Professionals) 8: setInstanceOfClass($I4$,$mainClass$)
Time Complexity	$O(n)$
Additional Notes	• $csTerms$ is a list that does not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class.

Continued on next page

Table 7.5: Pattern for re-engineering a classification scheme following the path enumeration data model into an ontology (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $CS = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CS \longrightarrow OS:$ $CG \longrightarrow C$ $CA \longrightarrow A$ $CR \longrightarrow R \cup S$ $CC \longrightarrow KB:$ $CI \longrightarrow I$ $Ct_G \longrightarrow t_C$ $Ct_A \longrightarrow t_A$ $Ct_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG+07]

7.2.2.2 Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology

The pattern for re-engineering non-ontological resource, shown in Table 7.6, provides a guide to transform a classification scheme following the adjacency list data model into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 7.6: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Adjacency List Data model into an Ontology
Identifier	PR-NOR-CLAX-11
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the adjacency list model, to design an ontology.
Example	Suppose that someone wants to build an ontology based on the water areas classification published by FAO. This classification scheme follows the adjacency list data model.

Continued on next page

Table 7.6: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology (continued).

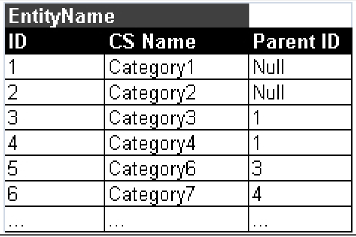
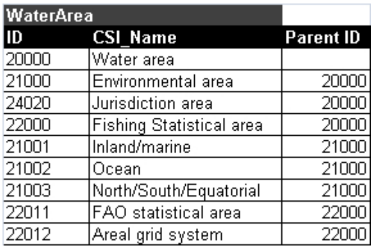
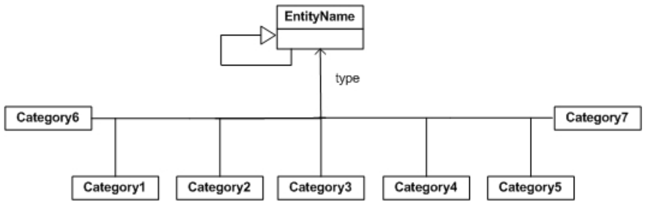
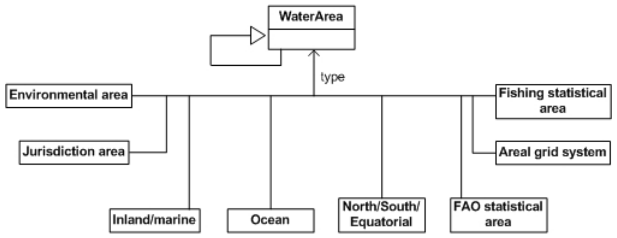
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td colspan="2"><b>Pattern for Re-engineering Non-ontological Resource</b></td></tr>
<tr><td colspan="2"><b>INPUT: Resource to be Re-engineered</b></td></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a classification scheme that follows the adjacency list 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 parent and children concepts may vary depending on the context.
The adjacency list data model [Bra05] for hierarchical classifications proposes to create an entity which holds a list of items with a linking column associated to their parent items.</td></tr>
<tr><td><b>Example</b></td><td>The FAO classification for water areas groups them according to some different criteria, such as environment, statistics, and jurisdiction, among others.
This classification scheme is available at
`http://www.fao.org/figis/servlet/RefServlet`</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="3">EntityName</th></tr>
<tr><th>ID</th><th>CS Name</th><th>Parent ID</th></tr>
<tr><td>1</td><td>Category1</td><td>Null</td></tr>
<tr><td>2</td><td>Category2</td><td>Null</td></tr>
<tr><td>3</td><td>Category3</td><td>1</td></tr>
<tr><td>4</td><td>Category4</td><td>1</td></tr>
<tr><td>5</td><td>Category6</td><td>3</td></tr>
<tr><td>6</td><td>Category7</td><td>4</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="3">WaterArea</th></tr>
<tr><th>ID</th><th>CSI_Name</th><th>Parent ID</th></tr>
<tr><td>20000</td><td>Water area</td><td></td></tr>
<tr><td>21000</td><td>Environmental area</td><td>20000</td></tr>
<tr><td>24020</td><td>Jurisdiction area</td><td>20000</td></tr>
<tr><td>22000</td><td>Fishing Statistical area</td><td>20000</td></tr>
<tr><td>21001</td><td>Inland/marine</td><td>21000</td></tr>
<tr><td>21002</td><td>Ocean</td><td>21000</td></tr>
<tr><td>21003</td><td>North/South/Equatorial</td><td>21000</td></tr>
<tr><td>22011</td><td>FAO statistical area</td><td>22000</td></tr>
<tr><td>22012</td><td>Areal grid system</td><td>22000</td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. The classification scheme item will be transformed into a class. The classification scheme item relationship will be transformed into a <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 7.6: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology (continued).

Slot	Value
Graphical Representation	
(UML) General Solution Ontology	 EntityName type Category6 Category7 Category1 Category2 Category3 Category4 Category5
(UML) Example Solution Ontology	 WaterArea type Environmental area Fishing statistical area Jurisdiction area Areal grid system Inland/marine Ocean North/South/ Equatorial FAO statistical area
PROCESS: How to Re-engineer	
General	$entityName \leftarrow$ name of the entity that contains the c.s. terms $mainClass \leftarrow$ createClass($entityName$) $relation \leftarrow$ *subClassOf* relate($relation$,$mainClass$,$mainClass$) $csTerms \leftarrow$ classification scheme terms **for** $csi \in csTerms$ **do** $Ii \leftarrow$ createInstance(csi) setInstanceOfClass(Ii,$mainClass$) **end for**

Continued on next page

Table 7.6: Pattern for re-engineering a classification scheme following the adjacency list data model into an ontology (continued).

Slot	Value
Example	2: $mainClass$ ← createClass(WaterArea) 3: $relation$ ← *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms$ ← [Environmental area;Jurisdiction area;Fishing Statistical area;Inland/marine;Ocean;North/South/Equatorial;FAO statistical area;Areal grid system] 7: $I1$ ← createInstance(Environmental area) 8: setInstanceOfClass($I1$,$mainClass$) 7: $I2$ ← createInstance(Jurisdiction area) 8: setInstanceOfClass($I2$,$mainClass$) 7: $I3$ ← createInstance(Fishing Statistical area) 8: setInstanceOfClass($I3$,$mainClass$) 7: $I4$ ← createInstance(Inland/marine) 8: setInstanceOfClass($I4$,$mainClass$) 7: $I5$ ← createInstance(Ocean) 8: setInstanceOfClass($I5$,$mainClass$) 7: $I6$ ← createInstance(North/South/Equatorial) 8: setInstanceOfClass($I6$,$mainClass$) 7: $I7$ ← createInstance(FAO statistical area) 8: setInstanceOfClass($I7$,$mainClass$) 7: $I8$ ← createInstance(Areal grid system) 8: setInstanceOfClass($I8$,$mainClass$)
Time Complexity	$O(n)$
Additional Notes	• $csTerms$ is a list that does not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class.
Formal Transformation	
General	Classification Scheme: $CS = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CS \longrightarrow OS$: $CG \longrightarrow C$ $CA \longrightarrow A$ $CR \longrightarrow R \cup S$ $CC \longrightarrow KB$: $CI \longrightarrow I$ $Ct_G \longrightarrow t_C$ $Ct_A \longrightarrow t_A$ $Ct_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG+07]

7.2.2.3 Pattern for re-engineering a classification scheme following the snowflake data model into an ontology

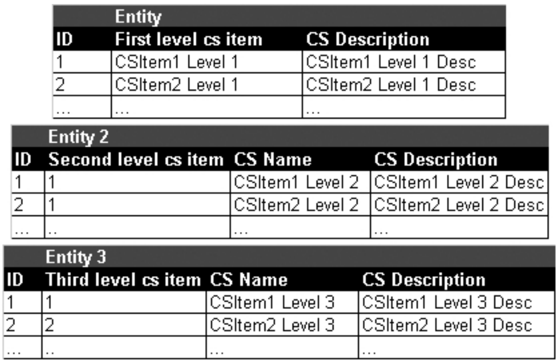
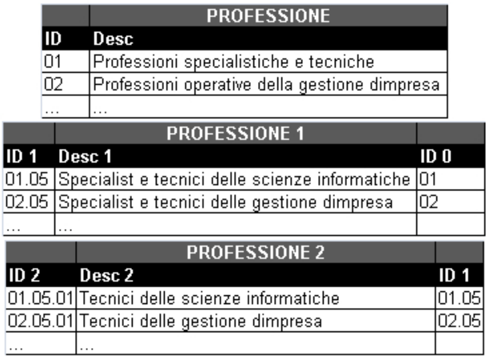
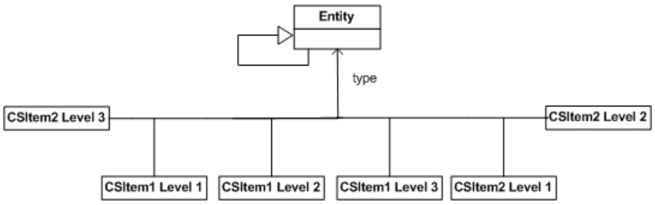
The pattern for re-engineering non-ontological resource, shown in Table 7.7, provides a guide to transform a classification scheme into an ontology. The classification scheme is modeled with a snowflake data model. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 7.7: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Snowflake Data model into an Ontology
Identifier	PR-NOR-CLAX-12
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the snowflake model, to design an ontology schema.
Example	Suppose that someone wants to build an ontology based on an occupation hierarchical classification following the snowflake data model.
Pattern for Re-engineering Non-ontological Resource	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a classification scheme that follows the snowflake model. A classification scheme is a rooted tree of terms, in which each term groups entities by some particular degree of similarity. The semantics of the hierarchical relation between parent and children concepts may vary depending on the context. The snowflake data model [MZ06] is a normalized structure for hierarchy representations. In this case, the classification scheme terms are grouped by levels or entities. There are as many groups as levels the classification scheme has.
Example	Snowflake models are widely used on data warehouses to build hierarchical classifications on structures known as dimensions. Some examples of dimension are Time, Product Category, Geography, Occupations, etc. In this pattern the example is an occupation hierarchical classification hold on four different tables, one for each level (PROFESSIONI_0, PROFESSIONI_1, PROFESSIONI_2, PROFESSIONI_3).

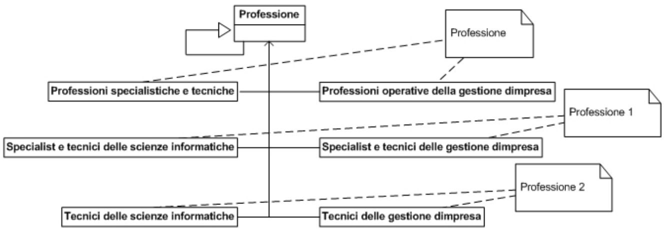
Continued on next page

Table 7.7: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>

<table>
<tr><th colspan="3">Entity</th></tr>
<tr><th>ID</th><th>First level cs item</th><th>CS Description</th></tr>
<tr><td>1</td><td>CSItem1 Level 1</td><td>CSItem1 Level 1 Desc</td></tr>
<tr><td>2</td><td>CSItem2 Level 1</td><td>CSItem2 Level 1 Desc</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th colspan="4">Entity 2</th></tr>
<tr><th>ID</th><th>Second level cs item</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>1</td><td>CSItem1 Level 2</td><td>CSItem1 Level 2 Desc</td></tr>
<tr><td>2</td><td>1</td><td>CSItem2 Level 2</td><td>CSItem2 Level 2 Desc</td></tr>
<tr><td>...</td><td>..</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th colspan="4">Entity 3</th></tr>
<tr><th>ID</th><th>Third level cs item</th><th>CS Name</th><th>CS Description</th></tr>
<tr><td>1</td><td>1</td><td>CSItem1 Level 3</td><td>CSItem1 Level 3 Desc</td></tr>
<tr><td>2</td><td>2</td><td>CSItem2 Level 3</td><td>CSItem2 Level 3 Desc</td></tr>
<tr><td>...</td><td>..</td><td>...</td><td>...</td></tr>
</table>
</td></tr>
<tr><td><b>Example</b></td><td>

<table>
<tr><th></th><th>PROFESSIONE</th></tr>
<tr><th>ID</th><th>Desc</th></tr>
<tr><td>01</td><td>Professioni specialistiche e tecniche</td></tr>
<tr><td>02</td><td>Professioni operative della gestione dimpresa</td></tr>
<tr><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th></th><th>PROFESSIONE 1</th><th></th></tr>
<tr><th>ID 1</th><th>Desc 1</th><th>ID 0</th></tr>
<tr><td>01.05</td><td>Specialist e tecnici delle scienze informatiche</td><td>01</td></tr>
<tr><td>02.05</td><td>Specialist e tecnici delle gestione dimpresa</td><td>02</td></tr>
<tr><td>...</td><td>...</td><td></td></tr>
</table>
<table>
<tr><th></th><th>PROFESSIONE 2</th><th></th></tr>
<tr><th>ID 2</th><th>Desc 2</th><th>ID 1</th></tr>
<tr><td>01.05.01</td><td>Tecnici delle scienze informatiche</td><td>01.05</td></tr>
<tr><td>02.05.01</td><td>Tecnici delle gestione dimpresa</td><td>02.05</td></tr>
<tr><td>...</td><td>...</td><td></td></tr>
</table>
</td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG$^+$07]. The classification scheme entity will be transformed into a class. The classification scheme item relationship will be transformed into a <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>


</td></tr>
<tr><td colspan="2" align="right">Continued on next page</td></tr>
</table>

Table 7.7: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology (continued).

Slot	Value
(UML) Example Solution Ontology	 Professione Professione Professioni specialistiche e tecniche Professioni operative della gestione dimpresa Professione 1 Specialist e tecnici delle scienze informatiche Specialist e tecnici delle gestione dimpresa Professione 2 Tecnici delle scienze informatiche Tecnici delle gestione dimpresa
PROCESS: How to Re-engineer	
General	1: $entityName \leftarrow$ name of the entity that contains the c.s. terms 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ classification scheme terms 6: **for** $csi \in csTerms$ **do** 7: $Ii \leftarrow$ createInstance(csi) 8: setInstanceOfClass(Ii,$mainClass$) 9: **end for**
Example	2: $mainClass \leftarrow$ createClass(Professione) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ [Professioni specialistiche e tecniche;Professioni operative della gestione dimpresa;Specialist e tecnici delle scienze informatiche;Specialist e tecnici delle gestione dimpresa;Tecnici delle scienze informatiche;Tecnici delle gestione dimpresa] 7: $I1 \leftarrow$ createInstance(Professioni specialistiche e tecniche) 8: setInstanceOfClass($I1$,$mainClass$) 7: $I2 \leftarrow$ createInstance(Professioni operative della gestione dimpresa) 8: setInstanceOfClass($I2$,$mainClass$) 7: $I3 \leftarrow$ createInstance(Specialist e tecnici delle scienze informatiche) 8: setInstanceOfClass($I3$,$mainClass$) 7: $I4 \leftarrow$ createInstance(Specialist e tecnici delle gestione dimpresa) 8: setInstanceOfClass($I4$,$mainClass$) 7: $I5 \leftarrow$ createInstance(Tecnici delle scienze informatiche) 8: setInstanceOfClass($I5$,$mainClass$) 7: $I6 \leftarrow$ createInstance(Tecnici delle gestione dimpresa) 8: setInstanceOfClass($I6$,$mainClass$)
Time Complexity	$O(n)$
Additional Notes	• $csTerms$ is a list that does not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class.

Continued on next page

Table 7.7: Pattern for re-engineering a classification scheme following the snowflake data model into an ontology (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $CS = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CS \longrightarrow OS:$ $CG \longrightarrow C$ $CA \longrightarrow A$ $CR \longrightarrow R \cup S$ $CC \longrightarrow KB:$ $CI \longrightarrow I$ $Ct_G \longrightarrow t_C$ $Ct_A \longrightarrow t_A$ $Ct_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG+07]

7.2.2.4 Pattern for re-engineering a classification scheme following the flattened data model into an ontology

The pattern for re-engineering non-ontological resource, shown in Table 7.8, provides a guide to transform a classification scheme following the flattened data model into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 7.8: Pattern for re-engineering a classification scheme following the flattened data model into an ontology.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Classification Scheme following the Flattened Data model into an Ontology
Identifier	PR-NOR-CLAX-13
Type of Component	Pattern for Re-engineering Non-ontological Resource (PR-NOR)
Use Case	
General	Re-engineering a classification scheme following the flattened model, to design an ontology
Example	Suppose that someone wants to build an ontology based on a classification published as one table with a column for each classification level.

Continued on next page

Table 7.8: Pattern for re-engineering a classification scheme following the flattened data model into an ontology (continued).

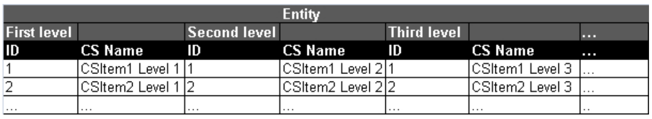
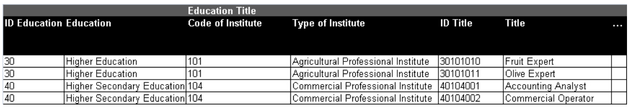
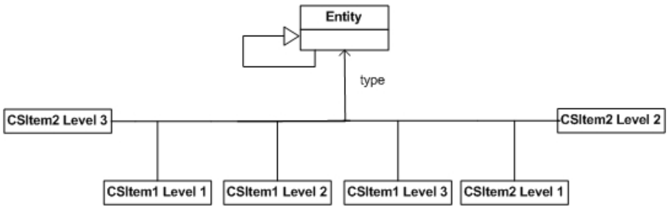
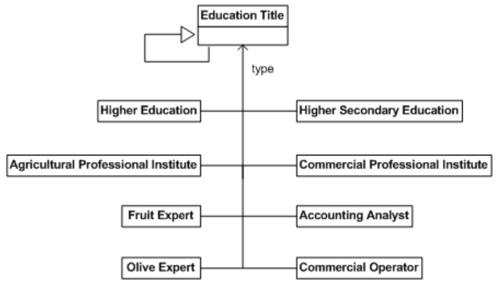
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">Pattern for Re-engineering Non-ontological Resource</th></tr>
<tr><th colspan="2">INPUT: Resource to be Re-engineered</th></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a classification scheme that follows the flattened data 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 parent and children concepts may vary depending on the context.
The flattened data model [MZ06] is a denormalized structure for hierarchy representations. In this case, each hierarchy level is represented on a different column. There are as many columns as levels the classification scheme has. Therefore each row has the complete path from the root to a leaf node.</td></tr>
<tr><td><b>Example</b></td><td>The Classification of Italian Education Titles published by the National Institute of Statistics (ISTAT) is represented following a flattened model. The first level of the classification (level code) is related to the education title level which comprises values as elementary, media, university, master, etc. The second level of the classification is the type of school or institute which offers the education title. The last level is the education title itself; it has a specific specialization code and also a code which is the concatenation of the previous code levels.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="7">Entity</th></tr>
<tr><th>First level</th><th></th><th>Second level</th><th></th><th>Third level</th><th></th><th>...</th></tr>
<tr><th>ID</th><th>CS Name</th><th>ID</th><th>CS Name</th><th>ID</th><th>CS Name</th><th>...</th></tr>
<tr><td>1</td><td>CSItem1 Level 1</td><td>1</td><td>CSItem1 Level 2</td><td>1</td><td>CSItem1 Level 3</td><td>...</td></tr>
<tr><td>2</td><td>CSItem2 Level 1</td><td>2</td><td>CSItem2 Level 2</td><td>2</td><td>CSItem2 Level 3</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>..</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="7">Education Title</th></tr>
<tr><th>ID Education</th><th>Education</th><th>Code of Institute</th><th>Type of Institute</th><th>ID Title</th><th>Title</th><th>...</th></tr>
<tr><td>30</td><td>Higher Education</td><td>101</td><td>Agricultural Professional Institute</td><td>30101010</td><td>Fruit Expert</td><td></td></tr>
<tr><td>30</td><td>Higher Education</td><td>101</td><td>Agricultural Professional Institute</td><td>30101011</td><td>Olive Expert</td><td></td></tr>
<tr><td>40</td><td>Higher Secondary Education</td><td>104</td><td>Commercial Professional Institute</td><td>40104001</td><td>Accounting Analyst</td><td></td></tr>
<tr><td>40</td><td>Higher Secondary Education</td><td>104</td><td>Commercial Professional Institute</td><td>40104002</td><td>Commercial Operator</td><td></td></tr>
</table></td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the taxonomy architectural pattern (AP-TX-01) [SFBG+07]. The classification scheme entity will be transformed into a class. The classification scheme item relationship will be transformed into a <i>subClassOf</i> relation. Finally, the content of the classification scheme will be transformed into ontology instances.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 7.8: Pattern for re-engineering a classification scheme following the flattened data model into an ontology (continued).

Slot	Value
(UML) Example Solution Ontology	 Education Title type Higher Education Higher Secondary Education Agricultural Professional Institute Commercial Professional Institute Fruit Expert Accounting Analyst Olive Expert Commercial Operator
PROCESS: How to Re-engineer	
General	1: $entityName \leftarrow$ name of the entity that contains the c.s. terms 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ classification scheme terms 6: **for** $csi \in csTerms$ **do** 7: $Ii \leftarrow$ createInstance(csi) 8: setInstanceOfClass(Ii,$mainClass$) 9: **end for**
Example	2: $mainClass \leftarrow$ createClass(Education Title) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $csTerms \leftarrow$ [Higher Education; Higher Secondary Education; Agricultural Professional Institute;Commercial Professional Institute;Fruit Expert;Olive Expert;Accounting Analyst;Commercial Operator] 7: $I1 \leftarrow$ createInstance(Higher Education) 8: setInstanceOfClass($I1$,$mainClass$) 7: $I2 \leftarrow$ createInstance(Higher Secondary Education) 8: setInstanceOfClass($I2$,$mainClass$) 7: $I3 \leftarrow$ createInstance(Agricultural Professional Institute) 8: setInstanceOfClass($I3$,$mainClass$) 7: $I4 \leftarrow$ createInstance(Commercial Professional Institute) 8: setInstanceOfClass($I4$,$mainClass$) 7: $I5 \leftarrow$ createInstance(Fruit Expert) 8: setInstanceOfClass($I5$,$mainClass$) 7: $I6 \leftarrow$ createInstance(Olive Expert) 8: setInstanceOfClass($I6$,$mainClass$) 7: $I7 \leftarrow$ createInstance(Accounting Analyst) 8: setInstanceOfClass($I7$,$mainClass$) 7: $I8 \leftarrow$ createInstance(Commercial Operator) 8: setInstanceOfClass($I8$,$mainClass$)
Time Complexity	$O(n)$
Additional Notes	• $csTerms$ is a list that does not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class.

Continued on next page

Table 7.8: Pattern for re-engineering a classification scheme following the flattened data model into an ontology (continued).

Slot	Value
Formal Transformation	
General	Classification Scheme: $CS = \langle CS, CC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $CS \longrightarrow OS:$ $CG \longrightarrow C$ $CA \longrightarrow A$ $CR \longrightarrow R \cup S$ $CC \longrightarrow KB:$ $CI \longrightarrow I$ $Ct_G \longrightarrow t_C$ $Ct_A \longrightarrow t_A$ $Ct_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: TX-AP-01 [SFBG+07]

7.3 Summary

This chapter has presented the solution we provide for those aspects related to the re-engineering of classification schemes for building ontologies. Our solution addresses some of the limitations identified in the state of art of this area.

First, we review the definition of a classification scheme, including its components. Then, we provide a formal definition for the classification schemes and the identified data models and implementations for them. Finally, we present the patterns for re-engineering classification schemes into ontologies, including those for the TBox and ABox transformation approaches. The time complexity of the TBox transformation algorithm is polynomial $O(n^2)$, whereas that of the ABox transformation algorithm is linear $O(n)$. This set of patterns are used within the method presented in Chapter 6.

The solutions presented in this chapter cover contribution **C7**, which partially addresses objective **O3** (see Chapter 3). Such a contribution is evaluated in Sections 11.1.1 and 11.2.1.

Chapter 8

PATTERNS FOR RE-ENGINEERING THESAURI

A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. Thesauri are the most valuable input for creating, at reasonable cost, ontologies in many domains. They contain, readily available, a wealth of category definitions plus a hierarchy, and they reflect some degree of community consensus [HdB07]. This chapter presents a definition of thesauri, the existing standards for thesauri, the data models for representing thesauri, and our main contribution, namely, the set of patterns for re-engineering thesauri into ontologies.

8.1 Thesaurus

In the field of thesaurus development there are several standards. These standards provide some guidelines about how the thesaurus should be structured. Figure 8.1, taken from [Lab07], depicts the thesaurus standards evolution. The ISO 2788:1986 standard is the seed of the rest of the standards. The ISO 5964:1985 extends the scope of the ISO 2788:1986 adding a multilingual context. The ANSI/NISO Z39.19-2003 adds management guidelines to the principles of monolingual thesauri. The ANSI/NISO Z39.19-2003 was superseded by ANSI/NISO Z39.19-2005. The BS 8723-1:2005 and BS 8723-2:2005 are the British version of the ISO 2788.

Next we briefly describe the most important thesaurus standards.

- *ISO 2788:1986*, which is the standard that sets the guidelines for the establishment and development of monolingual thesauri [ISO86]. This standard covers some aspects of the selection of indexing terms, the procedures for the control of the vocabulary, and specifically, the way of establishing relationships among these terms (particularly those relations that are used, a priori, in the thesauri), as well as the inclusion and suppression of terms, the

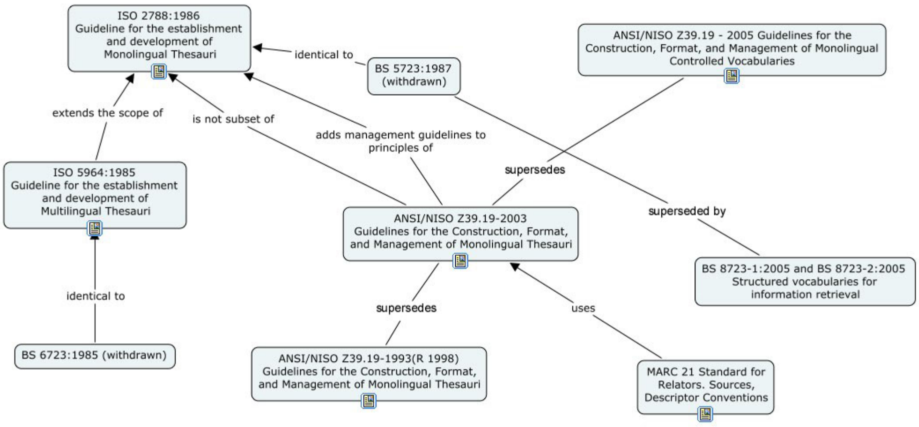

ISO 2788:1986
Guideline for the establishment and development of Monolingual Thesauri
identical to
BS 5723:1987 (withdrawn)
ANSI/NISO Z39.19 - 2005 Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies
extends the scope of
is not subset of
adds management guidelines to principles of
supersedes
ISO 5964:1985
Guideline for the establishment and development of Multilingual Thesauri
ANSI/NISO Z39.19-2003
Guidelines for the Construction, Format, and Management of Monolingual Thesauri
superseded by
BS 8723-1:2005 and BS 8723-2:2005
Structured vocabularies for information retrieval
identical to
supersedes
uses
BS 6723:1985 (withdrawn)
ANSI/NISO Z39.19-1993(R 1998)
Guidelines for the Construction, Format, and Management of Monolingual Thesauri
MARC 21 Standard for Relators. Sources, Descriptor Conventions

Figure 8.1: Thesaurus standards evolution [Lab07]

methods of compilation, the form and the content of the thesauri, the use of automatic data processing, etc. The indications established in this standard ensure the uniformity of each of the indexing areas or entities. The techniques described by the standard are based on general principles that can be applicable to any kind of subject.

- *ISO 5964:1985*, which sets the guidelines for the establishment and development of multilingual thesauri [ISO85]. These guidelines should be used in conjunction with ISO 2788 and regarded as an extension of the scope of the monolingual guidelines. The majority of procedures and recommendations contained in ISO 2788, namely, the forms of terms and the basic thesauri relationships as well as management operations such evaluation and maintenance, are equally valid for a multilingual thesaurus. Distinction is made between preferred terms and non-preferred terms.

- *ANSI/NISO Z39.19-2005*, which establishes the guidelines for the construction, format, and management of monolingual controlled vocabularies [ANS05]. This standard is related to ISO 2788. It presents guidelines and conventions for the contents, display, construction, testing, maintenance, and management of monolingual controlled vocabularies. It focuses on controlled vocabularies that are used for the representation of content objects in knowledge organization systems, including lists, synonym rings, taxonomies, and thesauri.

- *BS 8723-1:2005 and BS 8723-2:2005* [BS 05a, BS 05b]. The British Standard BS 8723-1 defines the terminology used throughout the rest of the BS

8723 series. It provides an excellent glossary for terminology relating to the use of thesauri for indexing and retrieval. The British Standard BS 8723-2 provides guidelines for the construction and maintenance of thesauri that are intended as retrieval tools. Guidance is also given for designers of software supporting the creation and maintenance process.

8.1.1 Components of a Thesaurus

A thesaurus is a collection of terms, and terms are the only type of entity considered in a thesaurus. Terms may be related to other terms traditionally using relationships, such as Broader Term (BT), Narrower Term (NT), Related Term (RT), Use For (UF), and Use (U/USE) [Soe95]. The ISO 2788:1986 [ISO86] standard proposes a thesaurus structure and identifies the thesaurus components, presented in Figure 8.2.

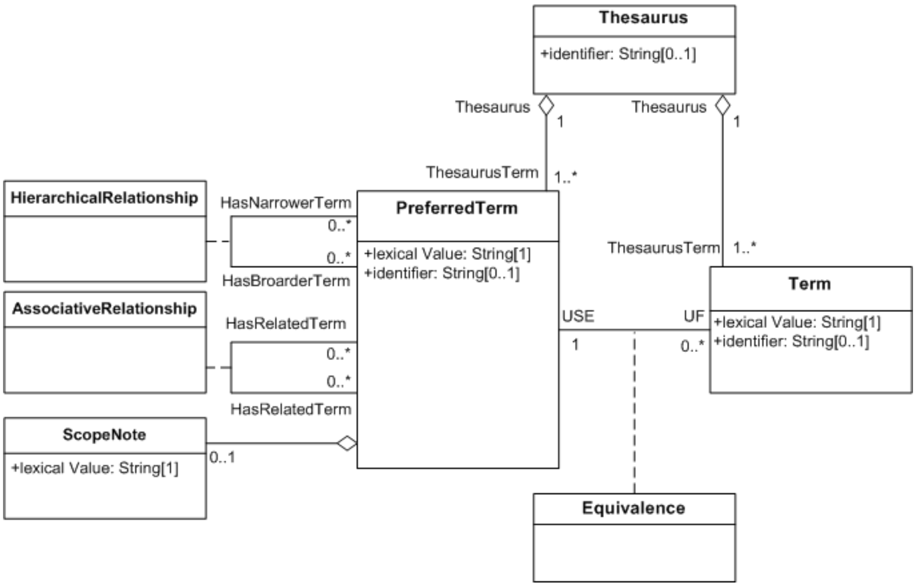


Figure 8.2: UML representation of the thesaurus components [ISO86]

- A *PreferredTerm*, also known as *descriptor*, is used consistently to represent concepts when indexing documents. It has the following elements: (1) LexicalValue, and (2) identifier.

- A *Term*, which is not assigned to documents when indexing but provided as user's entry point. It has the following elements: (1) LexicalValue, and (2) identifier.

- A *ScopeNote*, which is a note following a term explaining its coverage, specialized usage, or rules for assigning it. The *ScopeNote* has a lexicalValue element.

- A *HierarchicalRelationship*, which is a relationship between or among terms in the thesaurus that depicts broader (generic) to narrower (specific) or whole-part relationships.
- A *AssociativeRelationship*, which is a relationship between or among terms in the thesaurus that leads from one term to other terms that are related to or associated with it.
- A *Equivalence*, which is a relationship between or among terms in the thesaurus that leads to one or more terms that are to be used instead of the term from which the cross-reference is made.

8.1.2 Thesaurus Formal Definition

We formally define a thesaurus as the following tuple:

$$T = \langle TS, TC \rangle$$

Where TS represents the schema of the thesaurus, and TC represents the content of the thesaurus.

The schema of the thesaurus is defined as:

$$TS = \langle TT, TA, TB, TN, TR \rangle$$

where:

- $TT = \{tt_{pt}, tt_{npt}\}$, a set of two categories, tt_{pt} preferred term, and tt_{npt}, non-preferred term.
- $TA = \{a_1, ..., a_n\}$, a finite set of attributes, where every $a_i \subseteq TT$ x *Literal*.
- $TB = \{tsb_1, ..., tsb_n\}$, a finite set of *broader term* relations.
- $TN = \{tsn_1, ..., tsn_n\}$, a finite set of *narrower term* relations.
- $TR = \{tsr_1, ..., tsr_n\}$, a finite set of *related term* relations.

The content of the thesaurus is defined as:

$$TC = \langle TT, TA, TB, TN, TR, TI, Tt_T, Tt_A, Tt_B, Tt_N, Tt_R \rangle$$

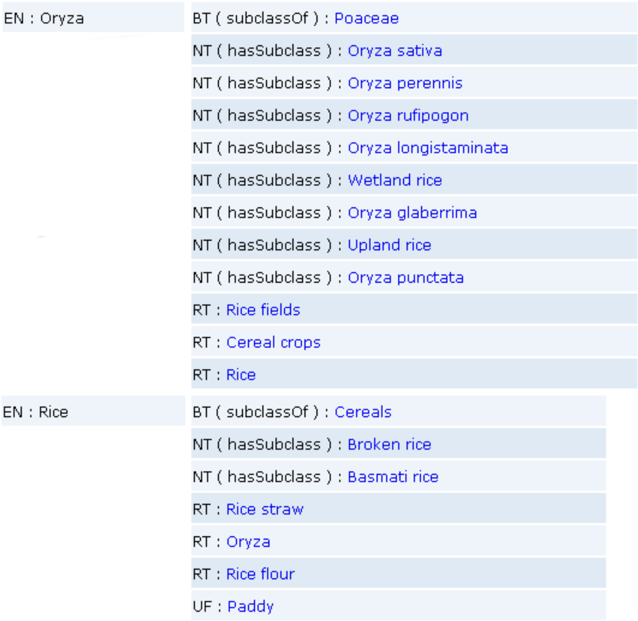
which consists of:

- The five TT, TA, TB, TN, and TR sets, as were defined before.
- A $TI = \{ti_1, ..., ti_n\}$ set whose elements are called thesaurus term identifiers
- A $Tt_T : TT \rightarrow TI$ function called thesaurus term instantiation
- A $Tt_A : TA \rightarrow TI$ function called thesaurus attribute instantiation

- A $Tt_B : TB \to TI^2$ function called thesaurus *broader term* relation instantiation
- A $Tt_N : TN \to TI^2$ function called thesaurus *narrower term* relation instantiation
- A $Tt_R : TR \to TI^2$ function called thesaurus *related term* relation instantiation

8.1.3 Thesaurus Data Models

As mentioned in Section 5.1 there are different ways of representing the knowledge encoded by a particular resource. This section presents the data models we found for thesauri. Soergel [Soe95] identifies two ways of representing the knowledge encoded by the thesauri: (1) the record-based model, and (2) the relation-based model. In order to exemplify the data models for thesauri, we use an excerpt from the FAO Thesaurus, AGROVOC[1] shown in Figure 8.3. This Figure shows the terms: Oryza and Rice. Next, we describe the data models for thesauri.



EN : Oryza	BT (subclassOf) : Poaceae
	NT (hasSubclass) : Oryza sativa
	NT (hasSubclass) : Oryza perennis
	NT (hasSubclass) : Oryza rufipogon
	NT (hasSubclass) : Oryza longistaminata
	NT (hasSubclass) : Wetland rice
	NT (hasSubclass) : Oryza glaberrima
	NT (hasSubclass) : Upland rice
	NT (hasSubclass) : Oryza punctata
	RT : Rice fields
	RT : Cereal crops
	RT : Rice
EN : Rice	BT (subclassOf) : Cereals
	NT (hasSubclass) : Broken rice
	NT (hasSubclass) : Basmati rice
	RT : Rice straw
	RT : Oryza
	RT : Rice flour
	UF : Paddy

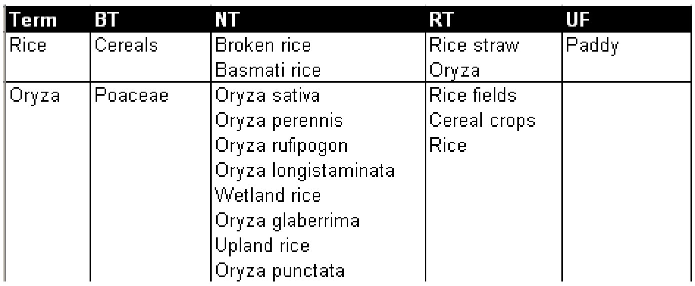
Figure 8.3: Excerpt of the AGROVOC thesaurus

8.1.3.1 Record-based Model

The record-based model, which is a denormalized structure, uses a record for every term with information about the term, such as synonyms, broader, narrower

[1] http://www.fao.org/agrovoc/

and related terms. In this model, the information is stored in large packages, and to access or change any piece of information we must get into the appropriate package. This model looks like the flattened model presented in Section 7.1.3. Figure 8.4 shows a thesaurus modelled with the record-based model.

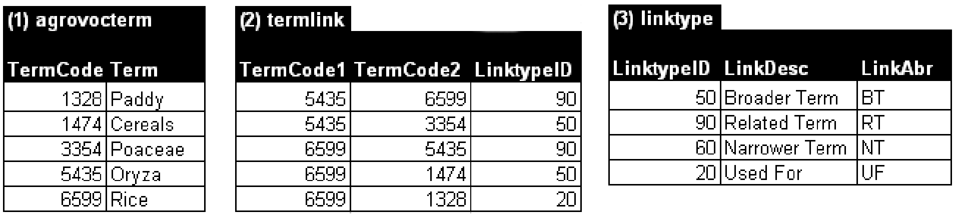


Term	BT	NT	RT	UF
Rice	Cereals	Broken rice Basmati rice	Rice straw Oryza	Paddy
Oryza	Poaceae	Oryza sativa Oryza perennis Oryza rufipogon Oryza longistaminata Wetland rice Oryza glaberrima Upland rice Oryza punctata	Rice fields Cereal crops Rice	

Figure 8.4: AGROVOC thesaurus modelled with the record-based model

8.1.3.2 Relation-based Model

The 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, but they are simply data values in a relationship record; thus new relationship types can be introduced with ease. As Figure 8.5 shows, 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 thesaurus relationships.



(1) agrovocterm

TermCode	Term
1328	Paddy
1474	Cereals
3354	Poaceae
5435	Oryza
6599	Rice

(2) termlink

TermCode1	TermCode2	LinktypeID
5435	6599	90
5435	3354	50
6599	5435	90
6599	1474	50
6599	1328	20

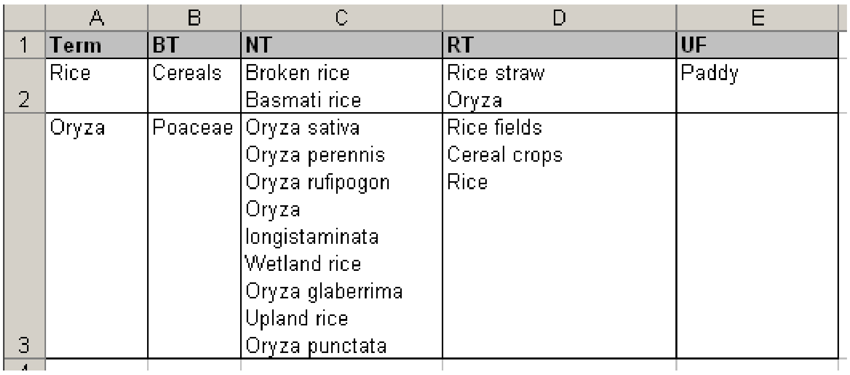
(3) linktype

LinktypeID	LinkDesc	LinkAbr
50	Broader Term	BT
90	Related Term	RT
60	Narrower Term	NT
20	Used For	UF

Figure 8.5: AGROVOC thesaurus modelled with the relation-based model

8.1.4 Thesaurus Implementations

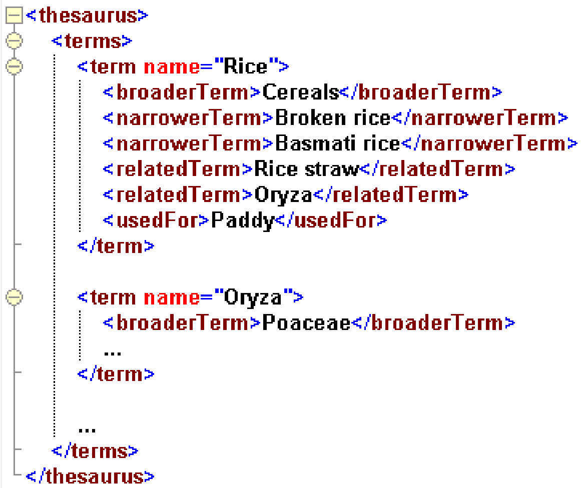
These data models can be implemented as any of the identified types on Section 5.1, i.e., databases, XML files, flat files, and spreadsheets. A direct implementation would be implemented as tables in a relational database or in a spreadsheet. Figure 8.6 presents a spreadsheet implementation of the record-based model of a thesaurus, and Figure 8.7 presents an XML implementation of the record-based



	A	B	C	D	E
1	**Term**	**BT**	**NT**	**RT**	**UF**
2	Rice	Cereals	Broken rice Basmati rice	Rice straw Oryza	Paddy
3	Oryza	Poaceae	Oryza sativa Oryza perennis Oryza rufipogon Oryza longistaminata Wetland rice Oryza glaberrima Upland rice Oryza punctata	Rice fields Cereal crops Rice	

Figure 8.6: AGROVOC thesaurus, spreadsheet implementation for the record-based model

model of a thesaurus. Both figures show the same excerpt of the AGROVOC thesaurus though represented in different implementations.



```
<thesaurus>
  <terms>
    <term name="Rice">
      <broaderTerm>Cereals</broaderTerm>
      <narrowerTerm>Broken rice</narrowerTerm>
      <narrowerTerm>Basmati rice</narrowerTerm>
      <relatedTerm>Rice straw</relatedTerm>
      <relatedTerm>Oryza</relatedTerm>
      <usedFor>Paddy</usedFor>
    </term>

    <term name="Oryza">
      <broaderTerm>Poaceae</broaderTerm>
      ...
    </term>

    ...
  </terms>
</thesaurus>
```

Figure 8.7: AGROVOC thesaurus, XML implementation for the record-based model

Figure 8.8 shows how a given type of thesauri 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 8.8 shows a thesaurus modelled following a record-based model. In this case, the thesaurus is implemented in a database and in an XML file.

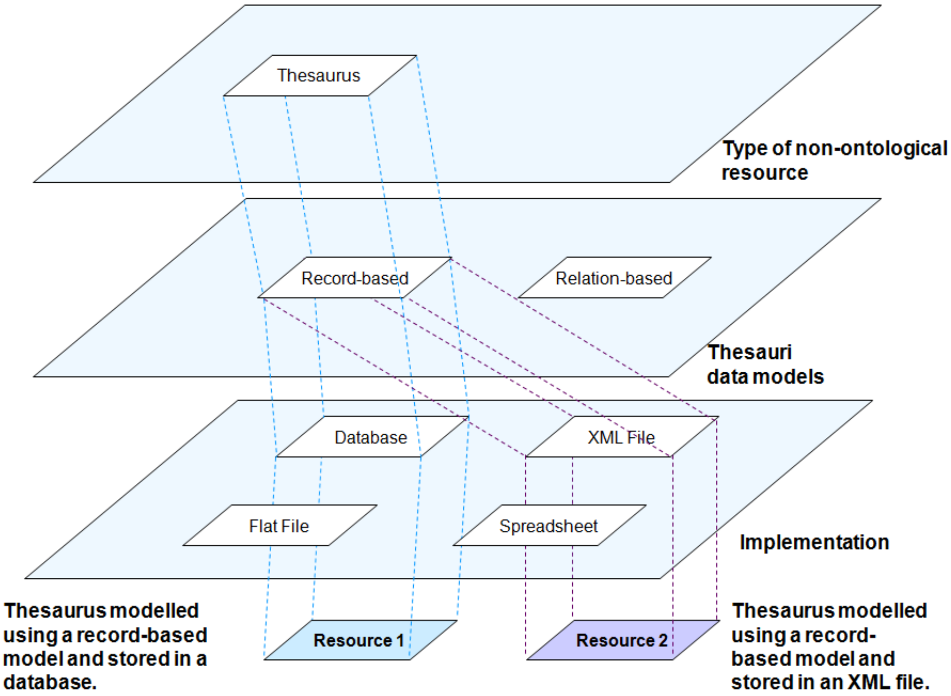

Thesaurus
Type of non-ontological resource
Record-based
Relation-based
Thesauri data models
Database
XML File
Flat File
Spreadsheet
Implementation
Thesaurus modelled using a record-based model and stored in a database.
Resource 1
Resource 2
Thesaurus modelled using a record-based model and stored in an XML file.

Figure 8.8: Thesauri categorization

8.2 Patterns for Re-engineering Thesauri into Ontologies

In this section we present re-engineering patterns (PR-NOR) for re-engineering thesauri into ontologies. The patterns follow the naming convention defined in Section 6.3. The patterns are

- Patterns for the TBox transformation
  - PR-NOR-TSTX-01. The pattern for re-engineering a thesaurus following the record-based data model into an ontology schema.
  - PR-NOR-TSTX-02. The pattern for re-engineering a thesaurus following the relation-based data model into an ontology schema.
- Patterns for the ABox transformation
  - PR-NOR-TSAX-10. The pattern for re-engineering a thesaurus following the record-based data model into an ontology.
  - PR-NOR-TSAX-11. The pattern for re-engineering a thesaurus following the relation-based data model into an ontology.

8.2.1 Patterns for the TBox Transformation

These patterns transform the resource content into an ontology schema. The TBox transformation approach tries to enforce a formal semantics to the re-engineered resources, even at the cost of changing their structure [SAd^{+}07]. For making explicit the semantics of the BT, NT and RT relations among thesaurus terms, the patterns rely on an external resource, WordNet, as we described in Section 6.4. For the UF/USE relations we use the logical pattern, SOE, proposed by Corcho et al. [CR09] and suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.

The time complexity of the algorithms described in the Section *PROCESS: How to Re-engineering* is polynomial $O(n^2)$.

8.2.1.1 Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema

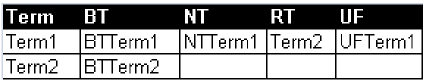
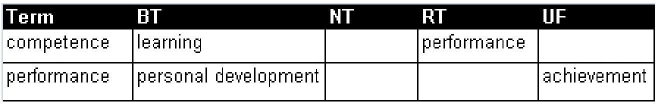
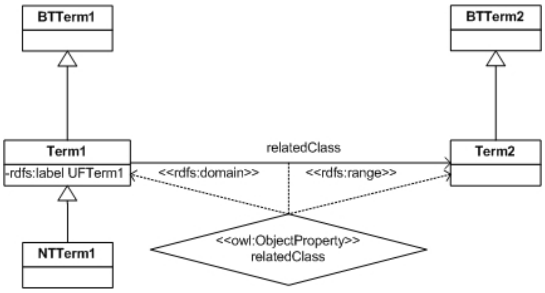
The pattern for re-engineering thesaurus, shown in Table 8.1, provides a guide to transform a thesaurus into an ontology schema. The thesaurus is modelled with a record-based data model.

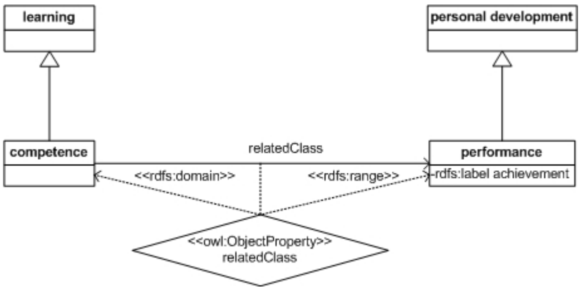
Table 8.1: Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a Thesaurus following the Record-based data model into an Ontology Schema.
Identifier	PR-NOR-TSTX-01
Type of Component	Pattern for Re-engineering Non-ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a thesaurus following the record-based model to design an ontology schema.
Example	Suppose that someone wants to build an ontology schema based on the European Training Thesaurus (ETT) following the record-based model.
Pattern for Re-engineering Non-ontological Resources	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a thesaurus that follows the record-based model. A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. The record-based data model [Soe95] is a denormalized structure, uses a record for every term with the information about the term, such as synonyms, broader, narrower and related terms.

Continued on next page

Table 8.1: Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>The European Training Thesaurus (ETT) constitutes the controlled vocabulary of reference in the field of vocational education and training (VET) in Europe. This thesaurus is available at <code>http://libserver.cedefop.europa.eu/ett/en/</code>.</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>Term</th><th>BT</th><th>NT</th><th>RT</th><th>UF</th></tr>
<tr><td>Term1</td><td>BTTerm1</td><td>NTTerm1</td><td>Term2</td><td>UFTerm1</td></tr>
<tr><td>Term2</td><td>BTTerm2</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th>Term</th><th>BT</th><th>NT</th><th>RT</th><th>UF</th></tr>
<tr><td>competence</td><td>learning</td><td></td><td>performance</td><td></td></tr>
<tr><td>performance</td><td>personal development</td><td></td><td></td><td>achievement</td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG^{+}07]. Each thesaurus term is mapped to a class. For making explicit the semantics of the BT, NT and RT relations among thesaurus terms the pattern relies on an external resource.</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
BTTerm1
BTTerm2
Term1
-rdfs:label UFTerm1
relatedClass
Term2
<<rdfs:domain>>
<<rdfs:range>>
NTTerm1
<<owl:ObjectProperty>>
relatedClass

</td></tr>
<tr><td><b>(UML) Example Solution Ontology</b></td><td>
</td></tr>
</table>

Continued on next page

Table 8.1: Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema (continued).

Slot	Value
PROCESS: How to Re-engineer	

General	**Require:** Identification of the BT/NT/RT/UF relations by using the *record-based* model
	$noBTerms \leftarrow$ terms without a broader term
	repeat
	  for $ti \in noBTerms$ **do**
	    if not alreadyCreatedClassFor(ti) **then**
	      $Ci \leftarrow$ createClass(ti)
	    end if
	    $NTerms \leftarrow$ narrowerTermOf(ti)
	    for $tj \in NTerms$ **do**
	      if alreadyCreatedClassFor(tj) **then**
	        remove($NTerms$,tj)
	      else
	        $Cj \leftarrow$ createClass(tj)
	      end if
	      $relation \leftarrow$ ExternalResource.getRelation(Ci,Cj)
	      relate($relation$,Ci,Cj)
	    end for
	    $RTerms \leftarrow$ relatedTermOf(ti)
	    for $tr \in RTerms$ **do**
	      if alreadyCreatedClassFor(tr) **then**
	        remove($RTerms$,tr)
	      else
	        $Cr \leftarrow$ createClass(tr)
	      end if
	      $relation \leftarrow$ ExternalResource.getRelation(Ci,Cr)
	      relate($relation$,Ci,Cr)
	    end for
	    $UFTerms \leftarrow$ usedForTermOf(ti)
	    for $tq \in UFTerms$ **do**
	      SOE(ti,tq)
	    end for
	    add($restOfTerms$,$NTerms$)
	    add($restOfTerms$,$RTerms$)
	  end for
	  $noBTerms \leftarrow restOfTerms$
	  removeAllTerms($restOfTerms$)
	until isEmpty($noBTerms$)

Continued on next page

Table 8.1: Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema (continued).

Slot	Value
Example	**Require:** Identification of the BT/NT/RT/UF relations by using the *record-based* model 1: $noBTerms \leftarrow$ [learning; personal development] 5: $C1 \leftarrow$ createClass(learning) 7: $NTerms \leftarrow$ narrowerTermOf(learning) 7: $NTerms \leftarrow$ [competence] // using the *record-based* model 12: $C11 \leftarrow$ createClass(competence) 14: $rel1 \leftarrow$ ExternalResource.getRelation($C1$,$C11$) 15: relate($rel1$,$C1$,$C11$) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(learning) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(learning) 31: $restOfTerms \leftarrow$ [competence] 5: $C2 \leftarrow$ createClass(personal development) 7: $NTerms \leftarrow$ narrowerTermOf(personal development) 7: $NTerms \leftarrow$ [performance] // using the *record-based* model 12: $C21 \leftarrow$ createClass(performance) 14: $rel2 \leftarrow$ ExternalResource.getRelation($C2$,$C21$) 15: relate($rel2$,$C2$,$C21$) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(personal development) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(personal development) 31: $restOfTerms \leftarrow$ [competence;performance] 34: $noBTerms \leftarrow restOfTerms \leftarrow$ [competence;performance] 35: removeAllTerms($restOfTerms$) 4: // competence class, $C11$, already created 7: $NTerms \leftarrow \emptyset \leftarrow$ narrowerTermOf(competence) 17: $RTerms \leftarrow$ relatedTermOf(competence) 17: $RTerms \leftarrow$ [performance] // using the *record-based* model 20: remove($RTerms$,performance) // performance class, C21, already created 24: $rel3 \leftarrow$ ExternalResource.getRelation($C11$,$C21$) 25: relate($rel3$,$C11$,$C21$) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(competence) 31: $restOfTerms \leftarrow \emptyset$ 4: // performance class, $C21$, already created 7: $NTerms \leftarrow \emptyset \leftarrow$ narrowerTermOf(performance) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(performance) 27: $UFTerms \leftarrow$ usedForTermOf(performance) 27: $UFTerms \leftarrow$ [achievement] // using the *record-based* model 29: SOE(performance,achievement) 31: $restOfTerms \leftarrow \emptyset$ 34: $noBTerms \leftarrow \emptyset \leftarrow restOfTerms$ 35: removeAllTerms($restOfTerms$)
Time Complexity	$O(n^2)$

Continued on next page

Table 8.1: Pattern for re-engineering a thesaurus following the record-based data model into an ontology schema (continued).

Slot	Value
Additional Notes	• $noBTerms, NTerms, RTerms, UFTerms, restOfTerms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *narrowerTermOf* is a function that returns the narrower terms of a given term. • *relatedTermOf* is a function that returns the related terms of a given term. • *usedForTermOf* is a function that returns the equivalent terms of a given term. • *remove* is a function that removes a given term from a given list. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Thesaurus: $T = \langle TS, TC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TC \longrightarrow OS$: $Tt_T \longrightarrow C$ $Tt_N \longrightarrow R \cup S$ $Tt_B \longrightarrow R \cup S$ $Tt_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG$^+$07]

8.2.1.2 Pattern for re-engineering a thesaurus following the relation-based data model into an ontology schema

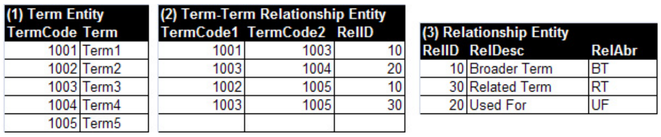
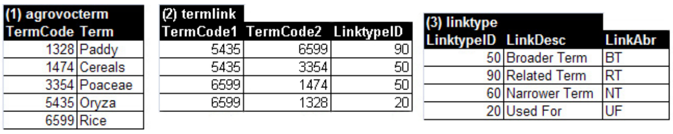
The pattern for re-engineering thesaurus, shown in Table 8.2, provides a guide to transform a thesaurus into an ontology schema. The thesaurus is modelled with a relation-based data model.

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema.

Slot	Value
General Information	
Name	Pattern for Re-engineering a thesaurus following the Relation-based Model, into an Ontology Schema
Identifier	PR-NOR-TSTX-02

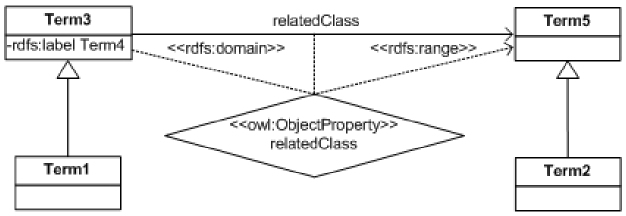
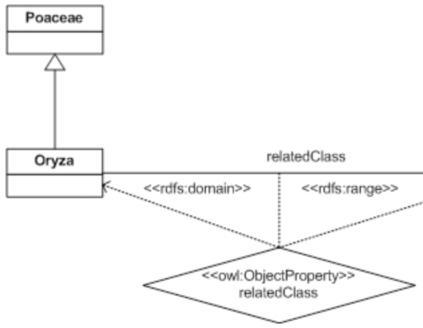
Continued on next page

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resources (PR-NOR)</td></tr>
<tr><td colspan="2"><b>Use Case</b></td></tr>
<tr><td><b>General</b></td><td>Re-engineering a thesaurus following the relation-based model to design an ontology schema.</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology schema based on earlier version of the AGROVOC Thesaurus, which is a thesaurus and it follows the relation-based model.</td></tr>
<tr><td colspan="2"><b>Pattern for Re-engineering Non-ontological Resources</b></td></tr>
<tr><td colspan="2"><b>INPUT: Resource to be Re-engineered</b></td></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a thesaurus that follows the relation-based model.
A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them.
The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.</td></tr>
<tr><td><b>Example</b></td><td>The AGROVOC Thesaurus is an structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains.
This thesaurus is available at <code>http://www.fao.org/agrovoc/</code>.</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="2">(1) Term Entity</th></tr>
<tr><th>TermCode</th><th>Term</th></tr>
<tr><td>1001</td><td>Term1</td></tr>
<tr><td>1002</td><td>Term2</td></tr>
<tr><td>1003</td><td>Term3</td></tr>
<tr><td>1004</td><td>Term4</td></tr>
<tr><td>1005</td><td>Term5</td></tr>
</table>
<table>
<tr><th colspan="3">(2) Term-Term Relationship Entity</th></tr>
<tr><th>TermCode1</th><th>TermCode2</th><th>RelID</th></tr>
<tr><td>1001</td><td>1003</td><td>10</td></tr>
<tr><td>1003</td><td>1004</td><td>20</td></tr>
<tr><td>1002</td><td>1005</td><td>10</td></tr>
<tr><td>1003</td><td>1005</td><td>30</td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
<table>
<tr><th colspan="3">(3) Relationship Entity</th></tr>
<tr><th>RelID</th><th>RelDesc</th><th>RelAbr</th></tr>
<tr><td>10</td><td>Broader Term</td><td>BT</td></tr>
<tr><td>30</td><td>Related Term</td><td>RT</td></tr>
<tr><td>20</td><td>Used For</td><td>UF</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="2">(1) agrovocterm</th></tr>
<tr><th>TermCode</th><th>Term</th></tr>
<tr><td>1328</td><td>Paddy</td></tr>
<tr><td>1474</td><td>Cereals</td></tr>
<tr><td>3354</td><td>Poaceae</td></tr>
<tr><td>5435</td><td>Oryza</td></tr>
<tr><td>6599</td><td>Rice</td></tr>
</table>
<table>
<tr><th colspan="3">(2) termlink</th></tr>
<tr><th>TermCode1</th><th>TermCode2</th><th>LinktypeID</th></tr>
<tr><td>5435</td><td>6599</td><td>90</td></tr>
<tr><td>5435</td><td>3354</td><td>50</td></tr>
<tr><td>6599</td><td>1474</td><td>50</td></tr>
<tr><td>6599</td><td>1328</td><td>20</td></tr>
</table>
<table>
<tr><th colspan="3">(3) linktype</th></tr>
<tr><th>LinktypeID</th><th>LinkDesc</th><th>LinkAbr</th></tr>
<tr><td>50</td><td>Broader Term</td><td>BT</td></tr>
<tr><td>90</td><td>Related Term</td><td>RT</td></tr>
<tr><td>60</td><td>Narrower Term</td><td>NT</td></tr>
<tr><td>20</td><td>Used For</td><td>UF</td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG+07]. Each thesaurus term is mapped to a class. For the disambiguation of the semantics of the BT, NT and RT relations among thesaurus terms the pattern relies on an external resource.</td></tr>
</table>

Continued on next page

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema (continued).

Slot	Value
Graphical Representation	
(UML) General Solution Ontology	 Term3 -rdfs:label Term4 relatedClass Term5 <<rdfs:domain>> <<rdfs:range>> <<owl:ObjectProperty>> relatedClass Term1 Term2
(UML) Example Solution Ontology	 Poaceae Cereals Oryza relatedClass Rice -rdfs:label Paddy <<rdfs:domain>> <<rdfs:range>> <<owl:ObjectProperty>> relatedClass

Continued on next page

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema (continued).

Slot	Value
PROCESS: How to Re-engineer	

General	**Require:** Identification of the BT/NT/RT/UF relations by using the *relation-based* model
	$noBTerms$ ← terms without a broader term
	repeat
	  for $ti \in noBTerms$ **do**
	    if not alreadyCreatedClassFor(ti) **then**
	      Ci ← createClass(ti)
	    end if
	    $NTerms$ ← narrowerTermOf(ti)
	    for $tj \in NTerms$ **do**
	      if alreadyCreatedClassFor(tj) **then**
	        remove($NTerms$,tj)
	      else
	        Cj ← createClass(tj)
	      end if
	      $relation$ ← ExternalResource.getRelation(Ci,Cj)
	      relate($relation$,Ci,Cj)
	    end for
	    $RTerms$ ← relatedTermOf(ti)
	    for $tr \in RTerms$ **do**
	      if alreadyCreatedClassFor(tr) **then**
	        remove($RTerms$,tr)
	      else
	        Cr ← createClass(tr)
	      end if
	      $relation$ ← ExternalResource.getRelation(Ci,Cr)
	      relate($relation$,Ci,Cr)
	    end for
	    $UFTerms$ ← usedForTermOf(ti)
	    for $tq \in UFTerms$ **do**
	      SOE(ti,tq)
	    end for
	    add($restOfTerms$,$NTerms$)
	    add($restOfTerms$,$RTerms$)
	  end for
	  $noBTerms$ ← $restOfTerms$
	  removeAllTerms($restOfTerms$)
	until isEmpty($noBTerms$)

Continued on next page

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema (continued).

Slot	Value
Example	**Require:** Identification of the BT/NT/RT/UF relations by using the *relation-based* model 1: $noBTerms \leftarrow$ [Poaceae; Cereals] 5: $C1 \leftarrow$ createClass(Poaceae) 7: $NTerms \leftarrow$ narrowerTermOf(Poaceae) 7: $NTerms \leftarrow$ [Oryza] // using the *relation-based* model 12: $C11 \leftarrow$ createClass(Oryza) 14: $rel1 \leftarrow$ ExternalResource.getRelation($C1$,$C11$) 15: relate($rel1$,$C1$,$C11$) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(Poaceae) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Poaceae) 31: $restOfTerms \leftarrow$ [Oryza] 5: $C2 \leftarrow$ createClass(Cereals) 7: $NTerms \leftarrow$ narrowerTermOf(Cereals) 7: $NTerms \leftarrow$ [Rice] // using the *relation-based* model 12: $C21 \leftarrow$ createClass(Rice) 14: $rel2 \leftarrow$ ExternalResource.getRelation($C2$,$C21$) 15: relate($rel2$,$C2$,$C21$) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(Cereals) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Cereals) 31: $restOfTerms \leftarrow$ [Oryza;Rice] 34: $noBTerms \leftarrow restOfTerms \leftarrow$ [Oryza;Rice] 35: removeAllTerms($restOfTerms$) 4: // Oryza class, $C11$, already created 7: $NTerms \leftarrow \emptyset \leftarrow$ narrowerTermOf(Oryza) 17: $RTerms \leftarrow$ relatedTermOf(Oryza) 17: $RTerms \leftarrow$ [Rice] // using the *relation-based* model 20: remove($RTerms$,Rice) // Rice class, C21, already created 24: $rel3 \leftarrow$ ExternalResource.getRelation($C11$,$C21$) 25: relate($rel3$,$C11$,$C21$) 27: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Oryza) 31: $restOfTerms \leftarrow \emptyset$ 4: // Rice, $C21$, already created 7: $NTerms \leftarrow \emptyset \leftarrow$ narrowerTermOf(Rice) 17: $RTerms \leftarrow \emptyset \leftarrow$ relatedTermOf(Rice) 27: $UFTerms \leftarrow$ usedForTermOf(Rice) 27: $UFTerms \leftarrow$ [Paddy] // using the *relation-based* model 29: SOE(Rice,Paddy) 31: $restOfTerms \leftarrow \emptyset$ 34: $noBTerms \leftarrow \emptyset \leftarrow restOfTerms$ 35: removeAllTerms($restOfTerms$)
Time Complexity	$O(n^2)$

Continued on next page

Table 8.2: Pattern for re-engineering a thesaurus following the relation-based model, into an ontology schema (continued).

Slot	Value
Additional Notes	• $noBTerms, NTerms, RTerms, UFTerms, restOfTerms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *narrowerTermOf* is a function that returns the narrower terms of a given term. • *relatedTermOf* is a function that returns the related terms of a given term. • *usedForTermOf* is a function that returns the equivalent terms of a given term. • *remove* is a function that removes a given term from a given list. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Thesaurus: $T = \langle TS, TC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TC \longrightarrow OS:$ $Tt_T \longrightarrow C$ $Tt_N \longrightarrow R \cup S$ $Tt_B \longrightarrow R \cup S$ $Tt_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG+07]

8.2.2 Patterns for the ABox Transformation

These patterns transform the resource schema into an ontology schema, and the resource content, into ontology instances. The ABox transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd+07].

As we mentioned in Section 8.1.1 the schema of a thesaurus has the following main components: (1) PreferredTerm, which will be transformed into a class, (2) Hierarchical Relationship, which will be transformed into a *subClassOf* relation, (3) Associative Relationship, which will be transformed into an *ad-hoc* relation, (4) Equivalent terms, the terms from the USE relationships, which will be transformed into labels, by using the logical pattern SOE, proposed by Corcho et al. [CR09]. Finally, the content of the thesaurus will be transformed into ontology instances.

The time complexity of the algorithms described in the Section *PROCESS: How to Re-engineering* is linear $O(n)$.

8.2.2.1 Pattern for re-engineering a thesaurus following the record-based data model into an ontology.

The pattern for re-engineering thesaurus, shown in Table 8.3, provides a guide to transform a thesaurus following the record-based data model into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 8.3: Pattern for re-engineering a thesaurus following the record-based data model into an ontology.

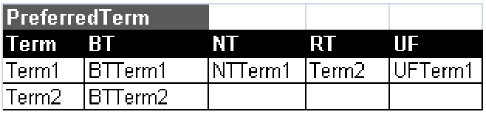
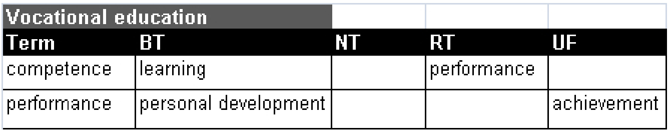
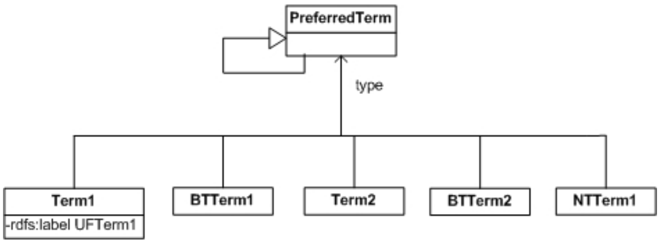
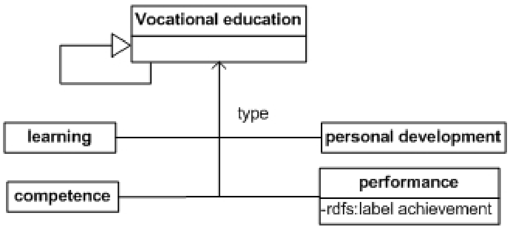
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">General Information</th></tr>
<tr><td><b>Name</b></td><td>Pattern for Re-engineering a thesaurus following the Record-based Data Model into an Ontology.</td></tr>
<tr><td><b>Identifier</b></td><td>PR-NOR-TSAX-10</td></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resources (PR-NOR)</td></tr>
<tr><th colspan="2">Use Case</th></tr>
<tr><td><b>General</b></td><td>Re-engineering a thesaurus following the record-based model to design an ontology.</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology based on the European Training Thesaurus (ETT), which is a thesaurus that follows the record-based model.</td></tr>
<tr><th colspan="2">Pattern for Re-engineering Non-ontological Resources</th></tr>
<tr><th colspan="2">INPUT: Resource to be Re-engineered</th></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a thesaurus that follows the record-based model.
A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them.
The record-based data model [Soe95] is a denormalized structure, uses a record for every term with the information about the term, such as synonyms, broader, narrower and related terms.</td></tr>
<tr><td><b>Example</b></td><td>The European Training Thesaurus (ETT) constitutes the controlled vocabulary of reference in the field of vocational education and training (VET) in Europe. This thesaurus is available at <code>http://libserver.cedefop.europa.eu/ett/en/</code>.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>PreferredTerm</th><th></th><th></th><th></th><th></th></tr>
<tr><th>Term</th><th>BT</th><th>NT</th><th>RT</th><th>UF</th></tr>
<tr><td>Term1</td><td>BTTerm1</td><td>NTTerm1</td><td>Term2</td><td>UFTerm1</td></tr>
<tr><td>Term2</td><td>BTTerm2</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 8.3: Pattern for re-engineering a thesaurus following the record-based data model into an ontology (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>


<table>
<tr><th colspan="5">Vocational education</th></tr>
<tr><th>Term</th><th>BT</th><th>NT</th><th>RT</th><th>UF</th></tr>
<tr><td>competence</td><td>learning</td><td></td><td>performance</td><td></td></tr>
<tr><td>performance</td><td>personal development</td><td></td><td></td><td>achievement</td></tr>
</table>
</td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG+07]. The thesaurus Term, schema component, will be transformed to a class, the hierarchical relationship will be transformed either to a <i>subClassOf</i> relation, the associative relationship will be transformed to an <i>ad-hoc</i> relation, and equivalent terms, the ones from the USE relationships, will be transformed to labels, by using the logical pattern proposed by Corcho et al. [CR09]. Finally, the content of the thesaurus will be transformed into ontology instances.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>


</td></tr>
<tr><td><b>(UML) Example Solution Ontology</b></td><td>


</td></tr>
</table>

Continued on next page

Table 8.3: Pattern for re-engineering a thesaurus following the record-based data model into an ontology (continued).

Slot	Value
PROCESS: How to Re-engineer	
General	1: $entityName \leftarrow$ name of the entity that contains the thesaurus terms 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation1 \leftarrow$ *relatedClass* 6: relate($relation1$,$mainClass$,$mainClass$) 7: $TTerms \leftarrow$ thesaurus terms 8: **for** $ti \in TTerms$ **do** 9: **if** not alreadyCreatedInstanceFor(ti) **then** 10: $Ii \leftarrow$ createInstance(ti) 11: setInstanceOfClass(Ii,$mainClass$) 12: **end if** 13: $UFTerms \leftarrow$ usedForTermOf(ti) 14: **for** $tq \in UFTerms$ **do** 15: SOE(ti,tq) 16: **end for** 17: **end for**
Example	1: $entityName \leftarrow$ name of the entity that contains the thesaurus terms 1: $entityName \leftarrow$ Vocational education 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation2 \leftarrow$ *relatedClass* 6: relate($relation2$,$mainClass$,$mainClass$) 7: $TTerms \leftarrow$ thesaurus terms 7: $TTerms \leftarrow$ [competence;learning;performance; personal development] 10: $I1 \leftarrow$ createInstance(competence) 11: setInstanceOfClass($I1$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(competence) 10: $I2 \leftarrow$ createInstance(learning) 11: setInstanceOfClass($I2$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(learning) 10: $I3 \leftarrow$ createInstance(performance) 11: setInstanceOfClass($I3$,$mainClass$) 13: $UFTerms \leftarrow$ usedForTermOf(performance) 13: $UFTerms \leftarrow$ [achievement] // using the *record-based* model 15: SOE(performance,achievement) 10: $I4 \leftarrow$ createInstance(personal development) 11: setInstanceOfClass($I4$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(personal development)
Time Complexity	$O(n)$

Continued on next page

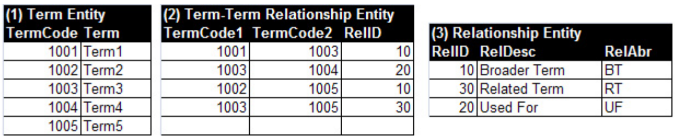
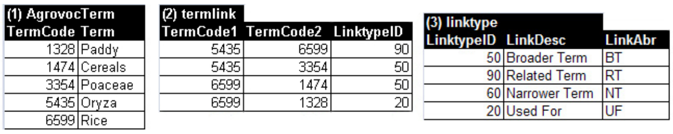
Table 8.3: Pattern for re-engineering a thesaurus following the record-based data model into an ontology (continued).

Slot	Value
Additional Notes	• $TTerms, UFTerms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class. • *usedForTermOf* is a function that returns the equivalent terms of a given term. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Thesaurus: $T = \langle TS, TC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TS \longrightarrow OS:$ $TT \longrightarrow C$ $TA \longrightarrow A$ $TB \longrightarrow R \cup S$ $TN \longrightarrow R \cup S$ $TR \longrightarrow R \cup S$ $TC \longrightarrow KB:$ $TI \longrightarrow I$ $Tt_A \longrightarrow t_A$ $Tt_N \longrightarrow t_R$ $Tt_B \longrightarrow t_R$ $Tt_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG+07]

8.2.2.2 Pattern for re-engineering a thesaurus following the relation-based data model into an ontology.

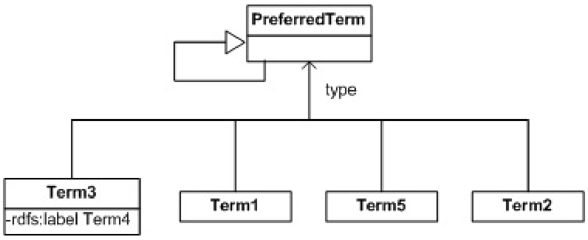
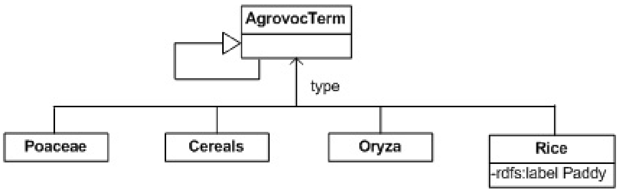
The pattern for re-engineering thesaurus, shown in Table 8.4, provides a guide to transform a thesaurus following the record-based data model into an ontology. The pattern transforms the resource schema into an ontology schema, and the resource content, into ontology instances.

Table 8.4: Pattern for re-engineering a thesaurus following the relation-based model into an ontology.

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">General Information</th></tr>
<tr><td><b>Name</b></td><td>Pattern for Re-engineering a thesaurus following the Relation-based Model into an Ontology</td></tr>
<tr><td><b>Identifier</b></td><td>PR-NOR-TSAX-11</td></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resources (PR-NOR)</td></tr>
<tr><th colspan="2">Use Case</th></tr>
<tr><td><b>General</b></td><td>Re-engineering a thesaurus following the relation-based model to design an ontology</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology based on earlier version of the AGROVOC Thesaurus, which is a thesaurus and it follows the relation-based model.</td></tr>
<tr><th colspan="2">Pattern for Re-engineering Non-ontological Resources</th></tr>
<tr><th colspan="2">INPUT: Resource to be Re-engineered</th></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a thesaurus that follows the relation-based model.
A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them.
The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.</td></tr>
<tr><td><b>Example</b></td><td>The AGROVOC Thesaurus is an structured and controlled vocabulary designed to cover the terminology of all subject fields in agriculture, forestry, fisheries, food and related domains.
This thesaurus is available at <code>http://www.fao.org/agrovoc/</code>.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table><tr><th colspan="2">(1) Term Entity</th></tr><tr><th>TermCode</th><th>Term</th></tr><tr><td>1001</td><td>Term1</td></tr><tr><td>1002</td><td>Term2</td></tr><tr><td>1003</td><td>Term3</td></tr><tr><td>1004</td><td>Term4</td></tr><tr><td>1005</td><td>Term5</td></tr></table>
<table><tr><th colspan="3">(2) Term-Term Relationship Entity</th></tr><tr><th>TermCode1</th><th>TermCode2</th><th>RelID</th></tr><tr><td>1001</td><td>1003</td><td>10</td></tr><tr><td>1003</td><td>1004</td><td>20</td></tr><tr><td>1002</td><td>1005</td><td>10</td></tr><tr><td>1003</td><td>1005</td><td>30</td></tr><tr><td></td><td></td><td></td></tr></table>
<table><tr><th colspan="3">(3) Relationship Entity</th></tr><tr><th>RelID</th><th>RelDesc</th><th>RelAbr</th></tr><tr><td>10</td><td>Broader Term</td><td>BT</td></tr><tr><td>30</td><td>Related Term</td><td>RT</td></tr><tr><td>20</td><td>Used For</td><td>UF</td></tr></table></td></tr>
<tr><td><b>Example</b></td><td>
<table><tr><th colspan="2">(1) AgrovocTerm</th></tr><tr><th>TermCode</th><th>Term</th></tr><tr><td>1328</td><td>Paddy</td></tr><tr><td>1474</td><td>Cereals</td></tr><tr><td>3354</td><td>Poaceae</td></tr><tr><td>5435</td><td>Oryza</td></tr><tr><td>6599</td><td>Rice</td></tr></table>
<table><tr><th colspan="3">(2) termlink</th></tr><tr><th>TermCode1</th><th>TermCode2</th><th>LinktypeID</th></tr><tr><td>5435</td><td>6599</td><td>90</td></tr><tr><td>5435</td><td>3354</td><td>50</td></tr><tr><td>6599</td><td>1474</td><td>50</td></tr><tr><td>6599</td><td>1328</td><td>20</td></tr></table>
<table><tr><th colspan="3">(3) linktype</th></tr><tr><th>LinktypeID</th><th>LinkDesc</th><th>LinkAbr</th></tr><tr><td>50</td><td>Broader Term</td><td>BT</td></tr><tr><td>90</td><td>Related Term</td><td>RT</td></tr><tr><td>60</td><td>Narrower Term</td><td>NT</td></tr><tr><td>20</td><td>Used For</td><td>UF</td></tr></table></td></tr>
</table>

Continued on next page

Table 8.4: Pattern for re-engineering a thesaurus following the relation-based model into an ontology (continued).

Slot	Value
OUTPUT: Designed Ontology	
General	The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01)[SFBG^{+}07]. The thesaurus Term, schema component, will be transformed to a class, the hierarchical relationship will be transformed either to a *subClassOf* relation, the associative relationship will be transformed to an *ad-hoc* relation, and equivalent terms, the ones from the USE relationships, will be transformed to labels, by using the logical pattern proposed by Corcho et al. [CR09]. Finally, the content of the thesaurus will be transformed into ontology instances.
Graphical Representation	
(UML) General Solution Ontology	 PreferredTerm type Term3 -rdfs:label Term4 Term1 Term5 Term2
(UML) Example Solution Ontology	 AgrovocTerm type Poaceae Cereals Oryza Rice -rdfs:label Paddy
PROCESS: How to Re-engineer	

General	$entityName \leftarrow$ name of the entity that contains the thesaurus terms
	$mainClass \leftarrow$ createClass($entityName$)
	$relation \leftarrow$ *subClassOf*
	relate($relation$,$mainClass$,$mainClass$)
	$relation1 \leftarrow$ *relatedClass*
	relate($relation1$,$mainClass$,$mainClass$)
	$TTerms \leftarrow$ thesaurus terms
	for $ti \in TTerms$ **do**
	  if not alreadyCreatedInstanceFor(ti) **then**
	    $Ii \leftarrow$ createInstance(ti)
	    setInstanceOfClass(Ii,$mainClass$)
	  end if
	  $UFTerms \leftarrow$ usedForTermOf(ti)
	  for $tq \in UFTerms$ **do**
	    SOE(ti,tq)
	  end for
	end for

Continued on next page

Table 8.4: Pattern for re-engineering a thesaurus following the relation-based model into an ontology (continued).

Slot	Value
Example	1: $entityName \leftarrow$ name of the entity that contains the thesaurus terms 1: $entityName \leftarrow$ AgrovocTerm 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation2 \leftarrow$ *relatedClass* 6: relate($relation2$,$mainClass$,$mainClass$) 7: $TTerms \leftarrow$ thesaurus terms 7: $TTerms \leftarrow$ [Poaceae;Cereals;Rice;Oryza] 10: $I1 \leftarrow$ createInstance(Poaceae) 11: setInstanceOfClass($I1$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Poaceae) 10: $I2 \leftarrow$ createInstance(Cereals) 11: setInstanceOfClass($I2$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Cereals) 10: $I3 \leftarrow$ createInstance(Rice) 11: setInstanceOfClass($I3$,$mainClass$) 13: $UFTerms \leftarrow$ usedForTermOf(Rice) 13: $UFTerms \leftarrow$ [Paddy] // using the *record-based* model 15: SOE(Rice,Paddy) 10: $I4 \leftarrow$ createInstance(Oryza) 11: setInstanceOfClass($I4$,$mainClass$) 13: $UFTerms \leftarrow \emptyset \leftarrow$ usedForTermOf(Oryza)
Time Complexity	$O(n)$
Additional Notes	• $TTerms, UFTerms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class. • *usedForTermOf* is a function that returns the equivalent terms of a given term. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.

Continued on next page

Table 8.4: Pattern for re-engineering a thesaurus following the relation-based model into an ontology (continued).

Slot	Value
Formal Transformation	
General	Thesaurus: $T = \langle TS, TC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $TS \longrightarrow OS:$ $TT \longrightarrow C$ $TA \longrightarrow A$ $TB \longrightarrow R \cup S$ $TN \longrightarrow R \cup S$ $TR \longrightarrow R \cup S$ $TC \longrightarrow KB:$ $TI \longrightarrow I$ $Tt_A \longrightarrow t_A$ $Tt_N \longrightarrow t_R$ $Tt_B \longrightarrow t_R$ $Tt_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG+07]

8.3 Summary

This chapter has presented our solution for the aspects related to the re-engineering of thesauri for building ontologies. Our solution addresses some of the limitations identified in the state of art in this area.

First, we review the definition of a thesaurus, including its components. Then, we provide a formal definition for the thesauri and the identified data models as well as implementations for them. Finally, we present the patterns for re-engineering thesauri into ontologies, including those for the TBox and ABox transformation approaches. The time complexity of the TBox transformation algorithm is polynomial $O(n^2)$, whereas that of the ABox transformation algorithm is linear $O(n)$. This set of patterns are used within the method presented in Chapter 6.

The solutions presented in this chapter cover contribution **C8**, which partially addresses objective **O3** (see Chapter 3). This contribution is evaluated in Sections 11.1.1 and 11.2.1.

Chapter 9

PATTERNS FOR RE-ENGINEERING LEXICA

The term lexicon is found in many ways, in conventional printed dictionaries, CD-ROM editions and Web-based versions. During the 1970s and 80s computational linguistics began to develop computational lexicons for natural language processing programs. Computational lexicons differ from dictionaries intended for human use in that they must contain much more explicit and specific linguistic information about phrases and words and must be encoded in strictly formal structures operable by computer programs. In this chapter we present a definition of lexicon, data models for representing lexicons and patterns for re-engineering lexicons into ontologies, which are our contribution to this area.

9.1 Lexicon

According to [Hir04] a lexicon is a list of words in a language (a vocabulary) that provides some knowledge of how to use each word. A lexicon may be general or domain-specific; we might have, for example, a lexicon of several thousand common words of English or German, or a lexicon of the technical terms of dentistry in some language. The words of interest are usually open-class or content words, such as nouns, verbs, and adjectives, rather than closed-class or grammatical function words such as articles, pronouns, and prepositions whose behaviour is more tightly bound to the grammar of the language. A lexicon may also include multi-word expressions such as fixed phrases (*by and large*), phrasal verbs (*tear apart*), and other common or popular expressions such as Merry Christmas! or Elvis has left the building.

Hirst [Hir04] also points out that an ordinary dictionary is an example of a lexicon. However, a dictionary is intended to be used by humans, and its style and format are unsuitable for computational use. A dictionary in a machine-readable format can serve as the basis for a computational lexicon, as in the ACQUILEX project[1], and it can also serve as the basis of a semantic hierarchy.

During the last decade the subject of lexicon standardization has been studied and developed by several projects, for example, EDR[2], EAGLES[3], MULTEXT[4], PAROLE[5], SIMPLE[6] and ISLE[7], among others.

Next, we briefly describe the most important and recent lexicon standards.

- **ISO 16642**. The ISO 16642:2003 [KSKR06] specifies a framework designed to provide guidance on the basic principles for representing data recorded in terminological data collections. This framework includes a meta-model and methods for describing specific terminological markup languages (TMLs) expressed in XML. The mechanisms for implementing constraints in a TML are defined in ISO 16642:2003, but the specific constraints for individual TMLs are not, except for the three TMLs defined in the annexes of ISO 16642:2003.

- **Lexical Markup Framework.** The Lexical Markup Framework (LMF; ISO/-CD 24613) [FGC$^+$06] is an abstract metamodel that provides a common, standardized framework for the construction of computational lexicons. LMF ensures the encoding of linguistic information in a way that enables reusability in different applications and for different tasks. LMF provides a common, shared representation of lexical objects, including morphological, syntactic and semantic aspects. LMF provides mechanisms that allow the development and integration of a variety of electronic lexical resource types. It supports lexical resource models like the Genelex [ALFZ94], the EAGLES International Standards for Language Engineering (ISLE) [CNZ96] and Multilingual ISLE Lexical Entry (MILE) models [ILC03].

- **WordNet-LMF.** WordNet-LMF [SMV09] is a dialect of ISO Lexical Markup Framework that instantiates LMF for representing wordnets. The goal of WordNet-LMF is 1) to give a preliminary assessment of LMF, by large-scale application to real lexical resources and 2) to endow WordNet with a format representation that will allow easier integration among resources sharing the same structure (i.e., wordnets). LMF specifications are fully compatible with the structural organization of lexical knowledge encoded in wordnet-like lexical resources. Starting from the meta-model provided by LMF, the additional package used in WordNet-LMF is the semantics extension package.

[1] http://www.cl.cam.ac.uk/research/nl/acquilex/

[2] http://www.wtec.org/loyola/kb/c5_s2.htm

[3] http://www.ilc.cnr.it/EAGLES/home.html

[4] http://aune.lpl.univ-aix.fr/projects/multext/

[5] http://www.elda.fr/catalogue/en/text/doc/parole.html

[6] http://www.ub.edu/gilcub/SIMPLE/simple.html

[7] http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm

9.1.1 Components of a Lexicon

Based on the WordNet-LMF standard we can identify the following components of a lexicon, presented in Figure 9.1

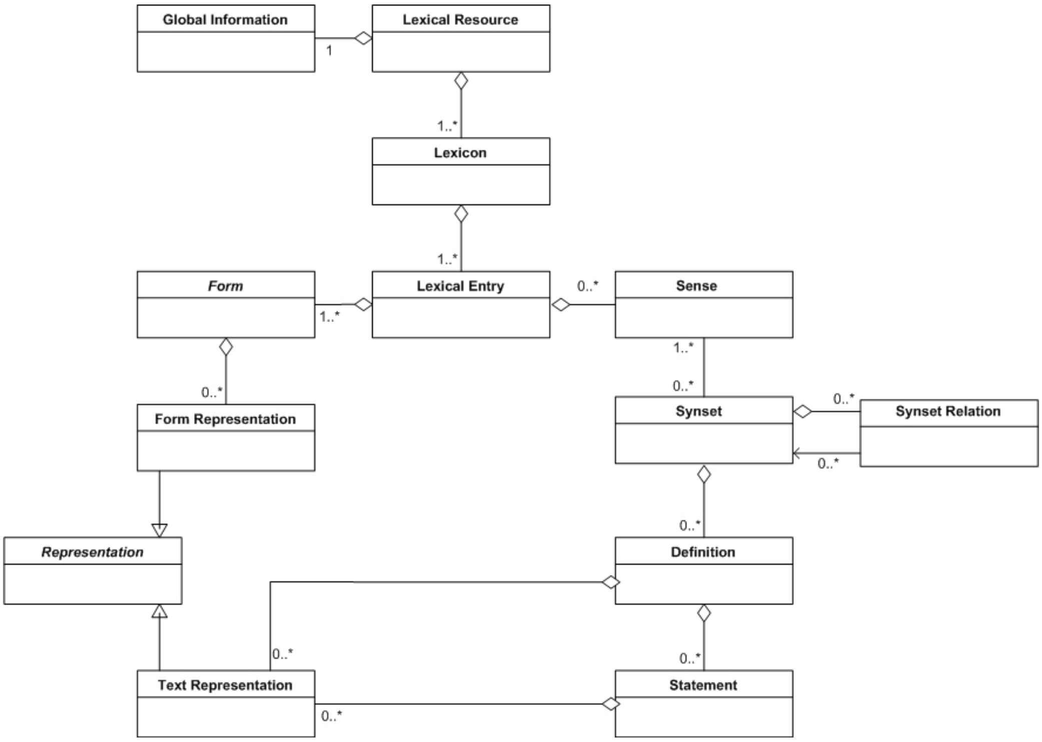


Figure 9.1: UML representation of the lexicon main components [FGC+06]

- A *Lexical Resource* component, which represents the entire resource. The Lexical Resource is a container for one or more lexicons.
- A *Global Information* component, which constitutes the administrative information and other general attributes. There is an aggregation relationship between the Lexical Resource and the Global Information in that the latter describes the administrative information and general attributes of the entire resource.
- A *Lexicon* component, which contains all the lexical entries of a given language within the entire resource. A Lexicon must contain at least one lexical entry.
- A *Lexical Entry* component, which represents a lexeme in a given language. The Lexical Entry is a container for managing the Form and Sense. Therefore, the Lexical Entry manages the relationship between the forms and their

related senses. A Lexical Entry can contain one or many different forms, and can have different sense ranging from zero to many.

- A *Form Representation* component, which constitutes one variant orthography of a Form. When there is more than one variant orthography, the Form Representation contains a Unicode string representing the Form as well as, if needed, the unique attribute-value pairs that describe the specific language, script, and orthography.

- A *Representation* component, which represents a Unicode string as well as, if needed, the unique attribute-value pairs that describe the specific language, script, and orthography.

- A *Sense* component, which represents one meaning of a lexical entry. It allows for hierarchical senses in that a sense may be more specific than another sense of the same lexical entry.

- A *Synset* component, which represents the set of shared meanings within the same language. A Synset instance can link senses of different Lexical Entry instances with the same part of speech.

- A *Synset Relation* component, which represents the oriented relationship between Synset instances.

- A *Definition* component, which represents a narrative description of a sense. It is displayed to facilitate human users to understand the meaning of a Lexical Entry and is not meant to be processable by computer programs. A Sense can have no definition or it can have many. Each Definition may be associated with zero to many Text Representation components in order to manage the text definition in more than one language or script. The narrative description can be expressed in a language and/or script different than the language of the Lexical Entry component.

- A *Statement* component, which constitutes a narrative description and refines or complements Definition. A Definition can have no Statement instances or it can have many.

- A *Text Representation* component, which represents a textual content of Definition or Statement. When there is more than one variant orthography, the Text Representation contains a Unicode string representing both the textual content and the unique attribute-value pairs that describe the specific language, script, and orthography.

9.1.2 Lexicon Formal Definition

We formally define a lexicon as the following tuple:

$$L = \langle LS, LC \rangle$$

Where LS represents the schema of the lexicon, and LC represents the content of the lexicon.

The schema of the lexicon, LS, is defined as

$$LS = \langle LE, SY, SR \rangle$$

where:

- $LE = \{le_1, ..., le_n\}$, a set of lexical entries.
- $SY = \{sy_i, ..., sy_n\}$, a set of synsets, where $sy_i \subseteq LE^m$.
- $SR = \{sr_i, ..., sr_n\}$, a set of synset relations, where $sy_i \subseteq SY$ x SY.

The content of the lexicon, LC, is defined as

$$LC = \langle LE, SY, SR, LI, SI, Lt_L, Lt_S, Lt_R \rangle$$

which consists of:

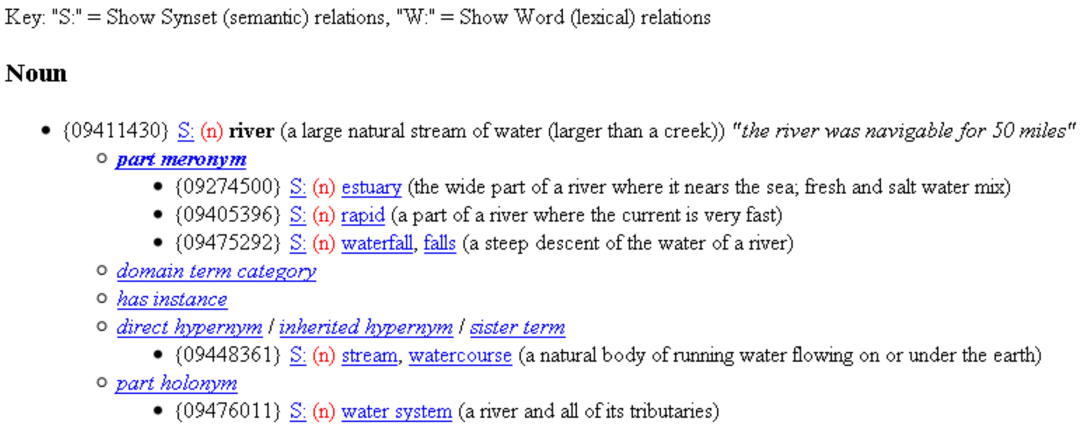
- The three LE, SY and SR sets, as defined before.
- A $LI = \{li_1, ..., li_n\}$ set whose elements are called lexical entry identifiers.
- A $SI = \{syi_1, ..., syi_n\}$ set whose elements are called synset identifiers.
- A $Lt_L : LE \rightarrow LI$ function called lexical entry instantiation.
- A $Lt_S : SY \rightarrow SI$ function called synset instantiation.
- A $Lt_R : SR \rightarrow SI^2$ function called synset relation instantiation.

9.1.3 Lexicon Data Models

As mentioned in Section 5.1 there are different ways of representing the knowledge encoded by a particular resource. After analysing several data models for lexicons, we have identified the same data models already identified for thesauri. In this section we present these data models, which are independent of the standards described in the previous section. In order to exemplify the data models for lexicons, we use an excerpt of WordNet, shown in Figure 9.2.

9.1.3.1 Record-based model

The record-based model [Soe95], which is a denormalized structure, uses a record for every element of the lexicon with information about the element, such as antonyms, hypernyms, hyponym, etc. In this model, the information is stored in large packages, and to access or change any piece of information we must get into the appropriate package. Figure 9.3 depicts this data model.

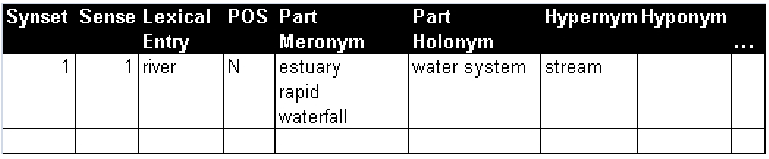


Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- {09411430} S: (n) **river** (a large natural stream of water (larger than a creek)) *"the river was navigable for 50 miles"*
  - ***part meronym***
    - {09274500} S: (n) estuary (the wide part of a river where it nears the sea; fresh and salt water mix)
    - {09405396} S: (n) rapid (a part of a river where the current is very fast)
    - {09475292} S: (n) waterfall, falls (a steep descent of the water of a river)
  - *domain term category*
  - *has instance*
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - {09448361} S: (n) stream, watercourse (a natural body of running water flowing on or under the earth)
  - *part holonym*
    - {09476011} S: (n) water system (a river and all of its tributaries)

Figure 9.2: Excerpt of WordNet lexicon

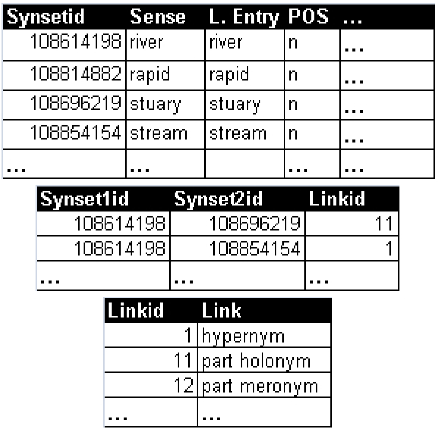


Synset	Sense	Lexical Entry	POS	Part Meronym	Part Holonym	Hypernym	Hyponym	...
1	1	river	N	estuary rapid waterfall	water system	stream		

Figure 9.3: WordNet modelled with the record-based model

9.1.3.2 Relation-based model

The relation-based model [Soe95] 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, but they are simply data values in a relationship record; thus new relationship types can be introduced with ease. In this case, Figure 9.4 shows there are three entities: (1) an element entity,



Synsetid	Sense	L. Entry	POS	...
108614198	river	river	n	...
108814882	rapid	rapid	n	...
108696219	stuary	stuary	n	...
108854154	stream	stream	n	...
...	...		...	...

Synset1id	Synset2id	Linkid
108614198	108696219	11
108614198	108854154	1
...	...	...

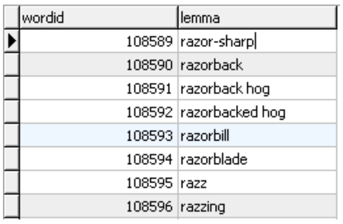
Linkid	Link
1	hypernym
11	part holonym
12	part meronym
...	...

Figure 9.4: WordNet modelled with the relation-based model

which contains the overall set of lexicon elements, (2) an element-element relationship entity, in which each record contains two different element codes and the relationship between them, and (3) a relationship source entity, which contains the overall lexicon relationships.

9.1.4 Lexicon Implementations

Finally these data models can be implemented as any of the identified types in Section 5.1, namely, databases, XML files, flat files, and spreadsheets. A direct implementation would be as tables in a relational database or in a spreadsheet. Figure 9.5 presents a database implementation of the relation-based model of WordNet, specifically the linktype table.



wordid	lemma
108589	razor-sharp
108590	razorback
108591	razorback hog
108592	razorbacked hog
108593	razorbill
108594	razorblade
108595	razz
108596	razzing

Figure 9.5: Excerpt of a WordNet database implementation

Figure 9.6 shows how a given lexicon can be modelled following one or more data models, each of which could be implemented in different ways at the implementation layer. Figure 9.6 shows an example of a lexicon modelled following a record-based model. The lexicon is implemented in a database and in an XML file.

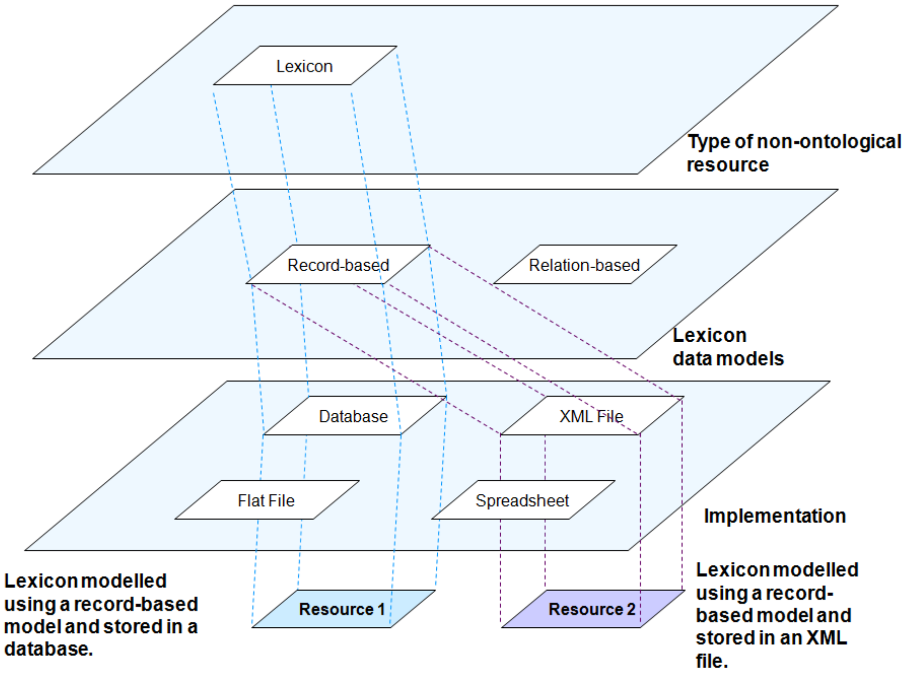

Lexicon
Type of non-ontological resource
Record-based
Relation-based
Lexicon data models
Database
XML File
Flat File
Spreadsheet
Implementation
Lexicon modelled using a record-based model and stored in a database.
Resource 1
Resource 2
Lexicon modelled using a record-based model and stored in an XML file.

Figure 9.6: Lexicon categorization

9.2 Patterns for Re-engineering Lexica into Ontologies

In this section we present re-engineering patterns (PR-NOR) for re-engineering lexica into ontologies. The patterns are

- Patterns for the TBox transformation
  - PR-NOR-LXTX-01. The pattern for re-engineering a lexicon following the record-based data model into an ontology schema.
  - PR-NOR-LXTX-02. The pattern for re-engineering a lexicon following the relation-based data model into an ontology schema.
- Patterns for the ABox transformation
  - PR-NOR-LXAX-10. The pattern for re-engineering a lexicon following the record-based data model into an ontology.
  - PR-NOR-LXAX-11. The pattern for re-engineering a lexicon following the relation-based data model into an ontology.

9.2.1 Patterns for the TBox Transformation

These patterns transform the resource content into an ontology schema. The TBox transformation approach tries to impose a formal semantics on the re-engineered resources, even at the cost of changing their structure [SAd+07]. The patterns rely on an external resource, WordNet, for making explicit the semantics of the relations among lexicon terms, as described in section 6.4. For the relations of synonyms we use the logical pattern proposed by Corcho et al. [CR09] and suggested as best practice in the context of this antipattern: The tendency to declare two classes equivalent when in fact their labels simply express synonym.

The time complexity of the algorithms described in the Section*PROCESS: How to Re-engineering* is polynomial $O(n^2)$.

9.2.1.1 Pattern for re-engineering a lexicon following the record-based data model into an ontology schema

The pattern for re-engineering lexicon, shown in Table 9.1, provides a guide to transform a lexicon into an ontology schema. The lexicon is modelled with the record-based data model.

Table 9.1: Pattern for re-engineering a lexicon following the record-based data model into an ontology schema.

Slot	Value
General Information	
Name	Pattern for re-engineering a lexicon following the record-based data model into an ontology schema.
Identifier	PR-NOR-LXTX-01
Type of Component	Pattern for Re-engineering Non-ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a lexicon following the record-based data model into an ontology schema.
Example	Suppose that someone wants to build an ontology based on the BioLexicon. The BioLexicon, and one of its variants is modelled with the record-based data model.
Pattern for Re-engineering Non-ontological Resources	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a lexicon that follows the record-based data model. A lexicon is a list of words in a language along with some knowledge of how to use each word. A lexicon may be general or domain-specific; we might have, for example, a lexicon of several thousand common words of English or German, or a lexicon of the technical terms of dentistry in some language. The record-based model [Soe95] is a denormalized structure, uses a record for every element of the lexicon with the information about the element, such as antonyms, hypernyms, hyponym, etc.

Continued on next page

Table 9.1: Pattern for re-engineering a lexicon following the record-based data model into an ontology schema (continued).

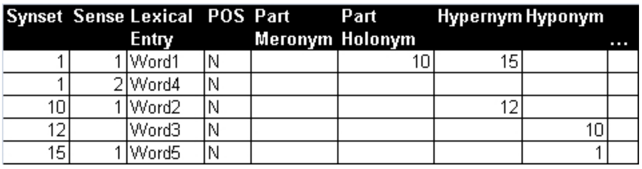
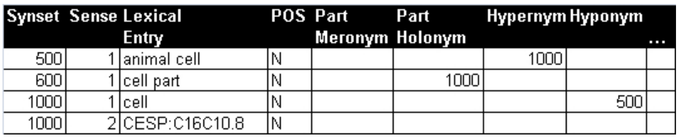
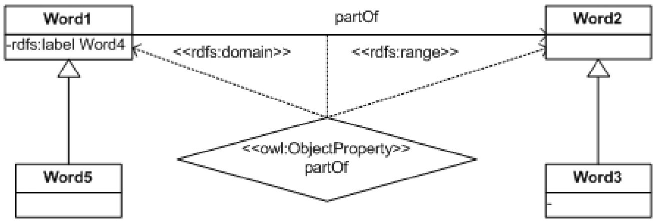
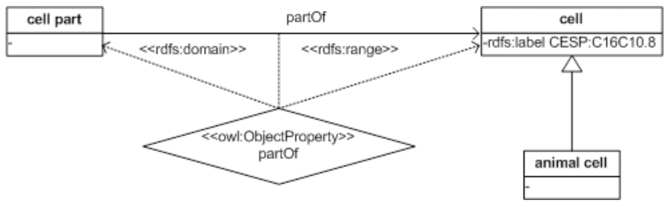
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>The BioLexicon is a large-scale terminological resource which has been developed to address the needs emerging in text mining efforts in the biomedical domain. This lexicon is available at <code>http://www.ebi.ac.uk/Rebholz-srv/BioLexicon/biolexicon.html</code></td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>Synset</th><th>Sense</th><th>Lexical Entry</th><th>POS</th><th>Part Meronym</th><th>Part Holonym</th><th>Hypernym</th><th>Hyponym</th><th>...</th></tr>
<tr><td>1</td><td>1</td><td>Word1</td><td>N</td><td></td><td>10</td><td>15</td><td></td><td></td></tr>
<tr><td>1</td><td>2</td><td>Word4</td><td>N</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>1</td><td>Word2</td><td>N</td><td></td><td></td><td>12</td><td></td><td></td></tr>
<tr><td>12</td><td></td><td>Word3</td><td>N</td><td></td><td></td><td></td><td>10</td><td></td></tr>
<tr><td>15</td><td>1</td><td>Word5</td><td>N</td><td></td><td></td><td></td><td>1</td><td></td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th>Synset</th><th>Sense</th><th>Lexical Entry</th><th>POS</th><th>Part Meronym</th><th>Part Holonym</th><th>Hypernym</th><th>Hyponym</th><th>...</th></tr>
<tr><td>500</td><td>1</td><td>animal cell</td><td>N</td><td></td><td></td><td>1000</td><td></td><td></td></tr>
<tr><td>600</td><td>1</td><td>cell part</td><td>N</td><td></td><td>1000</td><td></td><td></td><td></td></tr>
<tr><td>1000</td><td>1</td><td>cell</td><td>N</td><td></td><td></td><td></td><td>500</td><td></td></tr>
<tr><td>1000</td><td>2</td><td>CESP:C16C10.8</td><td>N</td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td colspan="2"><b>OUTPUT: Designed Ontology</b></td></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01) [SFBG+07]. Each BioLexicon synset is mapped to a class. The hyponymy/hypernym relations are mapped to subClassOf/superClassOf relations. The member meronym/holonym relations are mapped to partOf/hasPart. For synonyms we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonymy. For making explicit the semantics of rest of relations, the pattern relies on an external resource.</td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 9.1: Pattern for re-engineering a lexicon following the record-based data model into an ontology schema (continued).

Slot	Value
(UML) Example Solution Ontology	 cell part partOf cell - <<rdfs:domain>> <<rdfs:range>> -rdfs:label CESP:C16C10.8 <<owl:ObjectProperty>> partOf animal cell -
PROCESS: How to Re-engineer	
General	

```
Require: Identification of the relations by using the record-based model
$Synsets$ ← all the synsets of the lexicon
for $si \in Synsets$ do
  if not alreadyCreatedClassFor($si$) then $Ci$ ← createClass($si$) endif
  $Hyponyms$ ← hyponymOf($si$)
  for $sj \in Hyponyms$ do
    if not alreadyCreatedClassFor($sj$) then $Cj$ ← createClass($sj$) endif
    relate(subClassOf,$Ci$,$Cj$)
  end for
  $Hypernyms$ ← hypernymOf($si$)
  for $sk \in Hypernyms$ do
    if not alreadyCreatedClassFor($sk$) then $Ck$ ← createClass($sk$) endif
    relate(subClassOf,$Ck$,$Ci$)
  end for
  $Meronyms$ ← meronymOf($si$)
  for $sl \in Meronyms$ do
    if not alreadyCreatedClassFor($sl$) then $Cl$ ← createClass($sl$) endif
    relate(partOf,$Ci$,$Cl$)
  end for
  $Holonyms$ ← holonymOf($si$)
  for $sm \in Holonyms$ do
    if not alreadyCreatedClassFor($sm$) then $Cm$ ← createClass($sm$) enif
    relate(partOf,$Cm$,$Ci$)
  end for
  $Synonyms$ ← synonymOf($si$)
  for $sn \in Synonyms$ do
    CorchoEtAlPattern($si$,$tn$) // Corcho et al. [CR09]. Logical Pattern
  end for
  $RelatedSynsets$ ← relatedSynsetOf($si$)
  for $so \in RelatedSynsets$ do
    if not alreadyCreatedClassFor($so$) then $Co$ ← createClass($so$) endif
    $relation$ ← ExternalResource.getRelation($Ci$,$Co$)
    relate($relation$,$Ci$,$Co$)
  end for
end for
```

Continued on next page

Table 9.1: Pattern for re-engineering a lexicon following the record-based data model into an ontology schema (continued).

Slot	Value

Example	**Require:** Identification of the relations by using the *record-based* model
	1: $Synsets \leftarrow$ [cell;cell part;animal cell]
	3: $C1 \leftarrow$ createClass(cell)
	4: $Hyponyms \leftarrow$ hyponymOf(cell)
	4: $Hyponyms \leftarrow$ [animal cell] // using the *record-based* model
	6: $C2 \leftarrow$ createClass(animal cell)
	7: relate(subClassOf,$C2$,$C1$)
	9: $Hypernyms \leftarrow \emptyset \leftarrow$hypernymOf(cell)
	14: $Meronyms \leftarrow$ meronymOf(cell) // using the *record-based* model
	14: $Meronyms \leftarrow$ [cell part]
	16: $C3 \leftarrow$ createClass(cell part)
	17: relate(partOf,$C3$,$C1$)
	19: $Holonyms \leftarrow \emptyset \leftarrow$ holonymOf(cell)
	24: $Synonyms \leftarrow$ synonymOf(cell) // using the *record-based* model
	24: $Synonyms \leftarrow$ [CESP:C16C10.8]
	26: CorchoEtAlPattern(cell,CESP:C16C10.8) // Corcho et al. [CR09].
	28: $RelatedSynsets \leftarrow \emptyset \leftarrow$ relatedSynsetOf(cell)

Time Complexity	$O(n^2)$
Additional Notes	• $Synsets, Hyponyms, Hypernyms, Meronyms, Holonyms, Synonyms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given synset. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given synset. • *hyponymOf* is a function that returns the hyponyms of a given synset. • *hypernymOf* is a function that returns the hypernyms of a given synset. • *meronymOf* is a function that returns the meronyms of a given synset. • *holonymOf* is a function that returns the holonyms of a given synset. • *synonymOf* is a function that returns the synonyms of a given synset. • *relatedSynsetOf* is a function that returns the synsets related to a given synset. • *remove* is a function that removes a given synset from a given list. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Lexicon: $L = \langle LS, LC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $LC \longrightarrow OS$ $Lt_S \longrightarrow C$ $Lt_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG+07]

9.2.1.2 Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema

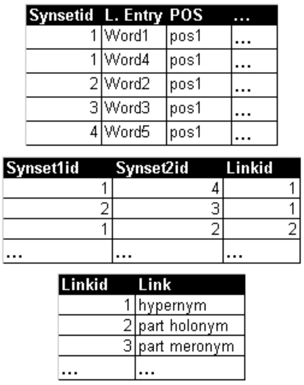
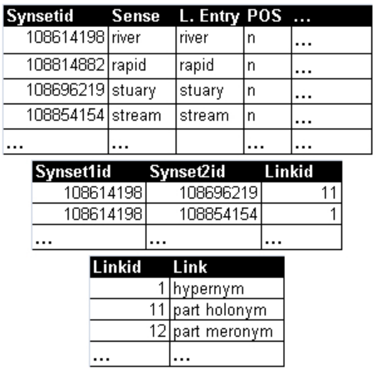
The pattern for re-engineering a lexicon, shown in Table 9.2, provides a guide to transform a lexicon into an ontology schema. The lexicon is modelled with the relation-based data model.

Table 9.2: Pattern for re-engineering a wordnet lexicon following the relation-based data model into an ontology schema.

Slot	Value
General Information	
Name	Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema.
Identifier	PR-NOR-LXTX-02
Type of Component	Pattern for Re-engineering Non-ontological Resources (PR-NOR)
Use Case	
General	Re-engineering a lexicon following the relation-based model into an ontology schema.
Example	Suppose that someone wants to build an ontology based on the Princeton WordNet. The Princeton WordNet is modelled with the relation-based data model.
Pattern for Re-engineering Non-ontological Resources	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a lexicon that follows the relation-based model. A lexicon is a list of words in a language along with some knowledge of how to use each word. A lexicon may be general or domain-specific; we might have, for example, a lexicon of several thousand common words of English or German, or a lexicon of the technical terms of dentistry in some language. The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.
Example	The Princeton WordNet is the best known computational lexicon of English. This lexicon is available at `http://wordnet.princeton.edu/`.

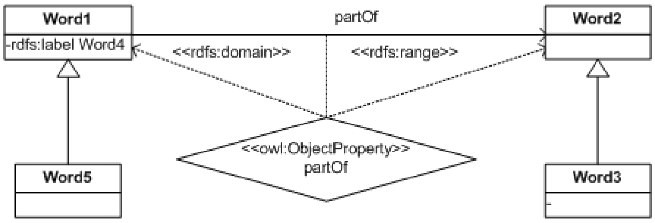
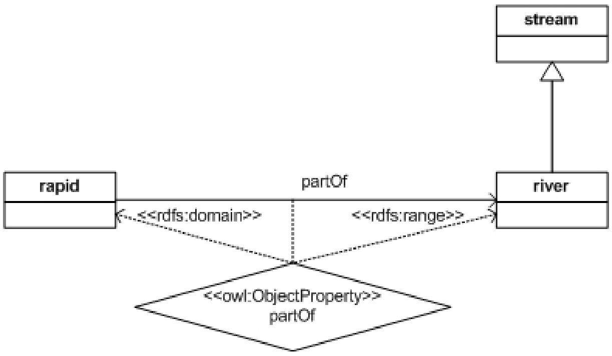
Continued on next page

Table 9.2: Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th>Synsetid</th><th>L. Entry</th><th>POS</th><th>...</th></tr>
<tr><td>1</td><td>Word1</td><td>pos1</td><td>...</td></tr>
<tr><td>1</td><td>Word4</td><td>pos1</td><td>...</td></tr>
<tr><td>2</td><td>Word2</td><td>pos1</td><td>...</td></tr>
<tr><td>3</td><td>Word3</td><td>pos1</td><td>...</td></tr>
<tr><td>4</td><td>Word5</td><td>pos1</td><td>...</td></tr>
</table>
<table>
<tr><th>Synset1id</th><th>Synset2id</th><th>Linkid</th></tr>
<tr><td>1</td><td>4</td><td>1</td></tr>
<tr><td>2</td><td>3</td><td>1</td></tr>
<tr><td>1</td><td>2</td><td>2</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th>Linkid</th><th>Link</th></tr>
<tr><td>1</td><td>hypernym</td></tr>
<tr><td>2</td><td>part holonym</td></tr>
<tr><td>3</td><td>part meronym</td></tr>
<tr><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th>Synsetid</th><th>Sense</th><th>L. Entry</th><th>POS</th><th>...</th></tr>
<tr><td>108614198</td><td>river</td><td>river</td><td>n</td><td>...</td></tr>
<tr><td>108814882</td><td>rapid</td><td>rapid</td><td>n</td><td>...</td></tr>
<tr><td>108696219</td><td>stuary</td><td>stuary</td><td>n</td><td>...</td></tr>
<tr><td>108854154</td><td>stream</td><td>stream</td><td>n</td><td>...</td></tr>
<tr><td>...</td><td>...</td><td></td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th>Synset1id</th><th>Synset2id</th><th>Linkid</th></tr>
<tr><td>108614198</td><td>108696219</td><td>11</td></tr>
<tr><td>108614198</td><td>108854154</td><td>1</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th>Linkid</th><th>Link</th></tr>
<tr><td>1</td><td>hypernym</td></tr>
<tr><td>11</td><td>part holonym</td></tr>
<tr><td>12</td><td>part meronym</td></tr>
<tr><td>...</td><td>...</td></tr>
</table></td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01) [SFBG+07]. Each WordNet synset is mapped to a class. The hyponymy/hypernym relations are mapped to subClassOf/superClassOf relations. The member meronym/holonym relations are mapped to partOf/hasPart. For synonyms we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonymy.</td></tr>
</table>

Continued on next page

Table 9.2: Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema (continued).

Slot	Value
Graphical Representation	
(UML) General Solution Ontology	 Word1 -rdfs:label Word4 partOf Word2 <<rdfs:domain>> <<rdfs:range>> <<owl:ObjectProperty>> partOf Word5 Word3
(UML) Example Solution Ontology	 stream rapid partOf river <<rdfs:domain>> <<rdfs:range>> <<owl:ObjectProperty>> partOf

Continued on next page

Table 9.2: Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema (continued).

Slot	Value
PROCESS: How to Re-engineer	
General	**Require:** Identification of the relations by using the *relation-based* model 1: $Synsets \leftarrow$ all the synsets of the lexicon 2: **for** $si \in Synsets$ **do** 3: **if** not alreadyCreatedClassFor(si) **then** $Ci \leftarrow$ createClass(si) **endif** 4: $Hyponyms \leftarrow$ hyponymOf(si) 5: **for** $sj \in Hyponyms$ **do** 6: **if** not alreadyCreatedClassFor(sj) **then** $Cj \leftarrow$ createClass(sj) **endif** 7: relate(subClassOf,Ci,Cj) 8: **end for** 9: $Hypernyms \leftarrow$ hypernymOf(si) 10: **for** $sk \in Hypernyms$ **do** 11: **if** not alreadyCreatedClassFor(sk) **then** $Ck \leftarrow$ createClass(sk) **endif** 12: relate(subClassOf,Ck,Ci) 13: **end for** 14: $Meronyms \leftarrow$ meronymOf(si) 15: **for** $sl \in Meronyms$ **do** 16: **if** not alreadyCreatedClassFor(sl) **then** $Cl \leftarrow$ createClass(sl) **endif** 17: relate(partOf,Ci,Cl) 18: **end for** 19: $Holonyms \leftarrow$ holonymOf(si) 20: **for** $sm \in Holonyms$ **do** 21: **if** not alreadyCreatedClassFor(sm) **then** $Cm \leftarrow$ createClass(sm) **enif** 22: relate(partOf,Cm,Ci) 23: **end for** 24: $Synonyms \leftarrow$ synonymOf(si) 25: **for** $sn \in Synonyms$ **do** 26: CorchoEtAlPattern(si,tn) // Corcho et al. [CR09]. Logical Pattern 27: **end for** 28: $RelatedSynsets \leftarrow$ relatedSynsetOf(si) 29: **for** $so \in RelatedSynsets$ **do** 30: **if** not alreadyCreatedClassFor(so) **then** $Co \leftarrow$ createClass(so) **endif** 31: $relation \leftarrow$ ExternalResource.getRelation(Ci,Co) 32: relate($relation$,Ci,Co) 33: **end for** 34: **end for**
Example	**Require:** Identification of the relations by using the *relation-based* model 1: $Synsets \leftarrow$ [river;rapid;stream] 3: $C1 \leftarrow$ createClass(river) 4: $Hyponyms \leftarrow \emptyset \leftarrow$ hyponymOf(river) 9: $Hypernyms \leftarrow$ hypernymOf(river) // using the *relation-based* model 9: $Hypernyms \leftarrow$ [stream] 11: $C2 \leftarrow$ createClass(stream) 12: relate(subClassOf,$C1$,$C3$) 14: $Meronyms \leftarrow$ meronymOf(river) 14: $Meronyms \leftarrow$ [rapid] // using the *relation-based* model 16: $C3 \leftarrow$ createClass(rapid) 17: relate(partOf,$C3$,$C1$) 19: $Holonyms \leftarrow \emptyset \leftarrow$ holonymOf(river) 24: $Synonyms \leftarrow \emptyset \leftarrow$ synonymOf(river) 28: $RelatedSynsets \leftarrow \emptyset \leftarrow$ relatedSynsetOf(river)
Time Complexity	$O(n^2)$

Continued on next page

Table 9.2: Pattern for re-engineering a lexicon following the relation-based data model into an ontology schema (continued).

Slot	Value
Additional Notes	• $Synsets, Hyponyms, Hypernyms, Meronyms, Holonyms, Synonyms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given synset. • *getRelation* is the algorithm 1 defined in section 6.4. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given synset. • *hyponymOf* is a function that returns the hyponyms of a given synset. • *hypernymOf* is a function that returns the hypernyms of a given synset. • *meronymOf* is a function that returns the meronyms of a given synset. • *holonymOf* is a function that returns the holonyms of a given synset. • *synonymOf* is a function that returns the synonyms of a given synset. • *relatedSynsetOf* is a function that returns the synsets related to a given synset. • *remove* is a function that removes a given synset from a given list. • *removeAllTerms* is a function that removes all the elements of a given list. • *isEmpty* checks if a list has elements or not. • *add* is a function that adds the elements of a list into another list. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Lexicon: $L = \langle LS, LC\rangle$ Ontology: $O = \langle OS, KB\rangle$ Transformation: $LC \longrightarrow OS$ $Lt_S \longrightarrow C$ $Lt_R \longrightarrow R \cup S$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG$^+$07]

9.2.2 Patterns for the ABox Transformation

These patterns transform the resource schema into an ontology schema, and the resource content, into ontology instances. The ABox transformation approach leaves the informal semantics of the re-engineered resources mostly untouched [SAd$^+$07].

As mentioned in Section 9.1, the schema of a lexica has the following main components: (1) a synset, which will be transformed to a class, (2) hyponymy/hypernym relations, which are mapped to subClassOf/superClassOf relations, (3) member meronym/holonym relations, which are mapped to partOf/hasPart, (4) synonym relations, which are mapped to labels by using the logical pattern proposed by Corcho et al. [CR09]; and the content of the lexicon, which will be transformed into ontology instances.

The time complexity of the algorithms described in the Section *PROCESS: How to Re-engineering* is linear $O(n)$.

9.2.2.1 Pattern for re-engineering a lexicon following the record-based data model into an ontology.

The pattern for re-engineering lexicon, shown in Table 9.3, provides a guide to transform a lexicon into an ontology. The lexicon is modelled with the record-based data model.

Table 9.3: Pattern for re-engineering a lexicon following the record-based data model into an ontology.

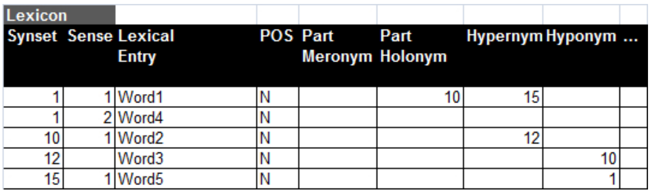
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td colspan="2"><b>General Information</b></td></tr>
<tr><td><b>Name</b></td><td>Pattern for re-engineering a lexicon following the record-based data model into an ontology.</td></tr>
<tr><td><b>Identifier</b></td><td>PR-NOR-LXAX-10</td></tr>
<tr><td><b>Type of Component</b></td><td>Pattern for Re-engineering Non-ontological Resources (PR-NOR)</td></tr>
<tr><td colspan="2"><b>Use Case</b></td></tr>
<tr><td><b>General</b></td><td>Re-engineering a lexicon following the record-based data model into an ontology.</td></tr>
<tr><td><b>Example</b></td><td>Suppose that someone wants to build an ontology based on the BioLexicon. The BioLexicon, and one of its variants is modelled with the record-based data model.</td></tr>
<tr><td colspan="2"><b>Pattern for Re-engineering Non-ontological Resources</b></td></tr>
<tr><td colspan="2"><b>INPUT: Resource to be Re-engineered</b></td></tr>
<tr><td><b>General</b></td><td>A non-ontological resource holds a lexicon that follows the record-based data model.
A lexicon contains a list of words in a language along with some knowledge of how to use each word. A lexicon may be general or domain-specific; we might have, for example, a lexicon of several thousand common words of English or German, or a lexicon of the technical terms of dentistry in some language.
The record-based model [Soe95] is a denormalized structure that uses a record for every element of the lexicon with information about the element, such as antonyms, hypernyms, hyponym, etc.</td></tr>
<tr><td><b>Example</b></td><td>The BioLexicon is a large-scale terminological resource that has been developed to address the needs emerging in text mining efforts within the biomedical domain. This lexicon is available at <code>http://www.ebi.ac.uk/Rebholz-srv/BioLexicon/biolexicon.html</code></td></tr>
<tr><td colspan="2"><b>Graphical Representation</b></td></tr>
<tr><td><b>General</b></td><td>
<table>
<tr><th colspan="9">Lexicon</th></tr>
<tr><th>Synset</th><th>Sense</th><th>Lexical Entry</th><th>POS</th><th>Part Meronym</th><th>Part Holonym</th><th>Hypernym</th><th>Hyponym</th><th>...</th></tr>
<tr><td>1</td><td>1</td><td>Word1</td><td>N</td><td></td><td>10</td><td>15</td><td></td><td></td></tr>
<tr><td>1</td><td>2</td><td>Word4</td><td>N</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>1</td><td>Word2</td><td>N</td><td></td><td></td><td>12</td><td></td><td></td></tr>
<tr><td>12</td><td></td><td>Word3</td><td>N</td><td></td><td></td><td></td><td>10</td><td></td></tr>
<tr><td>15</td><td>1</td><td>Word5</td><td>N</td><td></td><td></td><td></td><td>1</td><td></td></tr>
</table></td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 9.3: Pattern for re-engineering a lexicon following the record-based data model into an ontology (continued).

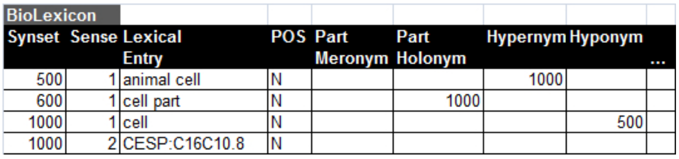
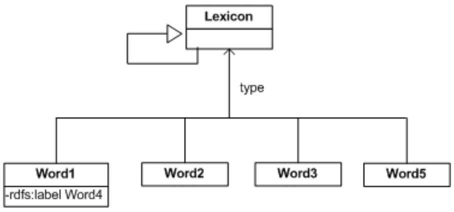
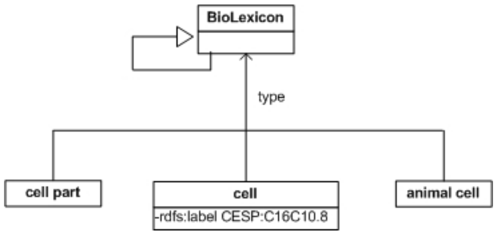
<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>
<table>
<tr><th colspan="9">BioLexicon</th></tr>
<tr><th>Synset</th><th>Sense</th><th>Lexical Entry</th><th>POS</th><th>Part Meronym</th><th>Part Holonym</th><th>Hypernym</th><th>Hyponym</th><th>...</th></tr>
<tr><td>500</td><td>1</td><td>animal cell</td><td>N</td><td></td><td></td><td>1000</td><td></td><td></td></tr>
<tr><td>600</td><td>1</td><td>cell part</td><td>N</td><td></td><td>1000</td><td></td><td></td><td></td></tr>
<tr><td>1000</td><td>1</td><td>cell</td><td>N</td><td></td><td></td><td></td><td>500</td><td></td></tr>
<tr><td>1000</td><td>2</td><td>CESP:C16C10.8</td><td>N</td><td></td><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01) [SFBG+07]. The lexicon synset will be transformed to a class. The hyponymy/hypernym relations are mapped to subClassOf/superClassOf relations. The member meronym/holonym relations are mapped to partOf/hasPart. For synonyms we use the logical pattern proposed by Corcho et al. [CR09] and suggested as best practice in the context of this antipattern: The tendency to declare two classes equivalent when in fact their labels simply express synonymy. Finally, the content of the lexicon will be transformed into ontology instances.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>
</td></tr>
<tr><td><b>(UML) Example Solution Ontology</b></td><td>
</td></tr>
<tr><td colspan="2">Continued on next page</td></tr>
</table>

Table 9.3: Pattern for re-engineering a lexicon following the record-based data model into an ontology (continued).

Slot	Value
PROCESS: How to Re-engineer	
General	**Require:** Identification of the relations by using the *record-based* model 1: $entityName \leftarrow$ name of the entity that contains the synsets 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation1 \leftarrow$ *partOf* 6: relate($relation1$,$mainClass$,$mainClass$) 7: $LSynsets \leftarrow$ lexicon synsets 8: **for** $si \in LSynsets$ **do** 9: **if** not alreadyCreatedInstanceFor(si) **then** 10: $Ii \leftarrow$ createInstance(si) 11: setInstanceOfClass(Ii,$mainClass$) 12: **end if** 13: $Synonyms \leftarrow$ synonymsOf(si) 14: **for** $sq \in Synonyms$ **do** 15: SOE(si,sq) 16: **end for** 17: **end for**
Example	**Require:** Identification of the relations by using the *record-based* model 1: $entityName \leftarrow$ name of the entity that contains the synsets 1: $entityName \leftarrow$ BioLexicon 2: $mainClass \leftarrow$ createClass($entityName$) 3: $relation \leftarrow$ *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation2 \leftarrow$ *partOf* 6: relate($relation2$,$mainClass$,$mainClass$) 7: $LSynsets \leftarrow$ lexicon synsets 7: $LSynsets \leftarrow$ [animal cell;cell part;cell;CESP:C16C10.8] 10: $I1 \leftarrow$ createInstance(animal cell) 11: setInstanceOfClass($I1$,$mainClass$) 13: $Synonyms \leftarrow \emptyset \leftarrow$ synonymsOf(animal cell) 10: $I2 \leftarrow$ createInstance(cell part) 11: setInstanceOfClass($I2$,$mainClass$) 13: $Synonyms \leftarrow \emptyset \leftarrow$ usedForTermOf(cell part) 10: $I3 \leftarrow$ createInstance(cell) 11: setInstanceOfClass($I3$,$mainClass$) 13: $Synonyms \leftarrow$ synonymsOf(cell) 13: $Synonyms \leftarrow$ [CESP:C16C10.8] // using the *record-based* model 15: SOE(cell,CESP:C16C10.8)
Time Complexity	$O(n)$

Continued on next page

Table 9.3: Pattern for re-engineering a lexicon following the record-based data model into an ontology (continued).

Slot	Value
Additional Notes	• $LSynsets, Synonyms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is a class already created for a given term. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class. • *synonymsOf* is a function that returns the synonyms of a given synset. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of the following antipattern: The tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Lexicon: $L = \langle LS, LC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $LS \longrightarrow OS:$ $SY \longrightarrow C$ $SR \longrightarrow R \cup S$ $LC \longrightarrow KB:$ $SI \longrightarrow I$ $Lt_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG$^+$07]

9.2.2.2 Pattern for re-engineering a lexicon following the relation-based data model into an ontology.

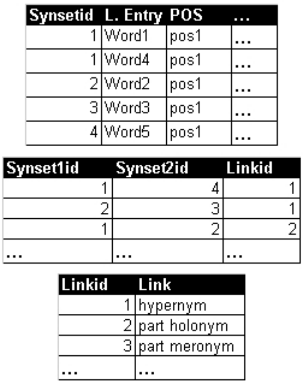
The pattern for re-engineering lexicon, shown in Table 9.4, provides a guide to transform a lexicon into an ontology. The lexicon is modelled with the relation-based data model.

Table 9.4: Pattern for re-engineering a lexicon following the relation-based data model into an ontology.

Slot	Value
General Information	
Name	Pattern for re-engineering a lexicon following the relation-based data model into an ontology.
Identifier	PR-NOR-LXAX-11
Type of Component	Pattern for Re-engineering Non-ontological Resources (PR-NOR)

Continued on next page

Table 9.4: Pattern for re-engineering a lexicon following the relation-based data model into an ontology (continued).

Slot	Value
Use Case	
General	Re-engineering a lexicon following the relation-based data model into an ontology.
Example	Suppose that someone wants to build an ontology based on the BioLexicon. The BioLexicon, and one of its verions is modelled with the relation-based data model.
Pattern for Re-engineering Non-ontological Resources	
INPUT: Resource to be Re-engineered	
General	A non-ontological resource holds a lexicon that follows the relation-based model. A lexicon contains a list of words in a language along with some knowledge of how to use each word. A lexicon may be general or domain-specific; we might have, for example, a lexicon of several thousand common words of English or German, or a lexicon of the technical terms of dentistry in some language. The relation-based data model [Soe95] is a normalized structure, in which relationship types are not defined as fields in a record, but they are simply data values in a relationship record, thus new relationship types can be introduced with ease.
Example	The BioLexicon is a large-scale terminological resource which has been developed to address the needs emerging in text mining efforts in the biomedical domain. This lexicon is available at `http://www.ebi.ac.uk/Rebholz-srv/BioLexicon/biolexicon.html`
Graphical Representation	
General	

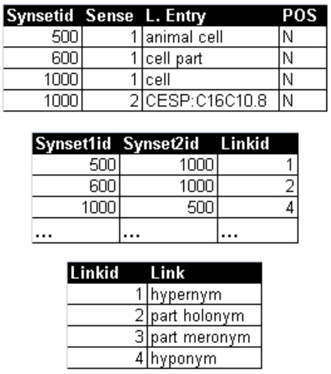
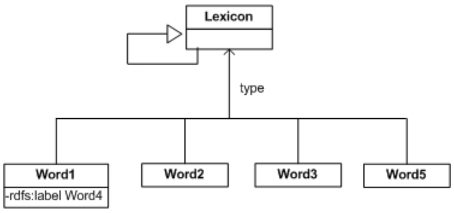
Synsetid	L. Entry	POS	...
1	Word1	pos1	...
1	Word4	pos1	...
2	Word2	pos1	...
3	Word3	pos1	...
4	Word5	pos1	...

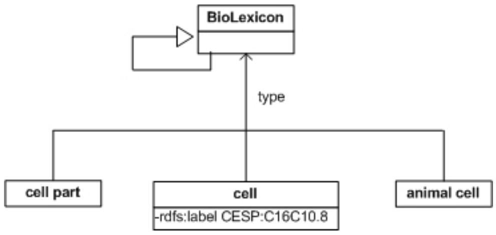
Synset1id	Synset2id	Linkid
1	4	1
2	3	1
1	2	2
...	...	...

Linkid	Link
1	hypernym
2	part holonym
3	part meronym
...	...

Continued on next page

Table 9.4: Pattern for re-engineering a lexicon following the relation-based data model into an ontology (continued).

<table>
<tr><th>Slot</th><th>Value</th></tr>
<tr><td><b>Example</b></td><td>

<table>
<tr><th>Synsetid</th><th>Sense</th><th>L. Entry</th><th>POS</th></tr>
<tr><td>500</td><td>1</td><td>animal cell</td><td>N</td></tr>
<tr><td>600</td><td>1</td><td>cell part</td><td>N</td></tr>
<tr><td>1000</td><td>1</td><td>cell</td><td>N</td></tr>
<tr><td>1000</td><td>2</td><td>CESP:C16C10.8</td><td>N</td></tr>
</table>
<table>
<tr><th>Synset1id</th><th>Synset2id</th><th>Linkid</th></tr>
<tr><td>500</td><td>1000</td><td>1</td></tr>
<tr><td>600</td><td>1000</td><td>2</td></tr>
<tr><td>1000</td><td>500</td><td>4</td></tr>
<tr><td>...</td><td>...</td><td>...</td></tr>
</table>
<table>
<tr><th>Linkid</th><th>Link</th></tr>
<tr><td>1</td><td>hypernym</td></tr>
<tr><td>2</td><td>part holonym</td></tr>
<tr><td>3</td><td>part meronym</td></tr>
<tr><td>4</td><td>hyponym</td></tr>
</table>
</td></tr>
<tr><th colspan="2">OUTPUT: Designed Ontology</th></tr>
<tr><td><b>General</b></td><td>The ontology generated will be based on the lightweight ontology architectural pattern (AP-LW-01) [SFBG$^+$07]. The lexicon synset will be transformed to a class. The hyponymy/hypernym relations are mapped to subClassOf/superClassOf relations. The member meronym/holonym relations are mapped to partOf/hasPart. For synonyms we use the logical pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonymy. Finally, the content of the lexicon will be transformed into ontology instances.</td></tr>
<tr><th colspan="2">Graphical Representation</th></tr>
<tr><td><b>(UML) General Solution Ontology</b></td><td>


</td></tr>
<tr><td><b>(UML) Example Solution Ontology</b></td><td>


</td></tr>
</table>

Continued on next page

Table 9.4: Pattern for re-engineering a lexicon following the relation-based data model into an ontology (continued).

Slot	Value
PROCESS: How to Re-engineer	
General	**Require:** Identification of the relations by using the *relation-based* model 1: $entityName$ ← name of the entity that contains the synsets 2: $mainClass$ ← createClass($entityName$) 3: $relation$ ← *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation1$ ← *partOf* 6: relate($relation1$,$mainClass$,$mainClass$) 7: $LSynsets$ ← lexicon synsets 8: **for** $si \in LSynsets$ **do** 9: **if** not alreadyCreatedInstanceFor(si) **then** 10: Ii ← createInstance(si) 11: setInstanceOfClass(Ii,$mainClass$) 12: **end if** 13: $Synonyms$ ← synonymsOf(si) 14: **for** $sq \in Synonyms$ **do** 15: SOE(si,sq) 16: **end for** 17: **end for**
Example	**Require:** Identification of the relations by using the *relation-based* model 1: $entityName$ ← name of the entity that contains the synsets 1: $entityName$ ← BioLexicon 2: $mainClass$ ← createClass($entityName$) 3: $relation$ ← *subClassOf* 4: relate($relation$,$mainClass$,$mainClass$) 5: $relation2$ ← *partOf* 6: relate($relation2$,$mainClass$,$mainClass$) 7: $LSynsets$ ← lexicon synsets 7: $LSynsets$ ← [animal cell;cell part;cell;CESP:C16C10.8] 10: $I1$ ← createInstance(animal cell) 11: setInstanceOfClass($I1$,$mainClass$) 13: $Synonyms$ ← ∅ ← synonymsOf(animal cell) 10: $I2$ ← createInstance(cell part) 11: setInstanceOfClass($I2$,$mainClass$) 13: $Synonyms$ ← ∅ ← usedForTermOf(cell part) 10: $I3$ ← createInstance(cell) 11: setInstanceOfClass($I3$,$mainClass$) 13: $Synonyms$ ← synonymsOf(cell) 13: $Synonyms$ ← [CESP:C16C10.8] // using the *record-based* model 15: SOE(cell,CESP:C16C10.8)
Time Complexity	$O(n)$

Continued on next page

Table 9.4: Pattern for re-engineering a lexicon following the relation-based data model into an ontology (continued).

Slot	Value
Additional Notes	• $LSynsets, Synonyms$ are lists that do not allow duplicates. • *createClass* is a function that creates a class from a given term. • *relate* is a function that relates two given classes by a given relation. • *alreadyCreatedClassFor* checks if there is an already class created for a given term. • *createInstance* is a function that creates an instance from a given term. • *setInstanceOfClass* is a function that sets up a given instance of a given class. • *synonymsOf* is a function that returns the synonyms of a given synset. • *SOE* is a pattern proposed by Corcho et al. [CR09] suggested as best practice in the context of this antipattern: the tendency to declare two classes equivalent when in fact their labels simply express synonym.
Formal Transformation	
General	Lexicon: $L = \langle LS, LC \rangle$ Ontology: $O = \langle OS, KB \rangle$ Transformation: $LS \longrightarrow OS:$ $SY \longrightarrow C$ $SR \longrightarrow R \cup S$ $LC \longrightarrow KB:$ $SI \longrightarrow I$ $Lt_R \longrightarrow t_R$
Relationships	
Relations to other modelling components	Use the Architectural Pattern: AP-LW-01 [SFBG$^+$07]

9.3 Summary

This chapter has presented our solution to those aspects related to the re-engineering of lexica for building ontologies. This solution addresses some of the limitations identified in the state of art in this area.

First, the chapter reviews the definition of a lexicon, including its components. Then, it provides a formal definition for the lexicon, the identified data models, and implementations for them. Finally, it presents the patterns for re-engineering lexica into ontologies, including those for the TBox and ABox transformation approaches. The time complexity of the TBox transformation algorithm is polynomial $O(n^2)$, whereas that of the ABox transformation algorithm is linear $O(n)$. This set of patterns are used within the method presented in Chapter 6.

The solutions here presented cover contribution **C9**, which partially addresses objective **O3** (see Chapter 3). This contribution is evaluated in Sections 11.1.1 and 11.2.1.

Chapter 10

TECHNOLOGICAL SUPPORT

Our technological support consists in the implementation of (i) NOR$_2$O, a software library that implements the transformation process suggested by the patterns, and (ii) a PR-NOR pattern library that includes the set of patterns for re-engineering non-ontological resources. Our pattern library is available at the ODP portal. In this section we start by presenting the software library (section 10.1) followed by the re-engineering patterns library (section 10.2).

10.1 NOR$_2$O

This section presents NOR$_2$O, a Java library that implements the transformation process suggested by the Patterns for Re-engineering Non-ontological Resources (PR-NOR), which are described in Chapters 7, 8 and 9. The library performs the ETL process[1] for transforming the non-ontological resource components into ontology terms. A high level conceptual architecture diagram of the modules involved is shown in Figure 10.1.

Figure 10.1 depicts the modules of the PR-NOR software library: `NOR Connector`, `Transformer`, `Semantic Relation Disambiguator`, `External Resource Service`, and `OR Connector`. In the following sections these modules are described in detail. For illustrating the modules, the example of the transformation of the ASFA thesaurus[2] into an ontology schema[3] is provided.

10.1.1 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.

[1] Extract, transform, and load (ETL) of legacy data sources, is a process that involves: (1) extracting data from the outside resources, (2) transforming data to fit operational needs, and (3) loading data into the end target resources [KC04].

[2] `http://www4.fao.org/asfa/asfa.htm`

[3] `http://droz.dia.fi.upm.es/ontologies/asfa.owl`

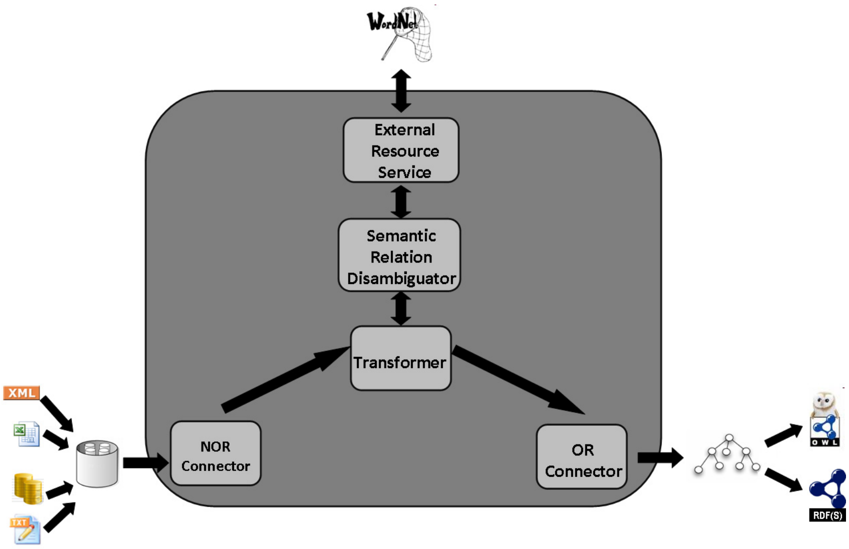


Figure 10.1: Modules of the NOR$_2$O software library.

This module utilizes an XML configuration file for describing the NOR. Figure 10.2 shows the graphical representation of the NOR connector XSD file, including the following main sections:

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

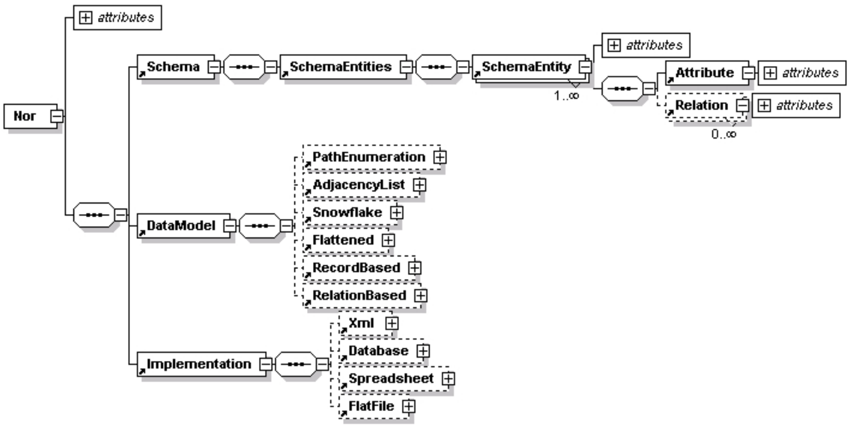


Figure 10.2: Graphical representation of the NOR Connector XSD file.

An example of the XML configuration file is presented in Listing 10.1. The Figure shows how the file describes a thesaurus. The thesaurus has two schema entities, *Term* and *NonPreferredTerm*, is modelled following the record-based data model and is implemented in XML.

Listing 10.1: NOR Connector configuration file example

```
<Nor type=”Classification␣Scheme” name=”cepa94”>
 <Schema>
  <SchemaEntities>
         <SchemaEntity name=”CSItem”>
          <Attribute name=”CSIdentifier”
                                        valueFrom=”cepa.CodeNumber”
                                        type=”string”/>
                <Attribute name=”CSName”
                                        valueFrom=”cepa.DescriptionEnglish”
                                        type=”string”/>
                <Relation name=”subType”
                                        using=”PathEnumeration”
                                        destination=”CSItem”/>
                <Relation name=”superType”
                                        using=”PathEnumeration”
                                        destination=”CSItem”/>
         </SchemaEntity>
        </SchemaEntities>
 </Schema>
 <DataModel>
  <PathEnumeration>
         <PathEntity>cepa</PathEntity>
         <PathSeparator>.</PathSeparator>
          <PathField>CodeNumber</PathField>
         </PathEnumeration>
 </DataModel>
 <Implementation>
  <Database>
         <Dbms>MSACCESS</Dbms>
         <Name>cepa94</Name>
         <Username></Username>
         <Password></Password>
         <Host></Host>
         <Port></Port>
        </Database>
 </Implementation>
</Nor>
```

10.1.2 Transformer

This module performs the transformation suggested by the patterns by implementing the sequence of activities included in the patterns. The module transforms the NOR elements, loaded by the `NOR Connector` module, into internal model representation elements. It also 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, objectproperties, dataproperties and individuals). Additionally, it indicates the transformation approach, e.g., TBox, ABox or Population.

Figure 10.3 shows the graphical representation of the PRNOR XSD file. Two examples of the XML configuration file are shown in Listings 10.2 and 10.3.

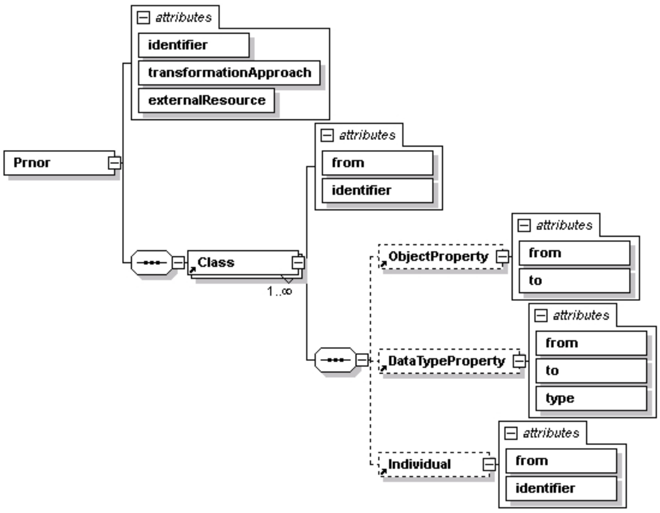


Figure 10.3: Graphical representation of the PRNOR XSD file.

Listing 10.2 indicates that the pattern follows the TBox transformation approach and that it 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.

Listing 10.2: PRNOR Connector configuration file example - Classification Scheme

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

Listing 10.3 indicates that the pattern follows the TBox transformation approach and that 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 disambiguation.

Listing 10.3: PRNOR Connector configuration file example - Thesaurus

```
<Prnor identifier=”PR−NOR−TSTX−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>
```

10.1.3 Semantic Relation Disambiguator

This module is in charge of obtaining the semantic relation between two NOR elements. Basically, the module receives two NOR elements from the `Transformer` module and returns the semantic relation between them. First the module verifies whether it can obtain the *subClassOf* relation by identifying attribute adjetives[4] within the two given elements of the resource. If this is not the case, then the module connects the external resource through the `External Resource Service` module to get the relation.

The TBox transformation approach 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. Thus, patterns that follow the TBox transformation approach must make explicit 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.

Algorithm 1, presented in Section 6.4, describes how to make explicit the semantics of the relations in the NOR terms.

It is worth mentioning that, when asserting the *partOf* relation the algorithm takes advantage of the use of the `PartOf content pattern`[5] to guarantee that the OWL code generated follows common practices in Ontological Engineering.

10.1.4 External Resource Service

The `External Resource Service` is in charge of interacting with external resources for obtaining the semantic relations between two NOR elements. At this

[4] 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, the attributive adjective usually precedes the noun in simple phrases, but often follows the noun when the adjective is modified or qualified by a phrase acting as an adverb.

[5] `http://ontologydesignpatterns.org/wiki/Submissions:PartOf`

moment the module interacts with WordNet. We are now implementing the access to DBpedia[6] because of the reasons explained in Section 6.4.

10.1.5 OR Connector

The `Ontological Resource (OR) Connector` generates the ontology in OWL Lite. To this end, this module relies on the OWL API[7]. It also utilizes an XML configuration file for describing the ontology to be generated. Figure 10.4 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, the URI, the file, and the implementation language of the ontology. Additionally, and in the case we want to populate an available ontology, this section indicates if the ontology already exists. Finally, this module includes the provenance information of the non-ontological resource and uses the NoRMV metadata vocabulary, described in Section 5.2.

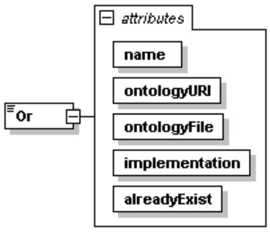


Figure 10.4: Graphical representation of the OR XSD file.

An example of the XML configuration file is shown in Listing 10.4. The Figure indicates that the ontology generated will be stored in the *asfa.owl* file, that its name will be *asfa ontology*, and that it will be implemented in OWL.

Listing 10.4: 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” separator=”#”>
</Or>
```

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 and include other types of NORs, data models, and implementations in a simple way, as well as to exploit other external resources for making explicit the hidden semantics in the relations of the NOR terms.

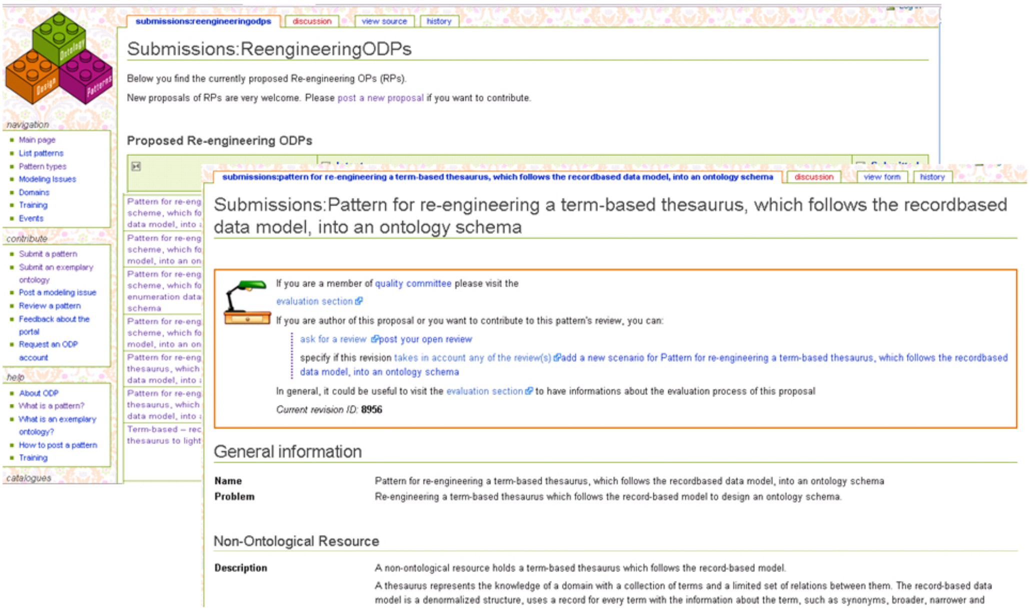
[6] `http://dbpedia.org/`

[7] `http://owlapi.sourceforge.net/`

10.2 PR-NOR Library at the ODP Portal

Ontologydesignpatterns.org (hereafter ODP Portal) is a Semantic Web portal dedicated to ontology design best practices for the Semantic Web, with a particular focus on ODPs. The ODP Portal software is based on Media Wiki[8], Semantic Media Wiki (SMW)[9], Semantic Forms (SF)[10], and other extensions[11]. This portal is maintained by the Semantic Technology Laboratory[12] at the *Consiglio Nazionale delle Ricerche*, in Rome, Italy. The ODP Portal is targeted at users interested in best practices for ontology design and ontology engineering. ODPs encode ontology engineering best practices to design high-quality ontologies. Currently the ODP portal supports the lifecycle of the following ODP types: Content ODPs, Re-engineering ODPs, Alignment ODPs, Logical ODPs, Architectural ODPs, and Lexico-syntatic ODPs.

The PR-NORs proposed in this thesis fit in the Re-engineering ODPs category available in the ODP Portal. Figure 10.5 shows a screenshot of the list of Re-engineering ODPs included in the ODP Portal. Also included in the portal are the overall set of patterns described in Chapters 7, 8, and 9.


Ontology
Design
Patterns
navigation
Main page
List patterns
Pattern types
Modeling Issues
Domains
Training
Events
contribute
Submit a pattern
Submit an exemplary ontology
Post a modeling issue
Review a pattern
Feedback about the portal
Request an ODP account
help
About ODP
What is a pattern?
What is an exemplary ontology?
How to post a pattern
Training
catalogues
submissions:reengineeringodps
discussion
view source
history
Pattern for re-eng scheme, which fo data model, into
Pattern for re-eng scheme, which fo model, into an on
Pattern for re-eng scheme, which fo enumeration data schema
Pattern for re-eng scheme, which fo model, into an on
Pattern for re-eng thesaurus, which data model, into
Pattern for re-eng thesaurus, which data model, into
Term-based – rec thesaurus to light
submissions:pattern for re-engineering a term-based thesaurus, which follows the recordbased data model, into an ontology schema
discussion
view form
history

Submissions:ReengineeringODPs

Below you find the currently proposed Re-engineering OPs (RPs).

New proposals of RPs are very welcome. Please post a new proposal if you want to contribute.

Proposed Re-engineering ODPs

Submissions:Pattern for re-engineering a term-based thesaurus, which follows the recordbased data model, into an ontology schema

If you are a member of quality committee please visit the evaluation section

If you are author of this proposal or you want to contribute to this pattern's review, you can:

ask for a review post your open review

specify if this revision takes in account any of the review(s) add a new scenario for Pattern for re-engineering a term-based thesaurus, which follows the recordbased data model, into an ontology schema

In general, it could be useful to visit the evaluation section to have informations about the evaluation process of this proposal

Current revision ID: **8956**

General information

Name	Pattern for re-engineering a term-based thesaurus, which follows the recordbased data model, into an ontology schema
Problem	Re-engineering a term-based thesaurus which follows the record-based model to design an ontology schema.

Non-Ontological Resource

Description	A non-ontological resource holds a term-based thesaurus which follows the record-based model. A thesaurus represents the knowledge of a domain with a collection of terms and a limited set of relations between them. The record-based data model is a denormalized structure, uses a record for every term with the information about the term, such as synonyms, broader, narrower and

Figure 10.5: Re-engineering ODPs at the ODP Portal

[8] http://www.mediawiki.org

[9] http://www.semantic-mediawiki.org

[10] http://www.mediawiki.org/Extension:SemanticForm

[11] The full list of the extensions can be found at http://ontologydesignpatterns.org/wiki/Special:Version

[12] http://stlab.istc.cnr.it/stlab/

Table 10.1: PR-NOR Library web accesses

Pattern	Release Date	Total
PR-NOR-CLTX-02	12-October-2009	1578
PR-NOR-TSTX-01	12-October-2009	1421
PR-NOR-CLTX-01	12-October-2009	1094

One of the goals of the PR-NOR Library is to become a community-accepted re-engineering pattern library for transforming resources into ontologies. Table 10.1 shows the total number of access to the three most visited patterns during a period of 12 months starting in October 2009.

The most visited patterns are (1) the pattern for re-engineering a classification scheme following the adjacency list data model into an ontology schema (PR-NOR-CLTX-02); (2) the pattern for re-engineering a thesaurus following the recordbased data model into an ontology schema (PR-NOR-TSTX-01); and (3) the pattern for re-engineering a classification scheme following the path enumeration data model into an ontology schema (PR-NOR-CLTX-01).

10.3 Summary

This chapter has presented the solution we provide for the aspects related to the technologial support for re-engineering non-ontological resources into ontologies. It has also addressesed some of the limitations identified in the state of art in this area.

Regarding the goals and contributions presented in Section 3.1 and Section 3.2 respectively, Section 10.1 presents the NOR$_2$O (part of contribution **C10**, which addresses objective **O4**) and section 10.2 presents the PR-NOR library included in the ODP portal (part of contributions **C7**,**C8**, and **C9**, which address objective **O3**).

Chapter 11

EVALUATION

This chapter presents the evaluation of the contributions of this thesis. The main contributions are (1) a set of methodological guidelines for reusing non-ontological resources when building ontologies, and (2) technological support for re-engineering, i.e., the PR-NOR pattern library and NOR$_2$O. Therefore, the evaluation covers both the methodological and the technological aspects.

Regarding the methodological guidelines, we evaluate with the following criteria: understandability, applicability, and usability of the guidelines.

As for the technological support, we evaluate with the following criterion: quality of the software library and patterns. Thus, quality is assessed by calculating the similarity of the ontologies generated against gold standard ontologies, as well as the applicability and usability of the technology.

Table 11.1 gathers the aforementioned criteria used for evaluating the contributions through the set of experiments.

The evaluation is divided into two parts. First, Section 11.1 describes the evaluation of the methodological aspects related to the reuse and re-engineering of non-ontological resources for building ontologies. Then, Section 11.2 presents the evaluation of the technological support focused on the PR-NOR pattern library and the NOR$_2$O software library.

Table 11.1: Evaluation criteria

Contribution	Criteria	Section
Method for Reusing NORs	• Understandability of the guidelines. • Applicability of the guidelines. • Usability of the guidelines.	Section 11.1
Method for Re-engineering NORs	• Understandability of the guidelines. • Applicability of the guidelines. • Usability of the guidelines.	Section 11.1
PR-NOR pattern library	• Quality of the patterns. • Understandability of the patterns. • Usability of the patterns.	Section 11.2
NOR$_2$O software library	• Quality of the software library. • Usability of the software library. • Applicability the NOR$_2$O.	Section 11.2

11.1 Evaluation of the Methodological Guidelines

This section presents three experiments with the objective of evaluating the understandability, applicability and usability of the methodological contributions. The first one was carried out with students attending a Master Course at the UPM. The other two are based on real case scenarios within the SEEMP[1] and mIO![2] projects.

11.1.1 Understandability, Applicability and Usability of the Methodological Aspects of Re-engineering within a Master Course

This example refers to the manual transformation of an excerpt from a thesaurus following the guidelines and the proposed set of patterns. The purpose is to assess the understandability, applicability and usability of the methodological guidelines of the re-engineering process and of the set of patterns for carrying out the NOR Re-engineering into an OWL ontology.

11.1.1.1 Settings

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

[1] http://www.seemp.org/

[2] http://www.cenitmio.es/

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

The participants had to build manually a conceptual model from a particular resource, analysing the methodological guidelines and the set of patterns. They had 30 minutes for generating the conceptual model and had to work on an excerpt of twenty terms of the ETT thesaurus[3].

11.1.1.2 Execution

The experiment was executed in four phases:

1. The participants were provided with the proposed guidelines.
2. The participants were organized in groups of two.
3. The groups analysed the methodological guidelines and the set of patterns in order to carry out the NOR re-engineering process. They generated manually a conceptual model.
4. The participants 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 student 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. The groups soon found out that they were dealing with a thesaurus modelled following the record-based data model and implemented in XML.

NOR Transformation. Within this activity the groups searched the ODP portal for a suitable PR-NOR, taking into account the following criteria: (1) the resource type: thesaurus; (2) the resource data model: record-based model; and (3) the selected transformation approach: the TBox transformation. Then thet chose as the most appropriate pattern the PR-NOR-TSTX-01, selected by all the participants. Finally, all the groups followed the procedure suggested by the pattern for creating the conceptual model manually. Each thesaurus term was mapped to a class. For making explicit the semantics of the BT, NT relations among thesaurus terms, the participants checked whether they could get the *subClassOf* relation by identifying attribute adjetives[4]. If they could not, they searched the WordNet web site. When

[3] `http://droz.dia.fi.upm.es/master/rd/homework/resources/ett.xml`

[4] 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.

the query results were empty, they related the terms to the default relation (see Algorithm 1 in Section 6.4). When they had to deal with a thesaurus, for the BT/NT relation, we recommended using the *subClassOf* relation by default.

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

11.1.1.3 Collecting results

We proposed the following questionnaire to the participants for collecting some empirical data.

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. Do you find these guidelines useful?

11.1.1.4 Findings and observations

Table 11.2 presents the 34 answers to the questionnaire. As a general conclusion we can state that the participants did not seem to find any problems regarding the use and understanding of each of the activities and tasks identified in the methodological guidelines.

Table 11.2: Answers to the proposed questionnaire

Questions	Answers
Q1.	Ninety-seven percent of the participants indicated that the guidelines were well explained.
Q2.	Eighty-eight percent of the participants considered that the guidelines need no more details; however twelve percent explained that they would welcome the improvement in 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 thought that the techniques and patterns to execute each activity of the guidelines were sufficient.
Q4.	Eighty-five percent of the participants suggested including more examples of how to use the proposed guidelines and what results were expected.
Q5.	One hundred percent of the participants thought that the guidelines were useful and also necessary.

From the comments received on this experiment, we can conclude that the methodological guidelines seem to be useful and understandable.

11.1.2 Understandability, Applicability and Usability of the Method for Reuse and Re-engineering within the SEEMP Project

In order to evaluate the understandability, applicability and usability of the methodological contributions related to the reuse and re-engineering of NORS in a complex ontology engineering setting, we conducted an experiment in a real case scenario within the SEEMP Project.

The main objective of this project was to develop an interoperable architecture for public e-Employment services (PES). The resultant architecture consisted of (1) a Reference Ontology, the core component of the system, that acts as a common "language" in the form of a set of controlled vocabularies that describes the details of a job posting; (2) a set of Local Ontologies, each PES uses its own Local Ontology, which describes the employment market in its own terms; (3) a set of mappings between each Local Ontology and the Reference Ontology; and (4) a set of mappings between the PES schema sources and the Local Ontologies.

In the following sections we describe the application of our methodological guidelines for reusing and re-engineering non-ontological resources when building an occupation ontology.

11.1.2.1 Reusing non-ontological resources

This section presents the application of the Method for Reusing Non-Ontological Resources within the SEEMP project. It shows the process we followed for selecting the non-ontological resources to be reused when building the occupation domain ontology.

Activity 1. Search non-ontological resources

Following the suggestions of some domain experts, we searched for the occupation classifications at (1) the Ramon Eurostat Portal[5], (2) the ONET Web site[6], and (3) the companies the project partners. Thus, we found the following classifications:

- Standard Occupational Classification System (SOC).
- International Standard Classification of Occupations (ISCO-88).
- International Standard Classification of Occupations, for European Union purposes, ISCO-88 (COM).
- Occupational Information Network (ONET).
- EURES[7] proprietary occupation classification.

[5] `http://ec.europa.eu/eurostat/ramon/`

[6] `http://online.onetcenter.org/`

[7] `http://www.eurodyn.com/`

Activity 2. Assess the set of candidate non-ontological resources

The goal of this activity was to assess the set of candidate non-ontological resources. Experts of the occupation domain, software developers and ontology practitioners carried out this activity taking as input the set of candidate non-ontological resources.

Task 1. Extract lexical entries

Within this task we extracted the lexical entries of the aforementioned occupation classifications. We developed an *ad-hoc* extraction tool for performing automatically the extraction task.

Task 2. Calculate precision

Since we were dealing with occupations related to the IT domain, it was impossible to cover all the IT domain occupations already identified in the Ontology Requirements Specification Document. Thus, we used a constant *K* that represents the complete set of IT domain occupations. Next, we present the precision for each occupation classification.

$$Precision = \frac{card\{\{NORLexicalEntries\} \cap \{ORSDTerminology\}\}}{card\{NORLexicalEntries\}}$$

- $SOCPrecision = \frac{6 \cap K}{26162} = \frac{6}{26162} = 0.0002$
- $ISCO - 88Precision = \frac{9 \cap K}{544} = \frac{9}{544} = 0.0165$
- $ISCO - 88COMPrecision = \frac{9 \cap K}{520} = \frac{9}{520} = 0.0173$
- $ONETPrecision = \frac{21 \cap K}{1167} = \frac{21}{1167} = 0.0179$
- $EURESPrecision = \frac{89 \cap K}{355} = \frac{89}{355} = 0.2507$

Task 3. Calculate coverage

Again, since we were dealing with the occupations related to the IT domain, it was impossible to cover all the IT domain occupations in the ORSD. Thus, we used a constant *K* that represents the complete set of IT domain occupations. Next, we present the coverage for each occupation classification.

$$Coverage = \frac{card\{\{NORLexicalEntries\} \cap \{ORSDTerminology\}\}}{card\{ORSDTerminology\}}$$

- $SOCPrecision = \frac{6 \cap K}{K} = \frac{6}{K}$
- $ISCO - 88Precision = \frac{9 \cap K}{K} = \frac{9}{K}$
- $ISCO - 88COMPrecision = \frac{9 \cap K}{K} = \frac{9}{K}$
- $ONETPrecision = \frac{21 \cap K}{K} = \frac{21}{K}$
- $EURESPrecision = \frac{89 \cap K}{K} = \frac{89}{K}$

Task 4. Evaluate the Consensus

It was important for the project that resources focused on the current European reality, because the user partners involved in SEEMP are European, and the outcoming prototype has to be validated in European scenarios. Thus, domain experts confirmed whether the resources were built with the consensus of the European community or not. They also explained that ISCO-88(COM) and EURES proprietary occupation classification contains terminology that had already reached a consensus.

Table 11.3 summarizes all the information of each non-ontological resource.

Table 11.3: Assessment table for SEEMP Occupation Standards

NOR	Precision	Coverage	Consensus
SOC	0.0002	6 / K	no
ISCO-88	0.0165	9 / K	no
ISCO-88 COM	0.0173	9 / K	yes
ONET	0.0179	21 / K	no
EURES	0.2507	89 / K	yes

Activity 3. Select the most appropriate non-ontological resources

Following Table 11.3 we selected a non-ontological resource, the EURES proprietary occupation classification.

We followed the same process for selecting the non-ontological resources when building the remaining ontologies. We provide a table (see Table 11.4) that summarizes the selection of standards, codes, and classification accomplished for building every domain ontology.

11.1.2.2 Re-engineering non-ontological resources

In this section we present the application of the Method for Re-engineering Non-Ontological Resources within the SEEMP project. Once we select the non-ontological resource, we have to transform it into an ontology. Next, we describe the process of generating an Occupation Ontology from the EURES proprietary occupation classification.

Activity 1. Non-ontological resource reverse engineering

In this activity we gathered documentation on the EURES occupation classification from the 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. The EURES occupation classification is modelled following the snowflake data model and is implemented in a MS Access database.

Table 11.4: Standards, codes and classifications reused

Domain	**Candidate Standards/-Classifications**	**Selected Standards /Classifications**	**Justification**
Economic Sector	ISIC, NACE, NAICS	NACE	Best Coverage and European scope
Education Fields	ISCED 97, FOET	FOET	Best Coverage and European Scope
Education Levels	ISCED 97	ISCED 97	Worldwide scope, widely accepted
Currency	Pacific Exchange, ISO 4217, WordAtlas	ISO 4217	Worldwide scope, widely accepted
Geographic	ISO 3166, Regions of the World	ISO 3166	Worldwide scope, widely accepted
Language	ISO 639	ISO 639	Worldwide scope, widely accepted
Language Levels	CEFR	CEFR	European scope, widely accepted
Driving License	EU Driving License	EU Driving License	European legislation
Skills	EURES	EURES	Coverage and European scope
Contract types	LE FOREM proprietary classification, ARL proprietary classification	Mix of both classifications	Aceptable Coverage in SEEMP scope
Work condition	LE FOREM proprietary classification	LE FOREM proprietary classification	Aceptable Coverage in SEEMP scope

Activity 2. Non-ontological resource transformation

Within this activity we carried out the following tasks:

1. We identified the transformation approach, the TBox transformation, i.e., transforming the resource content into an ontology schema.

2. Then, we searched 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.

3. The most appropriate pattern found for this case was 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.

Activity 3. Ontology forward engineering

WSML[8] is the ontology implementation language used in the SEEMP project. Because of the number of occupations of the EURES classification, it was not

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

practical to create the ontology manually. Therefore, we created an *ad-hoc* wrapper, implemented in Java, that reads the data from the resource implementation and automatically creates the corresponding classes and relations of the new ontology following the suggestions given by the pattern for re-engineering NORs and the conceptual model.

We followed this process for all the resources identified, being the patterns used those presented in Table 11.5.

Table 11.5: Resources transformed in the SEEMP project

Resource	Type	Data Model	Implementation	Pattern used
NACE	Classification Scheme	Path enumeration	Database	PR-NOR-CLTX-01
FOET	Classification Scheme	Path enumeration	Database	PR-NOR-CLTX-01
ISCED 97	Classification Scheme	Adjacency list	Database	PR-NOR-CLTX-02
ISO 4217	Classification Scheme	Snowflake	XML	PR-NOR-CLAX-12
ISO 3166	Classification Scheme	Snowflake	XML	PR-NOR-CLAX-12
ISO 639	Classification Scheme	Snowflake	XML	PR-NOR-CLAX-12
CEFR	Classification Scheme	Proprietary model	Proprietary format	
EU Driving License	Classification Scheme	Snowflake	Proprietary format	
EURES Skill	Classification Scheme	Path enumeration	Database	PR-NOR-CLTX-01
LE FOREM Contracts	Proprietary classification	Proprietary model	Proprietary format	

11.1.2.3 Analysis of the applicability of the method

The SEEMP Reference Ontology (SEEMP RO) was developed following the method for reusing and re-engineering non-ontological resources. It is composed of thirteen modular ontologies: *Competence, Compensation, Driving License, Economic Activity, Education, Geography, Job Offer, Job Seeker, Labour Regulatory, Language, Occupation, Skill, and Time*. The main subontologies are the *Job Offer* and *Job Seeker*, which are intended to represent the structure of a job posting and a CV respectively. While these main two subontologies were built taking as a starting point some HR-XML recommendations, the others derived from some available international standards (like NACE, ISCO-88 (COM), FOET, etc.), Employment Services classifications and international codes (like ISO 3166, ISO 6392, etc.) that best fitted the European requirements. Figure 11.1 presents these thirteen modular ontologies (each ontology is represented by a triangle), ten of which were obtained after re-engineering the standard/classification. The SEEMP Reference Ontology is available at `http://oeg-upm.net/index.php/en/ontologies/99-hrmontology`.

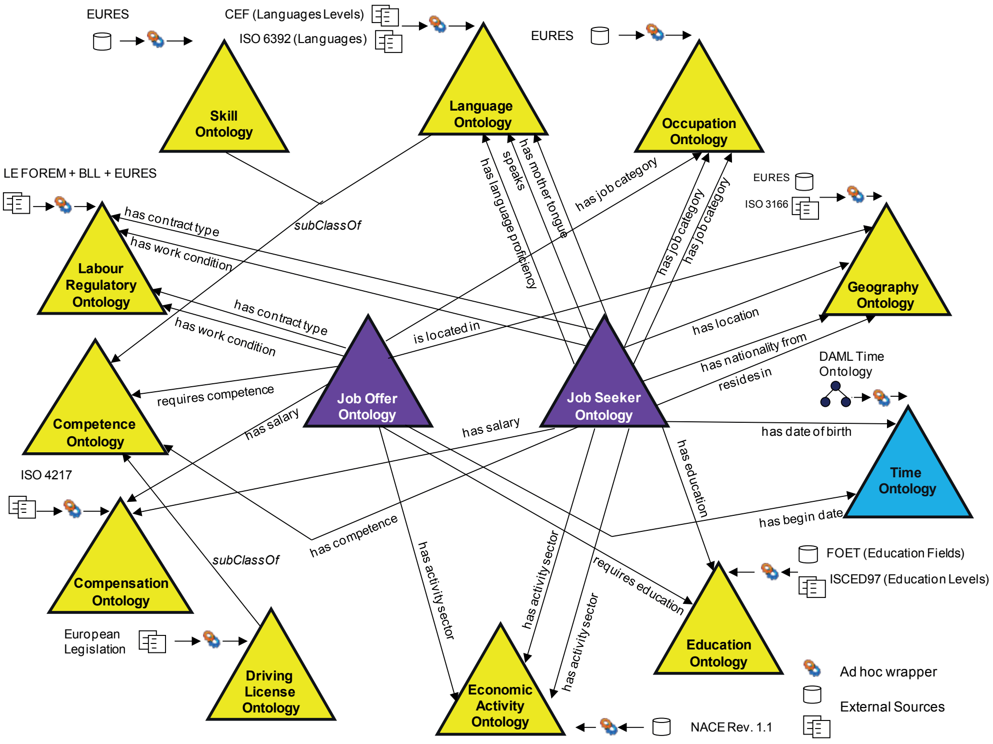


Figure 11.1: SEEMP Reference Ontology

In order to illustrate the dimension of the ontology and the ontological engineers' efforts required to build it, some statistical data are shown in Table 11.6.

Table 11.6: SEEMP Reference Ontology statistical data

Ontology	**Concepts**	**Attributes**	**Axioms**	**Instances**	**Efforts (man.months)**
SEEMP RO	1985	315	1037	1449	6

Our experience in SEEMP has served us to demonstrate that the approach of building ontologies by reusing and re-engineering non-ontological resources already agreed upon allows building ontologies faster, with less resources, and with an immediate consensus. This approach permits making explicit the knowledge implicitly coded in organization models and standards. By building ontologies in this fashion, we facilitate that ontologies become reference ontologies in their respective domains.

With respect to the application of the Method for Reuse and Re-engineering, this was especially useful for guiding the steps of the ontological engineers involved since this method provides detailed and sufficient guidelines. In addition, the existence of a well-defined and structured process for building the ontology network in the e-employment domain eased the planning, coordination and communication with other non-Semantic Web members of the development team, which in turn helped to convey reliability to the final result.

11.1.3 Understandability, Applicability and Usability of the Method within the *mIO!* Project

The evaluation of the understandability, applicability and usability of the methodological contributions for reusing and re-engineering NORs, including the PR-NOR library, were also validated in an experiment in a real case scenario within the context of the *mIO!* Spanish project[9].

The main objective of the *mIO!* project is to develop ubiquitous services in an intelligent environment, adapted to every user and its context by means of mobile interfaces. The project relies on ontologies for modelling the knowledge.

The following sections describe the application of our methodological guidelines for reusing and re-engineering non-ontological resources when building a geographical ontology, which includes continents, countries, and regions.

11.1.3.1 Reusing non-ontological resources

This section describes the activities carried out for reusing non-ontological resources.

Activity 1. Search non-ontological resources
Following some of the suggestions made by the domain experts, we searched geographical location resources on highly reliable Websites. Next, we list the geographic location classifications:

- ISO 3166[10] Maintenance agency (ISO 3166/MA) ISO's focal point for country codes.
- Guide to regions of the World[11]
- Regions of the World[12]

Activity 2. Assess the set of candidate non-ontological resources
Once we had the set of candidate non-ontological resources, we needed to assess them according to the following criteria: precision, coverage, consensus, and quality of the resources.

[9] `http://www.cenitmio.es/`

[10] `http://www.iso.org/iso/en/prods-services/iso3166ma/index.html`

[11] `http://www.countriesandcities.com/regions/`

[12] `http://park.org/Regions/`

Task 2.1 Extract lexical entries

Within this task we extracted the lexical entries of the aforementioned geographic location classifications. For this purpose, we used TreeTagger[13], a syntactic annotator.

Task 2.2 Calculate precision

It was impossible to cover all the geographic locations in the ORSD. Thus, we used a constant *K* that represents the complete set of geographical locations. Next, we present the precision for each geographic location classification.

$$Precision = \frac{card\{\{NORLexicalEntries\} \cap \{ORSDTerminology\}\}}{card\{NORLexicalEntries\}}$$

- $ISO3166 = \frac{195 \cap K}{200} = \frac{195}{200} = 0.975$
- $GuidetoregionsoftheWorld = \frac{102 \cap K}{193} = \frac{102}{193} = 0.528$
- $RegionsoftheWorld = \frac{110 \cap K}{154} = \frac{110}{154} = 0.714$

Task 2.3 Calculate coverage

Again, it was impossible to cover all the geographic locations in the ORSD. Thus, we used a constant *K* that represents the complete set of geographic locations. Next, we present the coverage for each geographic location classification.

$$Coverage = \frac{card\{\{NORLexicalEntries\} \cap \{ORSDTerminology\}\}}{card\{ORSDTerminology\}}$$

- $ISO3166 = \frac{195 \cap K}{K} = \frac{195}{K}$
- $GuidetoregionsoftheWorld = \frac{102 \cap K}{K} = \frac{102}{K}$
- $RegionsoftheWorld = \frac{110 \cap K}{K} = \frac{110}{K}$

Task 2.4 Evaluate the consensus

It was important for the project that resources focused on the current worldwide reality, because the outcoming prototype will be validated by users. Thus, domain experts evaluated whether the resource was built with the consensus of the worldwide community or not. They confirmed that ISO 3166 has the full consensus of the community, whereas the other resources have not.

Task 2.5 Evaluate the quality

In this case, domain experts evaluated whether the resource was built with an acceptable level of quality. They confirmed that ISO 3166 has an acceptable level of quality.

[13] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

Task 2.6 Build the assessment table

Table 11.7 summarizes all the information related to each non-ontological resource.

Table 11.7: Assessment table for the *mIO!* geographical locations

NOR	Precision	Coverage	Consensus	Quality
ISO 3166	0.975	195 / K	yes	yes
Guide to regions of the World	0.528	102 / K	no	no
Regions of the World	0.714	110 / K	no	no

Activity 3. Select the most appropriate non-ontological resources

According to Table 11.7 we selected the following non-ontological resource: ISO 3166.

11.1.3.2 Re-engineering non-ontological resources

This section presents the application of the Method for Re-engineering Non-Ontological Resources within the *mIO!* project. Once we have the non-ontological resource selected, the ISO 3166, we had to transform it into an ontology. Next, we describe the process of generating a Geographical Location Ontology.

Activity 1. Non-ontological resource reverse engineering

In this activity we gathered documentation about ISO 3166 from its website. From this documentation we extracted the schema of the classification scheme, which consists of one entity *ISO_31661_Entry*. Since the data model was not available in the documentation, it was necessary to extract it for the resource implementation itself. ISO 3166 is modelled following the snowflake data model and implemented in XML.

Activity 2. Non-ontological resource transformation

In this activity we carried out the following tasks:

1. We identified the transformation approach, the ABox transformation, i.e., the transformation of the resource schema into an ontology schema, and the resource content into ontology instances.

2. Then we searched our local pattern repository for a suitable pattern to re-engineer NORs, taking into account the transformation approach (ABox transformation), the non-ontological resource type (classification scheme), and the data model (snowflake data model) of the resource.

3. The most appropriate pattern for this case is the PR-NOR-CLAX-12 pattern. This pattern takes as input a classification scheme modelled with a snowflake data model.

4. Finally, we followed the procedure defined by the pattern selected for transforming the resource components into ontology elements.

Activity 3. Ontology forward engineering
In this activity we formalized and implemented the ontology in OWL. The ontology is available at `http://droz.dia.fi.upm.es/ontologies/`.

11.1.3.3 Analysis of the applicability of the method

The network of ontologies of the *mIO!* project was developed following the NeOn Methodology [SF10]. This ontology is composed of eleven modular ontologies: *Provider*, *Service*, *Source*, *Geographical Location*, *Environment*, *Time*, *Device*, *User*, *Network*, *Interface*, and *Role*. Only the geographical location ontology was built according to the method for reusing and re-engineering non-ontological resources. The other ontologies were built by reusing available ontologies or modules.

Figure 11.2 presents the *mIO!* ontology network and includes the location subontology. The ontology network is available at `http://oeg-upm.net/index.php/en/ontologies/82-mio-ontologies`

In order to illustrate the dimension of the ontology and the efforts required by the ontological engineers to build it, we outline some data in Table 11.8.

Table 11.8: *mIO!* Ontology statistical data

Ontology	Concepts	Attributes	Axioms	Instances	Efforts (man.months)
mIO! Ontology	432	276	154	120	3

Our experience in *mIO!* has served us to demonstrate that the approach of building ontologies by reuse and re-engineering non-ontological resources already agreed-upon allows building ontologies faster, with less resources, and with consensus. With respect to the application of the Method for Reuse and Re-engineering, this was especially useful for guiding the steps of the ontological engineers involved since the method provides detailed and sufficient guidelines.

11.1.4 Summary

As a conclusion of this section we can state that the experiment (Section 11.1.1), and the application of the methodological guidelines within the SEEMP and mIO! projects (Sections 11.1.2 and 11.1.3) verify hypothesis **H1**, that is, the reuse of

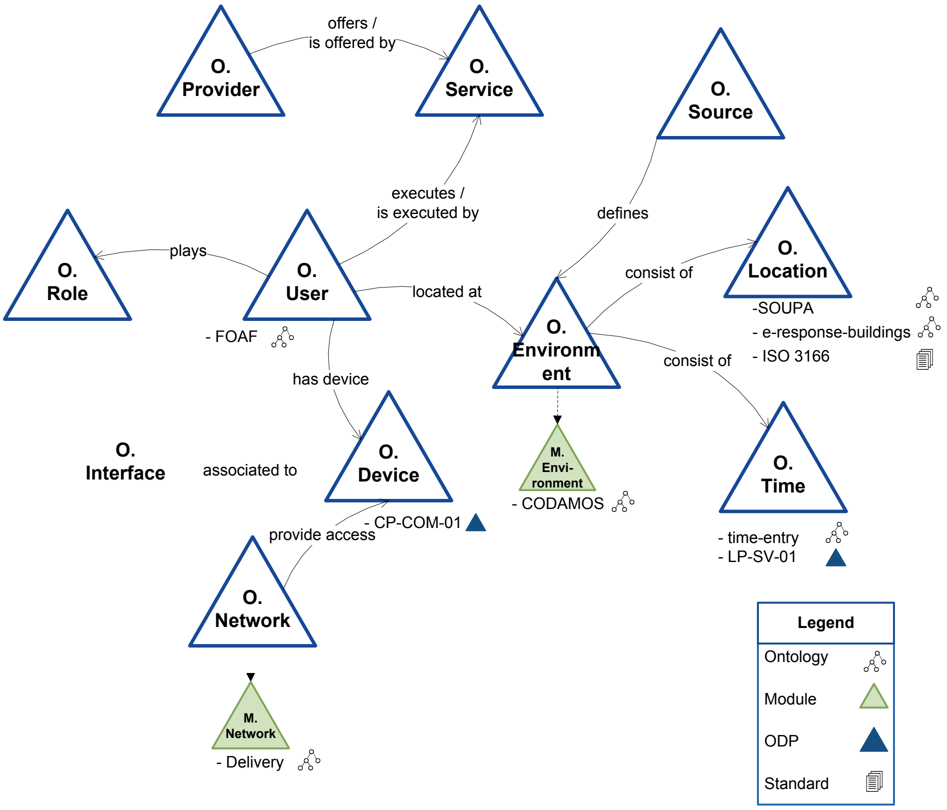


Figure 11.2: *mIO!* Ontology Network

non-ontologi-cal resources that have reached some degree of consensus in a community permits the development of ontologies in an easier and quicker fashion; hypothesis **H2**, that is possible to define a unified method for transforming non-ontological resources into ontologies independently (1) of the type, data model, or implementation of the resource, and (2) of the target ontology to be generated, i.e., ontology schema (TBox), ontology (TBox+ABox), or ontology instances (ABox); and hypothesis **H4**, that the set of patterns for re-engineering are independent of the domain of the resources, that is, the patterns can be used to build ontologies in different domains. Furthermore, this method is really valuable for guiding engineers that do not have any previous experience in building a huge ontology network, especially if the network needs to be solidly grounded in NORs.

11.2 Evaluation of the Technological Support

This section presents three experiments. The first one was conducted with the objective of evaluating the quality of the PR-NOR patterns and the NOR_2O software library by measuring the similarity of the ontologies generated against gold standard ontologies. The second experiment was carried out for evaluating the usability of the software library. Finally, the third experiment was presented in a real case scenario within the GeoLinkedData Project[14], in which the applicability and usability of the software library is evaluated.

11.2.1 Quality Evaluation of the Patterns and NOR$_2$O

The goal of this study is to evaluate the quality of the re-engineering patterns and NOR$_2$O software library by measuring the similarity of the ontologies generated against gold standard ontologies. The ontology generated is compared against a reference ontology (or gold standard) built manually by external ontology experts not involved in the experiment.

11.2.1.1 Settings

For this experiment, two ontology engineering experts built five excerpts of ontologies in OWL from available 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 the process we had five “gold standard” ontologies[15]. It is worth mentioning that the ontologies cover an excerpt of the resources. Table 11.9 shows the resources utilized in this experiment:

Table 11.9: 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

11.2.1.2 Execution

The experiment was executed in the three phases:

1. Each NOR was transformed automatically with the following patterns:

[14] http://geo.linkeddata.es/

[15] The ontologies are available at http://droz.dia.fi.upm.es/ontologies

- ASFA, with the PR-NOR-TSTX-01 pattern.
- ETT, with the PR-NOR-TSTX-01 pattern.
- ACM, with the PR-NOR-CLTX-02 pattern.
- FOET, with the PR-NOR-CLTX-01 pattern.
- BioLexicon, with the PR-NOR-LXTX-02 pattern.

2. For disambiguating the relations between entities of a particular resource we executed the disambiguation algorithm with WordNet.

3. In order to assess the quality of the ontologies generated, we compared the “gold standard” ontologies with the excerpts of the five ontologies generated automatically by means of similarity measures based on (1) the Cider System [Gra09], which considers the structure of the ontologies, that is, classes, object properties and datatype properties; and (2) the StrucSubsDistAlignment measure taken from the Ontology Alignment Evaluation Initiative[16], which contemplates the structure of the ontologies.

11.2.1.3 Collecting results

We built a table for comparing, by means of the similarity measures, each of the “Gold Standard” ontologies with the ontologies generated. Table 11.10 presents the similarity values of every ontology generated.

Table 11.10: Similarity values of every ontology 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

11.2.1.4 Finding and observations

We can state that the ontologies generated have an acceptable similarity degree when compared to the gold standard ones.

Based on the results obtained, we can say that the main strength of the NOR$_2$O software library and patterns is that they generate ontologies with an acceptable level of quality, meaning by quality the similarity of the ontologies to the gold standard ones.

[16] http://oaei.ontologymatching.org/

11.2.2 Usability Evaluation of the Software Library

This reported study refers to the evaluation of the usability of the NOR$_2$O software library in the context of the development of ontologies.

11.2.2.1 Settings

We performed this user study with the same participants involved in the Master Course (see Section 11.1.1). For the study we employed a classification scheme and a thesaurus.

User study 1: Usability of NOR$_2$O for building an ontology with a classification scheme. The classification scheme of this experiment was the Classification of Environmental Protection Activities (CEPA-94[17]), which has 72 terms and is implemented in a database. For this study we extracted an excerpt of 15 terms[18].

User study 2: Usability of NOR$_2$O for building an ontology with a thesaurus. The resource used was the ETT thesaurus, which has 2522 terms and is implemented in an XML file. In this study we extracted an excerpt of 21 terms[19].

Thus, we conducted two experiments following the Software Usability Measurement Inventory (SUMI) method [KC93].

11.2.2.2 Execution

The investigators met with all the participants for 10 minutes and explained the purpose of the evaluation session; then they presented the methodology of SUMI evaluation. Then, the participants had 20 minutes to test the NOR$_2$O software library, and 10 minutes to fill in the SUMI questionnaire on user-interaction satisfaction. During these two phases the participants were not allowed to ask questions to the investigators. The questionnaire was designed to measure the affect, efficiency, learnability, helpfulness and control [DR93]. SUMI is also mentioned in the ISO 9241 standard as a recognized method for testing user satisfaction [(IS98].

11.2.2.3 Collecting results

The SUMI questionnaire includes 50 items with three responses each (“agree”, “undecided”, “disagree”) and the user had to select one of the three responses for each item.

11.2.2.4 Findings and observations

As a general conclusion we can say that the results of the evaluation were positive. The analysis of the results of the experiment conducted reveals some very positive features of the NOR$_2$O software library; it also points out some issues that

[17] Available at `http://ec.europa.eu/eurostat/ramon/`

[18] `http://droz.dia.fi.upm.es/master/it/homework/cepa.zip`

[19] `http://droz.dia.fi.upm.es/master/it/homework/ett.zip`

should be improved in future works. Figure 11.3 depicts the results of the SUMI questionnaires.

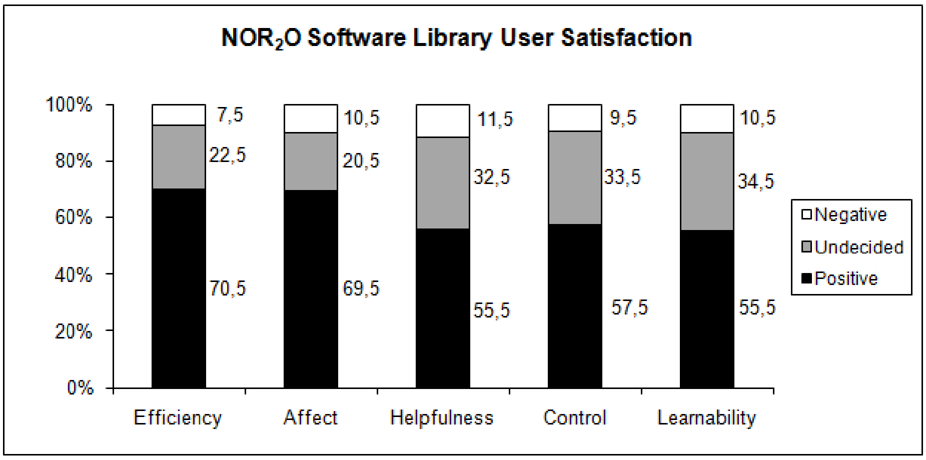


Figure 11.3: The results of SUMI questionnaires for the NOR$_2$O Software Library

Next, we describe the results obtained for each dimension of SUMI questionnaire:

Efficiency
This dimension obtained the higher value; therefore, we believe that the evaluation of the efficiency of the NOR$_2$O software library is satisfactory.

Affect
The affect dimension measures the user's general emotional reaction to the software and may be glossed as Likeability. The item regarding this dimension that most contributed to 10.5% of disagreement in the user's general reaction to the software was: "I feel safer if I use only a few familiar commands or operations". This is one of the aspects of the NOR$_2$O software library we should improve, if we want all the funcionalities to be perceived with the same degree of positiveness by the users.

Helpfulness
Fifty-five and one half percent of the users believe that the software is self-explanatory (helpful). Moreover, we found that the item that more contributed to 32.5% of indecision was: "This software is awkward when I want to do something not standard". This means that the majority of the users did not need to find alternative options to perform the actions available in the software library.

Control
The global control was calculated as the average of the 10 SUMI questions for

this dimension. We consider that the evaluation of this dimension is satisfactory, because we only obtained 9.5% of disagreement. In the same sense, 33.5% of indecision corresponds to aspects that did not appear in the software, such as "The software allows the user to be economic of keystrokes", which is positive.

Learnability

This dimension obtained the lowest value; therefore, we should improve this aspect of the NOR$_2$O software library if we want to increase the speed and facility with which the users learn how to use new features when necessary.

Considering the comments obtained in the experiment, we can state that its main strength is that the majority of students found NOR$_2$O useful and understandable.

11.2.3 Applicability and Usability of NOR$_2$O within the GeoLinkedData Project

In order to evaluate the applicability and usability of the NOR$_2$O software library, we conducted an experiment in a real case scenario within the GeoLinkedData Project[20].

GeoLinkedData is an open initiative whose aim is to enrich the Web of Data [Biz09] with Spanish geospatial data. This initiative started off by publishing diverse information sources belonging to the National Geographic Institute of Spain. Such sources are made available as RDF (Resource Description Framework) knowledge bases according to the Linked Data principles [Biz09]. Within this project we have searched for open government information in two institutions (1) the National Geographic Institute of Spain (IGN), and (2) the National Statistics Institute of Spain (INE). The datasets selected from the INE are available as Excel spreadsheets and these were the following: Population, Unemployment, Building Trade, Dwelling, and Industry.

In the process of linked data generation from the INE datasets, we had to create RDF instances of the Statistical Core Vocabulary (SCOVO) [HHR$^+$09]. Thus, basically we had to perform a Population from the INE datasets of the SCOVO vocabulary. This vocabulary provides an expressive modelling framework for statistical information, and has been used in a variety of applications that requires the representation of statistical information. The vocabulary is currently defined in RDF(S).

In the following sections we describe how to apply NOR$_2$O for generating RDF instances of SCOVO vocabulary.

11.2.3.1 Performing a Population of RDF instances of SCOVO vocabulary

Once we had the INE datasets selected, we had to transform them into ontology instances. Next, we describe the process of generating RDF instances of SCOVO

[20] http://geo.linkeddata.es/

vocabulary.

Activity 1. Non-ontological resource reverse engineering

In this activity we gathered documentation about the INE datasets. From this documentation we realized that resources are a set of bi-dimensional tables in which we have (1) the location in one dimension, (2) the time line in the other, and (3) the set of values of a particular variable. Finally, we realized that INE datasets are stored in Excel spreadsheets. Figure 11.4 shows an example of the Industry Production Index.

Activity 2. Non-ontological resource transformation

In this activity we carried out the following tasks:

1. We identified the transformation approach, Population, i.e., transforming the resource content into ontology instances.

2. Then, we searched our local pattern repository for a suitable pattern to re-engineer NORs, taking into account the transformation approach (Population) and the non-ontological resource type (bi-dimensional tables).

3. As we did not find any suitable pattern for the INE datasets; we had to perform an *ad-hoc* transformation. Thanks to the modular approach of NOR$_2$O, it was easy to extend the software library and include the new non-ontological resource.

4. After the enhancement of the library, we performed the transformation of the INE datasets automatically.

Activity 3. Ontology forward engineering

We relied on the NOR$_2$O software library for generating the ontology instances automatically.

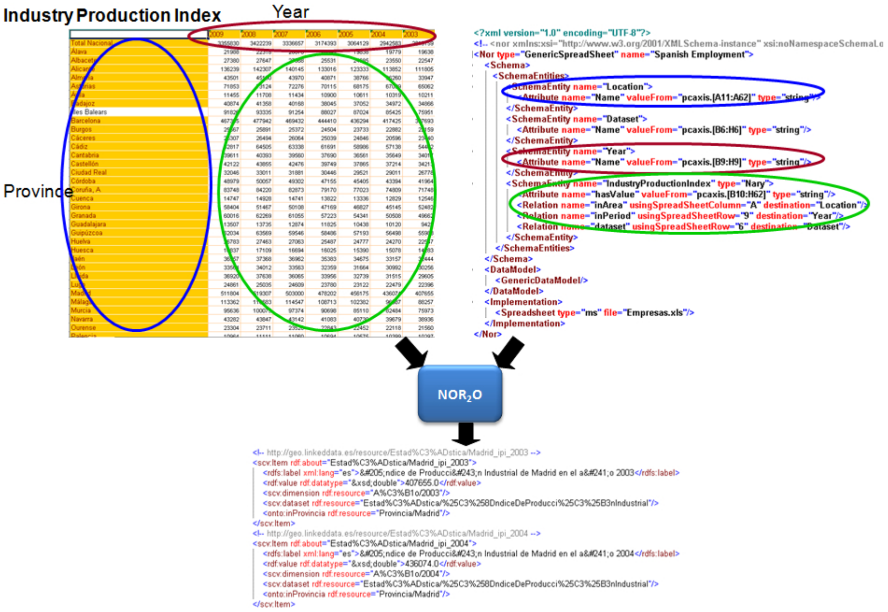
Figure 11.4 illustrates the transformation process from the Excel spreadsheet data to the RDF instances. On the left side we can see the spreadsheet and data that represents the Industry Production Index of Spanish provinces over the years. On the right side we have the nor.xml configuration file that describes the information stored in the spreadsheets. The configuration file describes the information from the spreadsheet: province, year, and the industry production index. Then, the NOR$_2$O software library, with all this information, generates the RDF instances.

Our experience in GeoLinkedData has served us to demonstrate that

- The method for re-engineering can be applied even though we did not find a suitable pattern for the transformation.

- The re-engineering patterns are extensible to other types of resources, and subsequently NOR$_2$O can be extended as well.

- The NOR$_2$O software library is easy to use in other projects.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <nor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLo
<Nor type="GenericSpreadSheet" name="Spanish Employment">
   <Schema>
      <SchemaEntities>
         <SchemaEntity name="Location">
            <Attribute name="Name" valueFrom="pcaxis.[A11:A62]" type="string"/>
         </SchemaEntity>
         <SchemaEntity name="Dataset">
            <Attribute name="Name" valueFrom="pcaxis.[B6:H6]" type="string"/>
         </SchemaEntity>
         <SchemaEntity name="Year">
            <Attribute name="Name" valueFrom="pcaxis.[B9:H9]" type="string"/>
         </SchemaEntity>
         <SchemaEntity name="IndustryProductionIndex" type="Nary">
            <Attribute name="hasValue" valueFrom="pcaxis.[B10:H62]" type="string"/>
            <Relation name="inArea" usingSpreadSheetColumn="A" destination="Location"/>
            <Relation name="inPeriod" usingSpreadSheetRow="9" destination="Year"/>
            <Relation name="dataset" usingSpreadSheetRow="6" destination="Dataset"/>
         </SchemaEntity>
      </SchemaEntities>
   </Schema>
   <DataModel>
      <GenericDataModel/>
   </DataModel>
   <Implementation>
      <Spreadsheet type="ms" file="Empresas.xls"/>
   </Implementation>
</Nor>
```

```
<!-- http://geo.linkeddata.es/resource/Estad%C3%ADstica/Madrid_ipi_2003 -->
<scv:Item rdf:about="Estad%C3%ADstica/Madrid_ipi_2003">
   <rdfs:label xml:lang="es">&#205;ndice de Producci&#243;n Industrial de Madrid en el a&#241;o 2003</rdfs:label>
   <rdf:value rdf:datatype="&xsd;double">407655.0</rdf:value>
   <scv:dimension rdf:resource="A%C3%B1o/2003"/>
   <scv:dataset rdf:resource="Estad%C3%ADstica/%25C3%258DndiceDeProducci%25C3%25B3nIndustrial"/>
   <onto:inProvincia rdf:resource="Provincia/Madrid"/>
</scv:Item>
<!-- http://geo.linkeddata.es/resource/Estad%C3%ADstica/Madrid_ipi_2004 -->
<scv:Item rdf:about="Estad%C3%ADstica/Madrid_ipi_2004">
   <rdfs:label xml:lang="es">&#205;ndice de Producci&#243;n Industrial de Madrid en el a&#241;o 2004</rdfs:label>
   <rdf:value rdf:datatype="&xsd;double">436074.0</rdf:value>
   <scv:dimension rdf:resource="A%C3%B1o/2004"/>
   <scv:dataset rdf:resource="Estad%C3%ADstica/%25C3%258DndiceDeProducci%25C3%25B3nIndustrial"/>
   <onto:inProvincia rdf:resource="Provincia/Madrid"/>
</scv:Item>
```

Figure 11.4: Generation of RDF instances from the excel spreadsheet data.

11.2.4 Summary

As a conclusion we can state that the set of experiments carried out in this section, verifies (1) hypothesis **H5**, the re-engineering patterns proposed can be implemented in a software library that facilitates the work of ontology engineers when developing ontologies; (2) hypothesis **H3**, the method for re-engineering non-ontological resources is extensible and adaptable to other types of resources. The method can be applied to any kind of non-ontological resource independently of its type, data model or implementation; and (3) hypothesis **H4**, it is possible to create patterns for re-engineering that allow generating ontologies from available non-ontological resources; these re-engineering patterns are extensible to other types of resources besides classification schemes, thesauri or lexica.

11.3 Evaluation Summary

This chapter has presented the evaluation of this thesis contributions through a set of experiments. The evaluation of the method proposed, and its technological support, provides positive evidence of the set of hypotheses introduced in Section 3.4. This positive evidence is presented in Sections 11.1.4 and 11.2.4. Moreover, based on the comments and results obtained in the experiments, we have demonstrated that the methodological guidelines and technological support proposed are really valuable and useful for guiding engineers with no previous experience in building a huge ontology network, especially, if the network needs to be solidly grounded in NORs.

Chapter 12

CONCLUSIONS AND FUTURE WORK

This thesis is focused on the reuse and possible subsequent re-engineering of knowledge resources, as opposed to the custom-building of new ontologies from scratch. A deep analysis of the state of the art has revealed that there are some methods and tools for transforming non-ontological resources into ontologies, but that they have some limitations, namely,

- Most of the methods presented are based on *ad-hoc* transformations for the resource type and the resource implementation.
- Only a few methods take advantage of the resource data model, an important artefact for the re-engineering process [GGPSFVT08].
- No integrated framework, method or corresponding tool considers the resources types, data models and implementations identified in a unified way.
- With regard to the transformation approach, most of the methods perform a TBox transformation, some perform an ABox transformation and just a few perform a population. However, no method includes the possibility to perform the three transformations.
- Regarding the degree of automation, almost all the methods perform a semi-automatic transformation of the resource.
- Regarding 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 of the relations of the resource components. Most of the 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 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.

- As for 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.

In this thesis we have provided a method and its technological support that rely on re-engineering patterns in order to speed up the ontology development process by reusing and re-engineering as much as possible available non-ontological resources. To achieve this overall goal, we have decomposed it in the following objectives: (1) the definition of methodological aspects related to the reuse of non-ontolo-gical resources for building ontologies; (2) the definition of methodological aspects related to the re-engineering of non-ontological resources for building ontologies; (3) the creation of a library of patterns for re-engineering non-ontological resources into ontologies; and (4) the development of a software library that implements the suggestions provided by the re-engineering patterns.

Having in mind these goals, in this chapter we present how the open research problems identified in Chapter 2 are solved. Then, we discuss the verification of our hypotheses, and finally we provide an outlook for the future lines of work.

12.1 Review of the Contributions

This section reviews the main contributions of this thesis and how we solved the open research problems.

- Up to the writing of this thesis no definition of non-ontological resources could be found. Moreover, an analysis of the literature has revealed that there are different ways of categorizing NORs, though an accepted and agreed upon typology of NORs does not exist yet. To address the previous limitations, we have introduced the definition of non-ontological resource and proposed a **categorization of them** according to three different features: type of non-ontological resource, data model and implementation. This categorization is neither exhaustive nor complete. Currently, we are enriching it by adding examples taken from RosettaNet[1] and Electronic Data Interchange, EDI[2].

We have presented the categorization in Chapter 5. According to the type of NOR we have classified them into classification schemes, thesauri, lexicons, folksonomies and glossaries. The identified datamodels for classifications schemes are path enumeration, adjacency list, snowflake, and flattened; for thesauri and lexica we have record-based and relation-based.

- Nowadays, most of the NORs exist in pure form without any additional information, e.g., a domain of interest or authorship information, such as that provided by Dublin Core for text documents or by OMV for ontologies. Therefore, it is difficult for academia and industry to identify, find and reuse NORs effectively and efficiently. As consequence, the reuse of NORs for building ontologies is at present a very hard task if not impossible. Thus, in order to address the previous limitations, we have introduced a metadata standard reflecting the most relevant properties of NORs for supporting their reuse, the so-called **Non-ontological Resource Metadata Vocabulary (NoRMV)**. This vocabulary allows (1) describing the non-ontological resources available, and (2) including the provenance information in the ontology generated by extending the Ontology Metadata Vocabulary (OMV).

- Previous efforts towards the reuse and subsequent transformation of available resources for building ontologies had assumed that the non-ontological resources were already selected for their transformation; therefore, they did not provide methodological guidelines for the selection of the resource. To overcome this limitation, we have presented a set of **methodological guidelines on how to find the most suitable non-ontological resources for the development of ontologies**. The methodological guidelines include the definition, goal, inputs, outputs, a set of the activities involved, performer of the activities, and execution time of the activities.

- Some methods and tools for transforming non-ontological resources into ontologies perform *ad-hoc* conversions. In order to cope with the categorization of non-ontological resources proposed in this thesis, we provide a **re-engineering model for non-ontological resources**. The model tries to solve the lack of a model for re-engineering non-ontological resources into ontologies. This model opens the foundations of the re-engineering process of NORs for building ontologies. The model, presented in Chapter 6, is based on the software re-engineering model introduced in Chapter 2. It describes the four software abstraction levels that define each activity in software development: conceptual, requirements, design, and implementation levels. Moreover, this re-engineering model introduces the four ontology

[1] http://www.rosettanet.org/

[2] http://www.edibasics.co.uk/

abstraction levels that define the activities in ontology engineering: specification, conceptualization, formalization, and implementation.

- The methods available for converting non-ontological resources into ontologies do not provide detailed guidelines for the transformation. Thus, to address this limitation we have proposed **a method for re-engineering non-ontological resources** by means of patterns. Our method tries to solve the problem of not having detailed guidelines on how to transform non-ontological resources into ontologies. The method, presented in Chapter 6, relies on the use of patterns guiding the transformation, although, the software library can also be used for generating the ontology automatically.

- The methods for the conversion of non-ontological resources into ontologies do not provide the techniques employed nor do they reuse good practices for the transformation. To overcome this limitation, we propose a set of **patterns**. These patterns for transforming non-ontological resources into ontologies has several advantages: (1) they include expertise in 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.

  According to the NOR categorization presented in Chapter 5, in this thesis we propose patterns for re-engineering classification schemes (Chapter 7), thesauri (Chapter 8), and lexicons (Chapter 9). The set of patterns are included in the the ODP Portal[3] as a PR-NOR library. One of the goals of the PR-NOR library is to become a community-accepted re-engineering pattern library for transforming resources into ontologies. The PR-NOR library includes the three transformation approaches (TBox, ABox and Population). Moreover, the patterns that perform the TBox transformation approach make explicit the hidden semantics in the relations of the NOR terms, by means of external resources, e.g., WordNet.

- The tools that transform non-ontological resources into ontologies do not cover the three transformation approaches (TBox, ABox, and Population) nor the non-ontological resource types identified, among other features. Within this thesis we have developed a software library, **NOR$_2$O**, that implements the transformation suggested by the patterns. This software library tries to solve the lack of technological support for an integrated method that takes into account the different types of NORs and their internal data models and implementations in an uniform way. The NOR$_2$O software library, presented in Chapter 10, is a Java library that performs an ETL process[4] for transforming the non-ontological resource terms into ontology elements. The implementation of NOR$_2$O follows a modular approach; therefore, it is possible

[3] http://ontologydesignpatterns.org/

[4] Extract, transform, and load (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 [KC04].

to extend it and include other types of NORs, data models, and implementations in a simple way, as well as to exploit other external resources for making explicit the hidden semantics in the relations of the NOR terms.

To conclude we present the comparison of our method with the three most representative methods in this area: Heep et al. [HdB07], Hyvönen et al. [HVTS08] and Soergel et al. [SLL^{+}04] (see Tables 12.1, 12.2, 12.3, and 12.4). The comparison is made according to the following features: non-ontological resources, reuse of NORs, transformation, and ontologies generated, which were analysed in the chapter dealing with the state of the art.

Table 12.1: A comparative analysis of the three most representative methods and the pattern-based method. NOR features

Features	Heep et al.	Hyvönen et al.	Soerger et al.	Villazón-Terrazas
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

With respect to the non-ontological resources (see Table 12.1), our method (1) deals with classification schemes, thesauri and lexica; (2) considers the internal data model; and (3) tackles NORs implemented in databases, XML files, spreadsheets, and flat files.

Table 12.2: A comparative analysis of the three most representative methods and the pattern-based method. Reuse features

Features	Heep et al.	Hyvönen et al.	Soerger et al.	Villazón-Terrazas
Reuse				
Detailed guidelines	No	No	No	Yes
Tool support	No	No	No	No
Provenance	No	No	No	Yes

Regarding the reuse of non-ontological resources (see Table 12.2), our method (1) provides methodological guidelines for the selection of the resources to be transformed; and (2) keeps track of the provenance of the resource.

Table 12.3: A comparative analysis of the three most representative methods and the pattern-based method. Transformation features

Features	Heep et al.	Hyvönen et al.	Soerger et al.	Villazón-Terrazas
Transformation				
Transformation approach	TBox	TBox	TBox	TBox, ABox, Population
Transformation aspects	syntactic, semantic	syntactic, semantic	syntactic, semantic	syntactic, semantic
Semantics of the NOR relations	subClassOf, ad-hoc relation	subClassOf, partOf	subClassOf, ad-hoc relation	subClassOf, partOf
Additional resources/Domain expert	No	DOLCE	Domain expert	WordNet
Automatic / Semiautomatic / Manual	Semiautomatic	Semiautomatic	Manual	Semiautomatic
Technique	Not mentioned	Not mentioned	Not mentioned	Re-engineering patterns
Tool support	SKOS2GenTax	ad-hoc tool	Not mentioned	NOR$_2$O

With respect to the transformation of the resources (see Table 12.3), our method (1) performs the three transformation approaches (TBox, ABox and population); (2) considers the syntactic and semantic transformation aspects; (3) contemplates the generation *subClassOf* and *partOf* relations; (4) relies on WordNet as external resource for discovering the hidden semantics of the NOR terms; (5) depends on re-engineering patterns for generating ontologies from the resources; and (6) is supported by the NOR$_2$O software library.

As for the ontologies generated (see Table 12.4), our method generates (1) classes, attributes, relations, and instances; and (2) single ontologies implemented in OWL Lite/RDF.

Table 12.4: A comparative analysis of the three most representative methods and the pattern-based method. Ontology features

Features	Heep et al.	Hyvönen et al.	Soerger et al.	Villazón-Terrazas
Ontology				
Components	classes, relations	classes, attributes, relations	classes, attributes, relations	classes, attributes, relations, instances
Language	RDF(S)/OWL-DLP	RDF(S)	OWL-DL	OWL Lite/RDF
Single / Several	single	single	single	single

12.2 Hypotheses Verification

We have verified the hypotheses of this thesis by different means:

- Within the evaluation of the methodological guidelines, an analysis of the results of the experiments, described in Sections 11.1, 11.1.2, and 11.1.3 some very positive features. For example,
  - The results of the understandability, applicability and usability of the methodological guidelines indicate that the method is specially useful for guiding the ontological engineers. Moreover, the method allows building ontologies faster and with fewer resources (hypothesis **H1**).
  - It is possible to define a unified method for transforming non-ontological resources into ontologies independently (1) of the type, data model or implementation of the resource, and (1) of the target ontology, TBox, TBox+ABox, or ABox (hypothesis **H2**).
  - The set of re-engineering patterns are independent of the domain of the resources; in other words, the patterns can be used to develop ontologies in different domains, e.g., occupation, geographical location, education and training (hypothesis **H4**).
- Within the evaluation of the technological support, the analysis of the results of the experiments, described in Sections 11.2.1, 11.2.2, and 11.2.3 shows also very positive features. For example,
  - The method for re-engineering non-ontological resources is extensible and adaptable to other types of resources, e.g., bidimensional tables (hypothesis **H3**).
  - Re-engineering patterns generate ontologies from available non-ontological resources independently of (1) how they have been implemented

(databases, XML); (2) the target ontology to be generated (TBox, TBox+ABox, or ABox); (3) the domain of the resource (statistical, occupation, eduaction, etc.); and (4) its being extended to other type of resources (hypothesis **H4**).

- NOR$_2$O, the software library that implements the suggestions given by the patterns, facilitates the work of ontology engineers (hypothesis **H5**).

12.3 Future Work

In this thesis we have tackled many open research problems within the context of the reuse and re-engineering of non-ontological resources for building ontologies but there are still open issues to resolve or extensions to implement in the near future. We would like to mention some of the most important from our perspective:

- The improvement of the process of reusing non-ontological resources with the creation of a registry of non-ontological resources that have reached some consensus in a community. The NORs would be annotated by means of NoRMV, described in Chapter 5, thus it would be easy to identify, find and reuse NORs effectively and efficiently.
- Regarding the process of re-engineering non-ontological resources, important features are
  - The building of richer ontologies by extending the taxonomic structures with disjoint knowledge.
  - The inclusion in the re-engineering patterns of the support for transforming excerpts of the resource, and not the whole resource.
  - The inclusion of the support of more non-ontological resource types, data models and implementations, as well as additional external resources like DBpedia for making explicit the semantics on the relations of the NOR terms.
  - The generation of GoodRelations-compliant ontologies for product types and product features. GoodRelations[5] is a standardized vocabulary for product, price and company data that (1) can be embedded into existing static and dynamic Web pages and (2) can be processed by other computers.
- The consideration of multilingual non-ontological resources for building multilingual ontologies. This would require the identification of how the multilingual information is represented in the non-ontological resources and the definition of a linguistic model for expressing the multilingual information

[5] http://www.heppnetz.de/projects/goodrelations/

of the ontologies. Moreover, this feature would also imply that the patterns have to rely on additional knowledge resources, i.e., multilingual, monolingual resources, and background-knowledge resources.

- The consideration of the integration of different knowledge resources. It would be interesting to investigate on methodological guidelines for selecting, comparing and combining non-ontological resources, ontological resources, and ontology design patterns with the aim of building ontology networks.
- The evolution of the non-ontological resources. It would be good to analyse how to transform non-ontological resources that change along the time, and to identify how the frequency of changes affects the ontologization of that resource, and to propose incremental transformations.
- Linked Data has been recently suggested as one of the best alternatives for creating shared information spaces [Biz09]. In the context of Linked Data the RDF language is used to describe resources in the form of triples. One extension of the work presented in this thesis is the generation of RDF data following the Linked Data principles. The NOR_2O software library can be used for this purpose since the library already generates RDF instances. Some of the features to consider are
  - To include the generation of links from the RDF instances generated into RDF resources of RDF datasets presented in the LOD cloud[6],
  - To follow best practices in the URI generation; for example, the *Cool URI for the Semantic Web*[7],
  - To suggest available vocabularies to reuse, when modelling the ontology, taking into account the domain of the resources and using semantic web search engines, such as Sindice[8].

The general goal of this thesis, i.e., the reuse and re-engineering of non-ontological resources for speeding up the ontology development process, is a core requirement for supporting and promoting the new paradigm of the reuse-based approach in ontology development. Thus, our results represent a step forward in the achievement of such a goal.

[6] http://richard.cyganiak.de/2007/10/lod/

[7] http://www.w3.org/TR/cooluris/

[8] http://sindice.com/

Bibliography

[ABM05] Y. An, A. Borgida, and J. Mylopoulos. Constructing Complex Semantic Mappings Between XML Data and Ontologies. In *International Semantic Web Conference*, pages 6–20, 2005.

[Ale79] C. Alexander. *The Timeless Way of Building*. Oxford University Press, New York, 1979.

[ALFZ94] M. Antoni-Lay, G. Francopoulo, and L. Zaysser. A generic model for reusable lexicons: The GENELEX project. *Literary and Linguistic Computing*, 9(1):47–54, 1994.

[AM05] Y. An and J. Mylopoulos. Translating XML Web Data into Ontologies. In *OTM Workshops*, pages 967–976, 2005.

[ANS05] ANSI/NISO. *Documentation – Guidelines for the construction, format, and management of monolingual controlled vocabularies.*, 2005. Report ANSINISO Z3919.

[ASC07] R. Abbasi, S. Staab, and P. Cimiano. Organizing resources on tagging systems using t-org. In *In proceedings of Workshop on Bridging the Gap between Semantic Web and Web 2.0 at ESWC 2007*, June 2007.

[BA05] E. Biesalski and A. Abecker. Human resource management with ontologies. In *Springer Postproceedings: Workshop on IT Tools for Knowledge Management Systems: Applicability, Usability, and Benefits (KMTOOLS)*, pages 499–507. 2005.

[Bar07] J. Barrasa. *Modelo para la definición automática de correspondencias semánticas entre ontologías y modelos relacionales*. PhD thesis, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, March 2007.

[BCGP04] J. Barrasa, O. Corcho, and A. Gómez-Pérez. R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In *Second Workshop on Semantic Web and Databases (SWDB2004)*, 2004.

[Ber94] J. Berge. *The EDIFACT Standards*. Blackwell Publishers, Inc., Cambridge, MA, USA, 1994.

[BH06] S. Brockmans and P. Haase. A Metamodel and UML Profile for Networked Ontologies. A Complete Reference. Technical report, Universitt Karlsruhe,, 2006.

[BHM+05] C. Bizer, R. Heese, M. Mochol, R. Oldakowski, R. Tolksdorf, and R. Eckstein. The impact of semantic web technologies on job recruitment processes. In *International Conference Wirtschaftsinformatik (WI 2005), Bamberg, Germany*, 2005.

[Biz09] C. Bizer. The Emerging Web of Linked Data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.

[BMR+96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: a system of patterns*, volume 1. John Wiley and Sons, 1996.

[Bor97] W. Borst. Construction of Engineering Ontologies, 1997.

[Bra05] D. Brandon. Recursive database structures. *Journal of Computing Sciences in Colleges*, 2005.

[BS 05a] British Standards Institution, BSI. *Documentation – Structured vocabularies for information retrieval - Guide - Part 1: Definitions, symbols and abbreviations.*, 2005. Report BS 8723-1.

[BS 05b] British Standards Institution, BSI. *Documentation – Structured vocabularies for information retrieval - Guide - Part 2: Thesauri.*, 2005. Report BS 8723-2.

[Byr92] E. Byrne. A conceptual foundation for software re-engineering. In *International Conference on Software Maintenance and Reengineering*. IEEE Computer Society, 1992.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1st edition, May 1999.

[Car02] B. Carkenord. Why Build a Logical Data Model. http://www.embarcadero.com/resources/tech_papers/datamodel.pdf, 2002.

[CCI90] E. J. Chikofsky and J. H. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Softw.*, 7(1):13–17, 1990.

[CHPG09] C. Caracciolo, J. Heguiabehere, V. Presutti, and A. Gangemi. Initial Network of Fisheries Ontologies. Technical report, NeOn project deliverable D7.2.3, 2009.

[CNZ96] N. Calzolari, M. Naught, and J. Zampolli. EAGLES Editor's Introduction. http://www.ilc.cnr.it/EAGLES96/edintro/edintro.html, 1996.

[Cor05] O. Corcho, editor. *A Layered Declarative Approach to Ontology Translation with Knowledge Preservation*. IOS Press, 2005.

[CR09] O. Corcho and C. Roussey. SynonymOrEquivalence (SOE) Pattern. *http://ontologydesignpatterns.org*, 2009.

[CTP00] P. Clark, J. Thompson, and B. W. Porter. Knowledge Patterns. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 591–600, 2000.

[CXH04] I. F. Cruz, H. Xiao, and F. Hsu. An ontology-based framework for xml semantic integration. In *IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium*, pages 217–226, Washington, DC, USA, 2004. IEEE Computer Society.

[DR93] J. Dumas and J. Redish. A practical guide to usability testing. *Exeter, UK Intellect*, 1993.

[EPJ06] H. Edwards, R. Puckett, and A. Jolly. Analyzing communication patterns in software engineering projects. In *Software Engineering Research and Practice*, pages 310–315, 2006.

[ES07] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[FB06] D. Foxvog and C. Bussler. Ontologizing EDI Semantics. In *ER (Workshops)*, pages 301–311, 2006.

[Fel98] C. Fellbaum, editor. *WordNet - An Electronic Lexical Database*. MIT Press, 1998.

[FGC$^+$06] G. Francopoulo, M. George, N. Calzolari, M. Monachini, N. Bel, M. Pet, and C. Soria. Lexical markup framework (lmf). In *Proceedings of the fifth international conference on Language Resources and Evaluation, LREC 2006*, Genoa, Italy, 2006.

[GC05] R. García and O. Celma. Semantic Integration and Retrieval of Multimedia Metadata. In *Proceedings of the ISWC 2005 Workshop on Knowledge Markup and Semantic Annotation (Semannot'2005)*, 2005.

[GCCL06] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling ontology evaluation and validation. In *Proceedings of the*

3rd European Semantic Web Conference (ESWC2006), number 4011 in LNCS, Budva. Springer, 2006.

[GGMO03] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WORDNET with DOLCE. *AI Mag.*, 24(3):13–24, 2003.

[GGPSFVT08] A. García, A. Gómez-Pérez, M. C. Suárez-Figueroa, and B. Villazón-Terrazas. A Pattern Based Approach for Re-engineering Non-Ontological Resources into Ontologies. In *Proceedings of the 3rd Asian Semantic Web Conference (ASWC2008)*. Springer-Verlag, 2008.

[GHJV95] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995.

[GNV03] A. Gangemi, R. Navigli, and P. Velardi. The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. In *CoopIS/DOA/ODBASE*, 2003.

[GP08] A. Gangemi and V. Presutti. *Handbook of Ontologies (2nd edition)*, chapter Ontology Design Patterns. Springer: Berlin, 2008.

[GPC08] J. M. Gómez-Pérez and O. Corcho. Problem-solving methods for understanding process executions. *Computing in Science and Engg.*, 10(3):47–52, 2008.

[GPFLC03] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer Verlag, 2003.

[GPMM04] A. Gómez-Pérez and D. Manzano-Macho. An overview of methods and tools for ontology learning from texts. *Knowl. Eng. Rev.*, 19(3):187–212, 2004.

[GPS98] A. Gangemi, D. Pisanelli, and G. Steve. Ontology integration: Experiences with medical terminologies. *Ontology in Information Systems*, pages 163–178, 1998.

[GPSF09] A. Gómez-Pérez and M. C. Suárez-Figueroa. Scenarios for Building Ontology Networks within the NeOn Methodology. In P. in Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009), editor, *Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009)*, 2009.

[Gra09] J. Gracia. *Integration and Disambiguation Techniques for Semantic Heterogeneity Reduction on the Web*. PhD thesis, University of Zaragoza, October 2009.

[Gru93a] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *In Formal Ontology in Conceptual Analysis and Knowledge Representation, In Press. Substantial Revision of Paper Presented at the International Workshop on Formal Ontology*. Kluwer Academic Publishers, 1993.

[Gru93b] T. R. Gruber. A translation approach to portable ontology specifications. *KNOWLEDGE ACQUISITION*, 5:199–220, 1993.

[Hah03] V. Hahn. Turning informal thesauri into formal ontologies: a feasibility study on biomedical knowledge re-use. *Comparative and Functional Genomics*, 4:94–97(4), January/February 2003.

[HdB07] M. Hepp and J. de Brujin. 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)*. Springer-Verlag, 2007.

[Hep06] M. Hepp. Products and services ontologies: A methodology for deriving owl ontologies from industrial categorization standards. *Int. J. Semantic Web Inf. Syst.*, 2(1):72–99, 2006.

[Hep07] M. Hepp. Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. *IEEE Internet Computing*, 11(1):90–96, 2007.

[HFP$^+$06] L. Han, T. Finin, C. Parr, J. Sachs, and A. Joshi. RDF123: a mechanism to transform spreadsheets to RDF. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[HHR$^+$09] M. Hausenblas, W. Halb, Y. Raimond, L. Feigenbaum, and D. Ayers. SCOVO: Using Statistics on the Web of Data. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvnen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 708–722. Springer, 2009.

[HHST06] S. Hakkarainen, L. Hella, D. Strasunskas, and S. Tuxen. A Semantic Transformation Approach for ISO 15926. In *Proceedings of the OIS 2006 First International Workshop on Ontologizing Industrial Standards*, 2006.

[Hir04] G. Hirst. Ontology and the lexicon. In *Handbook on Ontologies in Information Systems*, pages 209–230. Springer, 2004.

[Hod00] G. Hodge. Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files. http://www.clir.org/pubs/reports/pub91/contents.html, 2000.

[HPS05] J. Hartmann, R. Palma, and Y. Sure. Omv ontology metadata vocabulary. In *ISWC 2005 Workshop on Ontology Patterns for the Semantic Web*, 2005.

[HS03] U. Hahn and S. Schulz. Towards a broad-coverage biomedical ontology based on description logics. pac symp biocomput. pages 577–588, 2003.

[HVTS08] E. Hyvönen, K. Viljanen, J. Tuominen, and K. Seppälä. Building a national semantic web ontology and ontology service infrastructure -the finnonto approach. In *ESWC*, pages 95–109, 2008.

[ILC03] N. Ide, A. Lenci, and N. Calzolari. Rdf instantiation of isle/mile lexical entries. In *Proceedings of the ACL 2003 workshop on Linguistic annotation*, pages 30–37, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[(IS98] I. S. O. (ISO). Ergonomic requirements for office work with visual display terminals (vdts) part 11: Guidance on usability, 1998.

[ISO85] International Standard Organization (ISO). *Documentation – Guidelines for the establishment and development of multilingual thesauri*, 1985. Report ISO 5964.

[ISO86] International Standard Organization (ISO). *Documentation – Guidelines for the establishment and development of monolingual thesaurus*, 1986. Report ISO 2788.

[ISO04] International Standard Organization (ISO). *Information technology - Metadata registries - Part 1: Framework*, 2004. Report ISO/IEC FDIS 11179-1.

[JYJRBLRS09] A. Jimeno-Yepes, E. Jimnez-Ruiz, R. Berlanga-Llavori, and D. Rebholz-Schuhmann. Reuse of terminological resources for efficient ontological engineering in life sciences. *BMC Bioinformatics*, 10 Suppl 10, 2009.

[KBH+97] T. Koch, A. Bummer, D. Hiom, M. Peereboom, A. Poulter, and E. Worsfold. Specification for resource description methods Part 3. the role of classification schemes in Internet resource description and discovery. Technical report, DESIRE project deliverable D3.2, 1997.

[KC93] J. Kirakowski and M. Corbett. Sumi: The software usability measurement inventory. *British Journal of Educational Technology*, 24(3):210–212, 1993.

[KC04] R. Kimball and J. Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleanin*. John Wiley & Sons, 2004.

[KSKR06] M. Khayari, S. Schneider, I. Kramer, and L. Romary. Unification of multi-lingual scientific terminological resources using the iso 16642 standard. the termsciences initiative. *CoRR*, abs/cs/0604027, 2006.

[Lab07] L. B. N. Laboratory. eXtended MetaData Registry (XMDR) Project. http://www.xmdr.org/standards/cmaps/Thesaurus Standards Relationships.html, 2007.

[LDP] A. D. Lloyd, R. Dewar, and R. Pooley. Legacy information systems and business process change: a patterns perpective. *Commun. AIS*, page 2.

[LS06] B. Lauser and M. Sini. From agrovoc to the agricultural ontology service/concept server: an owl model for creating ontologies in the agricultural domain. In *DCMI '06: Proceedings of the 2006 international conference on Dublin Core and Metadata Applications*, pages 76–88. Dublin Core Metadata Initiative, 2006.

[LW09] A. Langegger and W. W. Xlwrap - querying and integrating arbitrary spreadsheets with sparql. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[MB05] A. Miles and D. Brickley. SKOS Core Vocabulary Specification. Technical report, World Wide Web Consortium (W3C), November 2005. http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/.

[MDA07] M. Z. Maala, A. Delteil, and A. Azough. A conversion process from flickr tags to rdf descriptions. In *SAW*, 2007.

[Mil05] A. Miles. Quick Guide to Publishing a Thesaurus on the Semantic Web. Technical report, World Wide Web Consortium (W3C), May 2005. http://www.w3.org/TR/2005/WD-swbp-thesaurus-pubguide-20050510/.

[MPBS06] M. Mochol and E. Paslaru Bontas Simperl. Practical Guidelines for Building Semantic eRecruitment Applications. In

Proc. of the International Conference on Knowledge Management (iKnow'06), Special Track: Advanced Semantic Technologies, 2006.

[MS01] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 2001.

[MZ06] E. Malinowski and E. Zimnyi. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data and Knowledge Engineering*, 2006.

[NFF+91] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Mag.*, 12(3):36–56, 1991.

[PG01] L. Paradela-Gonzlez. *Una Metodología para la Gestión del Conocimiento*. PhD thesis, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, March 2001.

[PS98] R. Pooley and P. Stevens. Software reengineering patterns. Technical report, 1998.

[PTS04] H. S. Pinto, C. Tempich, and S. Staab. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397, 2004.

[SAd+07] M. Sabou, S. Angeletou, M. dAquin, J. Barrasa, K. Dellschaft, A. Gangemi, J. Lehman, H. Lewen, D. Maynard, D. Mladenic, M. Nissim, W. Peters, V. Presutti, and B. Villazón. Selection and integration of reusable components from formal or informal specifications. Technical report, NeOn project deliverable D2.2.1, 2007.

[SBF98] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods, 1998.

[SF10] M. C. Suárez-Figueroa. *NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse*. PhD thesis, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain, 2010.

[SFBG+07] M. C. Suárez-Figueroa, S. Brockmans, A. Gangemi, A. Gómez-Pérez, J. Lehmann, H. Lewen, V. Presutti, and M. Sabou. Neon modelling components. Technical report, NeOn project deliverable D5.1.1, 2007.

[SFGP08] M. C. Suárez-Figueroa and A. Gómez-Pérez. Towards a Glossary of Activities in the Ontology Engineering Field. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008)*, 2008.

[SFGPVT09] M. C. Suárez-Figueroa, A. Gómez-Pérez, and B. Villazón-Terrazas. How to Write and Use the Ontology Requirements Specification Document. In *OTM Conferences (2)*, pages 966–982, 2009.

[SLL$^+$04] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering thesauri for new applications: The agrovoc example. *J. Digit. Inf.*, 4(4), 2004.

[SMV09] C. Soria, M. Monachini, and P. Vossen. Wordnet-lmf: fleshing out a standardized format for wordnet interoperability. In *IWIC '09: Proceeding of the 2009 international workshop on Intercultural collaboration*, pages 139–146, New York, NY, USA, 2009. ACM.

[Soe95] D. Soergel. Data models for an integrated thesaurus database. *Comatibility and Integration of Order Systems*, 24(3):47–57, 1995.

[SSSS01] S. Staab, H. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. In *IEEE Intelligent Systems 16(1)*, pages 26–34, 2001.

[SSV02] L. Stojanovic, N. Stojanovic, and R. Volz. A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web. In *Proceedings of the Conference on Intelligent Information Processing*, 2002.

[Tic97] W. F. Tichy. A Catalogue of General-Purpose Software Design Patterns. In *TOOLS '97: Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems*, page 330, Washington, DC, USA, 1997. IEEE Computer Society.

[vAGS06] M. van Assem, A. Gangemi, and G. Schreiber. Conversion of WordNet to a standard RDF/OWL representation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006.

[vAMMS06] M. van Assem, V. Malaisé, A. Miles, and G. Schreiber. A Method to Convert Thesauri to SKOS. In *The Semantic Web: Research and Applications*, pages 95–109. 2006.

[vAMSW04] M. van Assem, M. Menken, G. Schreiber, and J. Wielemaker. A method for converting thesauri to RDF/OWL. In *Proceed-*

ings of the Third International Semantic Web Conference (ISWC). Springer, 2004.

[VTAGS+08] B. Villazón-Terrazas, S. Angeletou, A. García-Silva, A. Gómez-Pérez, D. Maynard, M. C. Suárez-Figueroa, and W. Peters. NeOn Deliverable D2.2.2 Methods and Tools for Supporting Re-engineering. Technical report, NeOn, 2008.

[WB97] S. Wright and G. Budin, editors. *Handbook of terminology management, Basic aspects of terminology management*. John Benjamins Publishing Company, 1997.

[WSWS01] B. Wielinga, A. T. Schreiber, J. Wielemaker, and J. Sandberg. From thesaurus to ontology. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 194–201, New York, NY, USA, 2001. ACM Press.