



NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 – “Semantic-based knowledge and content systems”

D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks

Deliverable Co-ordinator: Mari Carmen Suárez-Figueroa

Deliverable Co-ordinating Institution: UPM

Other Authors: Guadalupe Aguado de Cea (UPM), Carlos Buil (iSOCO), Klaas Dellschaft (UKO-LD), Mariano Fernández-López (CEU), Andrés García (UPM), Asunción Gómez-Pérez (UPM), German Herrero (ATOS), Elena Montiel-Ponsoda (UPM), Marta Sabou (OU), Boris Villazon-Terrazas (UPM), and Zheng Yufei (UPM).

This deliverable presents the first version of the NeOn methodology for building ontology networks.

Document Identifier:	NEON/2008/D5.4.1/v1.0	Date due:	February 29, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 29, 2008
Project start date:	March 1, 2006	Version:	V1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is a part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

<p>Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk</p>	<p>Universität Karlsruhe – TH (UKARL) Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 28 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de</p>
<p>Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es</p>	<p>Software AG (SAG) Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com</p>
<p>Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com</p>	<p>Institut 'Jožef Stefan' (JSI) Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si</p>
<p>Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr</p>	<p>University of Sheffield (USFD) Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk</p>
<p>Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de</p>	<p>Consiglio Nazionale delle Ricerche (CNR) Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it</p>
<p>Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de</p>	<p>Food and Agriculture Organization of the United Nations (FAO) Viale delle Terme di Caracalla 1 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org</p>
<p>Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.pariantelobo@atosorigin.com</p>	<p>Laboratorios KIN, S.A. (KIN) C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es</p>

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

UPM

CEU

UKO-LD

OU

iSOCO

ATOS

Change Log

Version	Date	Amended by	Changes
0.00	20-10-2007	Asunción Gómez-Pérez	ToC and first version of introduction
0.10	30-10-2007	Mari Carmen Suárez-Figueroa	First draft including state of the art
0.11	12-11-2007	Mari Carmen Suárez-Figueroa	Update of first draft including new version of scenarios
0.12	23-11-2007	Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez	Update of first draft including first version of guidelines for ontology specification
0.13	30-11-2007	Mari Carmen Suárez-Figueroa	Update of first draft including first version of guidelines for ontology reuse
0.14	10-12-2007	Mari Carmen Suárez-Figueroa	Revision of the state of the art
0.15	15-12-2007	Mari Carmen Suárez-Figueroa	Update of guidelines for ontology reuse
0.16	20-12-2007	Elena Montiel-Ponsoda, Mari Carmen Suárez-Figueroa	Update of scenarios for building ontology networks
0.17	8-1-2008	Mari Carmen Suárez-Figueroa	Update of the state of the art
0.18	10-1-2008	Asunción Gómez-Pérez	Revision of the state of the art
0.19	15-1-2008	Mari Carmen Suárez-Figueroa	Update of guidelines for ontology reuse
0.20	15-1-2008	Asunción Gómez-Pérez	Revision of ontology specification section
0.21	15-1-2008	Asunción Gómez-Pérez	Revision of ontology reuse section
0.22	15-1-2008	Asunción Gómez-Pérez	Revision of scenarios for building ontology networks
0.23	21-1-2008	Mari Carmen Suárez-Figueroa	Update of guidelines for ontology reuse
0.24	22-1-2008	Klaas Dellschaft	Update of the DILIGENT section in the state of the art
0.25	23-1-2008	Mari Carmen Suárez-Figueroa	Inclusion of chapter about NeOn methodology
0.26	24-1-2008	Mari Carmen Suárez-Figueroa	Inclusion of knowledge reuse section
0.27	25-1-2008	Klaas Dellschaft	Inclusion of argumentation section

0.28	29-1-2008	Mari Carmen Suárez-Figueroa	Update of guidelines for ontology (requirements) specification
0.29	4-2-2008	Mari Carmen Suárez-Figueroa	Update of guidelines for ontology (requirements) specification
0.30	4-2-2008	Mari Carmen Suárez-Figueroa	Revision of argumentation section
0.31	5-2-2008	Mari Carmen Suárez-Figueroa	Revision of scenarios for building ontology networks
0.32	5-2-2008	Mari Carmen Suárez-Figueroa	Inclusion of guidelines for deciding between single ontologies and ontology networks
0.33	6-2-2008	Klaas Dellschaft	Update of argumentation section
0.34	6-2-2008	Boris Villazón-Terrazas	Inclusion of ontology specification example from SEEMP project
		German Herrero	Inclusion of ontology specification example from the Semantic Nomenclature use case
		Carlos Build	Inclusion of ontology specification example from the Invoice Management use case
0.35	6-2-2008	Boris Villazón-Terrazas, Mari Carmen Suárez-Figueroa	Inclusion of ontology specification example from SEEMP project
0.36	7-2-2008	Asunción Gómez-Pérez	Global revision of the deliverable
0.37	9-2-2008	Mari Carmen Suárez-Figueroa	Update and reorganization of the whole deliverable
0.38	11-2-2008	Mari Carmen Suárez-Figueroa	Revision of the state of the art
0.39	12-2-2008	Mari Carmen Suárez-Figueroa	Revision of chapter about NeOn methodology
0.40	12-2-2008	Mariano Fernández-López	Inclusion of guidelines for reusing general/common ontologies
0.41	13-2-2008	Mari Carmen Suárez-Figueroa	Revision of guidelines for reusing general/common ontologies
0.42	13-2-2008	Mariano Fernández-López	Revision of chapters about the state of the art and about the methodology
0.43	14-2-2008	Boris Villazón-Terrazas, Andrés García, Mari Carmen Suárez-Figueroa	Revision of sections in the chapter about the methodology (introduction, types of knowledge resources, importance of reengineering and scenario about reuse and reengineering non ontological resources
0.44	14-2-2008	Mariano Fernández-López	Global revision of the deliverable
0.45	15-2-2008	Mari Carmen Suárez-Figueroa	Global revision of the deliverable
0.46	15-2-2008	Marta Sabou, Mari Carmen Suárez-Figueroa	Inclusion of guidelines for reusing ontology statements
0.47	15-2-2008	Elena Montiel-Ponsoda, Mari Carmen Suárez-Figueroa	Inclusion of guidelines for reusing ontology design patterns
0.48	15-2-2008	Boris Villazon	Inclusion of chapter about non ontological resource reuse
0.49	15-2-2008	Andrés García	Inclusion of chapter about non ontological resource reengineering
0.50	21-2-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Revision of introduction and chapters about non ontological resource reuse and reengineering

0.51	22-2-2008	Mariano Fernández-López	Revision of guidelines for reusing general/common ontologies
0.52	26-2-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Inclusion of a new chapter explaining the research methodology followed for building the NeOn methodology
0.53	26-2-2008	Boris Villazon-Terrazas	Update of chapter about non ontological resource reuse
0.54	26-2-2008	Andrés García	Update of chapter about non ontological resource reengineering
0.55	27-2-2008	Asunción Gómez-Pérez	Revision of introduction
0.56	28-2-2008	Mari Carmen Suárez-Figueroa	Revision of chapter explaining the research methodology followed for building the NeOn methodology
0.57	29-2-2008	Asunción Gómez-Pérez	General revision of the deliverable
0.58	4-3-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Revision and update of chapter about the NeOn methodology
0.59	5-3-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Revision of chapter explaining the research methodology followed for building the NeOn methodology
0.60	5-3-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Revision of chapter about the NeOn methodology
0.61	6-3-2008	Mari Carmen Suárez-Figueroa	Update of chapter about the NeOn methodology
0.62	7-3-2008	Elena Montiel-Ponsoda, Mari Carmen Suárez-Figueroa	Inclusion of guidelines for reusing ontology design patterns
0.63	11-3-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figueroa	Revision of chapter about ontology specification
0.64	12-3-2008	Boris Villazon-Terrazas	Update of example about ontology specification in SEEMP
0.65	12-3-2008	German Herrero	Update of example about Nomenclature ontology specification
0.66	12-3-2008	Mari Carmen Suárez-Figueroa	Update of chapter about the ontological resource reuse
0.67	13-3-2008	Asunción Gómez-Pérez	Revision of chapters 1 to 3
0.68	14-3-2008	Asunción Gómez-Pérez	Revision of chapters 4 to 5
0.69	19-3-2008	Mari Carmen Suárez-Figueroa	Update of chapter about the ontological resource reuse
0.70	24-3-2008	Mari Carmen Suárez-Figueroa	Update of chapter about the ontological resource reuse
0.71	25-3-2008	Elena Montiel-Ponsoda	Revision of English in chapter about the ontological resource reuse
0.72	26-3-2008	Mariano Fernández-López	General revision of the deliverable
0.73	27-3-2008	Boris Villazon-Terrazas, Andres García	Update of chapter about the non ontological resource reuse and reengineering
0.74	27-3-2008	Elena Montiel-Ponsoda	Update of chapter about the ontology design pattern reuse
0.75	28-3-2008	Carlos Build	Update of example about Invoice Management ontology specification
0.76	28-3-2008	Asunción Gómez-Pérez	General revision

0.77	28-3-2008	Mari Carmen Suárez -Figuerola	Update of chapters 1 to 3
0.78	1-4-2008	Mariano Fernández-López	Update of section about general or common ontologies
0.79	1-4-2008	Asunción Gómez-Pérez, Mari Carmen Suárez-Figuerola	Revision and update of sections about ontology reuse as a whole and ontology statements
0.80	1-4-2008	Mari Carmen Suárez-Figuerola	Conclusion and future work
0.81	1-4-2008	Asunción Gómez-Pérez	Revision of chapters 1 to 5
0.82	1-4-2008	Elena Montiel-Ponsoda, Mari Carmen Suárez-Figuerola	Update of chapter about the ontology design pattern reuse
0.83	2-4-2008	Boris Villazon-Terrazas, Andres García	Update of chapter about the non ontological resource reuse and reengineering
0.84	2-4-2008	Asunción Gómez-Pérez	Revision and update of introduction, executive summary, and conclusion
0.85	3-4-2008	Elena Montiel-Ponsoda, Mari Carmen Suárez-Figuerola	Update of chapter about the ontology design pattern reuse
		Boris Villazon-Terrazas, Andres García	Update of chapter about the non ontological resource reuse and reengineering
		Mari Carmen Suárez-Figuerola	Update of chapter about the ontological resource reuse
0.90	3-4-2008	Mari Carmen Suárez-Figuerola	Global revision of the deliverable
0.95	3-4-2008	Martin Dzbor	Q.A. comments
1.0	3-4-2008	Mari Carmen Suárez-Figuerola	Final version

Executive Summary

Research on Ontology Engineering methodologies is reaching its “adolescence”. The mid 1990s and the first years of this new millennium have witnessed the growing interest of many practitioners in approaches that support the creation and management as well as the population of single ontologies built from scratch. There are some well recognized methodological approaches (e.g., METHONTOLOGY, On-To-Knowledge, and DILIGENT) that provided guidelines to help researchers to develop ontologies. However, they have at least four important limitations:

1. The methodologies lack guidelines for building ontologies by reusing and reengineering other ontologies and existing knowledge resources widely consensuated in a particular domain.
2. The methodologies lack of guidelines for contextualizing an existing ontology and plugging it in with existing ontologies that might be in continuous evolution.
3. These methodologies do not explain the ontology building process with the same style and granularity than those methodologies for developing software.

The main goal of this deliverable is to present the first version of the NeOn methodology for building network of ontologies. The principles that guide the construction of such methodology are:

1. The methodology should be general enough in the sense that it should help software developers and ontology practitioners to build network of ontologies with NeOn toolkit and with other widely used platforms such as Protégé or Top Braid Composer.
2. For each process or each activity, the methodology should define it precisely, state clearly its purpose, its inputs and outputs, the actors involved, when it is more convenient its execution, and the set of methods, techniques and tools to be used for executing the

activity. Furthermore, the methodology should provide prescriptive guidelines for each process or each activity.

3. To facilitate a promptly assimilation by software developers and ontology practitioners, we present the methodology in a manner non oriented to researchers. We also include examples of how to use the methodology in different use cases.

The scope of this deliverable is limited to the following:

1. Ontology specification activity.
2. Reuse and reengineering of non ontological resources. By non ontological resource we mean: a knowledge aware resource whose semantics has not been formalized yet by means of an ontology. Elements in this set are: glossaries, dictionaries, lexicons, classification schemes and taxonomies, and thesauri.
3. Reuse of ontological resources. By ontological resources we mean: a set of elements extracted from a set of available ontologies in order to solve a need. Elements from this set can be: ontologies, ontology modules, ontology statements or ontology design patterns. In this deliverable we analyze: general or common ontologies, domain ontologies, ontology statements, and ontology design patterns.

Table of Contents

NeOn Consortium	2
Work package participants	3
Change Log	3
Executive Summary	6
Table of Contents	8
List of Tables	10
List of Figures	11
1. Introduction	13
1.1. WP5 Objectives and Main Tasks	13
1.2. Deliverable Main Goals and Contributions	14
1.3. Deliverable Structure	15
1.4. Relation with D5.3.1 and D5.6.1 and the Rest of WPs within the NeOn Project	15
2. State of the Art on Methodologies	17
2.1. Definitions for Methodology, Method, and Technique	17
2.2. METHONTOLOGY	18
2.3. On-To-Knowledge	21
2.4. DILIGENT	23
2.5. Comparison of Presented Methodologies	25
3. Research Methodology	28
3.1. General Framework for Describing the NeOn Methodology	28
3.2. Conditions for the NeOn Methodology for Building Ontology Networks	31
3.2.1. Necessary Conditions	31
3.2.2. Sufficient Conditions	32
4. NeOn Methodology for Building Ontology Networks	34
4.1. Scenarios for Building Ontology Networks	34
4.2. Argumentation and Collaboration in NeOn Scenarios	36
4.3. When do Ontologies become Ontology Networks?	37
5. Ontology Specification	40
5.1. State of the Art	40
5.1.1. Methods	40
5.1.2. Techniques	40
5.1.3. Tools	41
5.1.4. Conclusion	42
5.2. Proposed Guidelines for Ontology Specification	42
5.3. Examples	48
5.3.1. SEEMP Reference Ontology Specification	49
5.3.2. Invoice Reference Ontology Specification	55
5.3.3. Semantic Nomenclature Reference Ontology Specification	57

5.4. Future Work	63
6. Non Ontological Resource Reuse and Reengineering.....	64
6.1. Introduction	64
6.2. State of the Art	65
6.2.1. <i>Methods</i>	65
6.2.2. <i>Techniques</i>	68
6.2.3. <i>Tools</i>	68
6.2.4. <i>Conclusion</i>	69
6.3. Type of Non Ontological Resources	69
6.4. The NeOn Approach for Non Ontological Resource Reuse and Reengineering	75
6.5. NeOn Proposed Guidelines for Non Ontological Resource Reuse	77
6.5.4. <i>Example</i>	81
6.6. NeOn Approach for Non Ontological Resource Reengineering	82
6.6.1. <i>NeOn General Model for Non Ontological Resource Reengineering</i>	82
6.6.2. <i>NeOn Activities and Tasks for Non Ontological Resource Reengineering</i>	83
6.6.3. <i>Patterns for Reengineering Non Ontological Resources</i>	87
6.7. Conclusions and Future Work	91
7. Ontological Resource Reuse	93
7.1. Introduction	93
7.2. General Criteria for Ontological Resource Reuse	96
7.3. Proposed Guidelines for Reusing General or Common Ontologies	97
7.3.1. <i>Detailed Guidelines for Situation 1: the comparative study exists</i>	98
7.3.2. <i>Detailed Guidelines for Situation 2: the comparative study does not exist</i>	101
7.3.3. <i>Example of Situation 2</i>	102
7.4. Proposed Guidelines for Reusing Domain Ontologies as a Whole	111
7.5. Proposed Guidelines for Reusing Ontology Statements	116
7.6.1. <i>Experiments on Ontology Statement Reuse</i>	121
8. Ontology Design Patterns Reuse	122
8.1. Introduction	122
8.2. State of the Art	123
8.2.1. <i>Methods</i>	125
8.2.2. <i>Techniques</i>	125
8.2.3. <i>Tools</i>	125
8.2.4. <i>Conclusion</i>	126
8.3. NeOn Method for the Reuse of Ontology Design Patterns by Naive Users	126
8.3.1. <i>Enrichment of NeOn Ontology Design Patterns with Lexico-Syntactic Patterns</i>	127
8.3.2. <i>SOS NeOn plug-in, System for Ontology design patterns Support</i>	132
8.3.3. <i>Input Refinement</i>	134
8.4. Proposed Guidelines for Ontology Design Patterns Reuse by Naive Users	136
9. Conclusions and Future Work	140
References.....	142
Annex A. Hands-on Experiments in using the NeOn Watson Plug-in	150
Use Case 1: Ontology Enrichment.....	150
Use Case 2: Ontology Development.....	150

List of Tables

Table 1. Summary of Conclusions	26
Table 2. Template for Process and Activity Filling Card	30
Table 3. Ontology Specification Filling Card	43
Table 4. Template for the OSRD	44
Table 5. Examples of Terminology and Frequency	52
Table 6. Examples of Objects	53
Table 7. Excerpt of SEEMP Reference Ontology Requirement Specification Document.....	55
Table 8. Example of Terms and Frequency	61
Table 9. Examples of Objects	61
Table 10. Excerpt of Semantic Nomenclature Reference Ontology Requirement Specification Document	63
Table 11. Non Ontological Resource Reuse Filling Card	78
Table 12. Assessment Table	81
Table 13. Assessment Table for SEEMP Occupation Standards	82
Table 14. Non Ontological Resource Reengineering Filling Card	84
Table 15. <i>Pattern for Reengineering Non Ontological Resource</i> Template.....	88
Table 16. Example of Pattern for Reengineering Non Ontological Resource.....	91
Table 17. Common Ontology Reuse Filling Card	98
Table 18. Features of Ontologies that implement Mereotopology Theories	107
Table 19. Competency Question Analysis for Mereology Ontology Reuse	108
Table 20. Required Features for the Ontology to be developed (in grey).....	110
Table 21. Domain Ontology Reuse Filling Card.....	112
Table 22. Hypothetical Example of Domain Ontology Assessment Table.....	114
Table 23. Hypothetical Example of Domain Ontology Selection Table.....	115
Table 24. Ontology Statement Reuse Filling Card.....	117
Table 25. Examples of Hearst Patterns	127
Table 26. LSPs Field included in the NeOn ODPs Template	128
Table 27. Restricted Words and Symbols in LSPs	129
Table 28. LSPs (en) for the <i>SubClassOf</i> ODP (LP-SC-01).....	130
Table 29. LSPs (en) for the <i>DisjointClasses</i> ODP (LP-Di-01).....	130
Table 30. LSPs for the <i>ExhaustiveClasses</i> ODP (LP-EC-01).....	130
Table 31. JAPE Rule for LSP 4 of LP-SC-01	131
Table 32. Example of Annotation Results by ANNIE (GATE).....	133
Table 33. ODPs Reuse by Naive users Filling Card.....	137
Table 34. Comparative Analysis of Three Analyzed Methodologies and the NeOn Methodology Version 1.....	141

List of Figures

Figure 1. Graphical Representation of Terminological Relationships in Methodologies [59]	18
Figure 2. METHONTOLOGY Ontology Life Cycle	20
Figure 3. Ontology Reengineering Activities [61]	21
Figure 4. On-To-Knowledge Ontology Life Cycle [107]	22
Figure 5. Life Cycle Model of the DILIGENT Methodology [40]	25
Figure 6. Inputs taken into account for obtaining the NeOn Methodology	28
Figure 7. Process, Activities and Tasks	29
Figure 8. Scenarios for Building Ontology Networks	36
Figure 9. Simple Graphical Examples of Single Ontologies and Ontology Networks	38
Figure 10. Tasks for Ontology Specification	45
Figure 11. Excerpt of the Competency Questions and Answers in an Excel File	50
Figure 12. Excerpt of the Competency Questions in a Mind Map Tool.....	51
Figure 13. Competency Questions Groups.....	51
Figure 14. Competency Questions Groups in detail	51
Figure 15. Excerpt of the Semantic Nomenclature Competency Questions	59
Figure 16. Semantic Nomenclature Competency Questions Groups	60
Figure 17. Examples of Competency Questions in Groups	60
Figure 18. Classification Schemes Data Models	71
Figure 19. Non Ontological Resources Categorization	73
Figure 20. Water Area Classification	73
Figure 21. Water Area Classification Data Models	74
Figure 22. Water Area Classification XML Implementation for the Adjacency List Model	74
Figure 23. Water Area Classification Spreadsheet Implementation for the Adjacency List Model .	75
Figure 24. Water Area Classification XML Implementation for the Path Enumeration Model.....	75
Figure 25. Non Ontological Reuse and Reengineering Approach	76
Figure 26. General Model for Software Reengineering [27]	77
Figure 27. Proposed Activities in NeOn for the Non Ontological Resource Reuse Process	79
Figure 28. Reengineering Model for Non Ontological Resources.....	83
Figure 29. Proposed Activities in NeOn for the Non Ontological Resource Reengineering Process	85
Figure 30. Different Types of Ontological Resource Reuse.....	94
Figure 31. Ontological Resource Reuse Definitions	95
Figure 32. Activities for Reusing Common Ontologies in Situation 1	99

Figure 33. Activities for Reusing Common Ontologies in Situation 2.....	101
Figure 34. Hasse Diagram of Mereological Theories (from weaker to stronger, going uphill) [118]	105
Figure 35. Activities for Reusing Domain Ontologies as a Whole.....	113
Figure 36. Activities for the Ontology Statement Reuse	118
Figure 37. SOS NeOn Plug-in Workflow.....	132
Figure 38. Possibilities for enriching Taxonomies.....	135
Figure 39. Tasks for ODPs Reuse by Naive Users.....	138

1. Introduction

Research on Ontology Engineering methodologies is reaching its “adolescence”. The mid 1990s and the first years of this new millennium have witnessed the growing interest of many practitioners in approaches that support the creation and management, as well as the population, of single ontologies built from scratch. There are some well recognized methodological approaches (e.g., METHONTOLOGY, On-To-Knowledge, and DILIGENT) that provide guidelines to help researchers to develop ontologies. However, they have at least four important limitations:

1. The methodologies lack guidelines for building ontologies by reusing and reengineering other ontologies and existing knowledge resources widely consensuated in a particular domain.
2. The methodologies lack guidelines for contextualizing an existing ontology and plugging it in with existing ontologies that might be in continuous evolution.
3. These methodologies do not explain the ontology building process with the same style and granularity than those methodologies for developing software.

The development of large-scale semantic applications in the near future will be characterized by using a very large number of ontologies embedded in ontology networks¹. Such ontologies will be built collaboratively by distributed teams. With the goal of speeding up the ontology development process, ontology practitioners are starting to reuse as much as possible other ontologies, ontology modules, ontology statements and ontology design patterns as well as knowledge aware resources such as thesauri, lexicons, DBs, UML diagrams and classification schemas built by others that already have some degree of consensus. This combination has at least three important benefits:

1. Existing authoritative ontologies will be reused more and more.
2. Knowledge aware resources usually contain terminology already consensuated by a broad community of people using a given protocol for reaching consensus. So, at least labels used for naming terms are consensuated, and in the case of classification schemas, they provide a taxonomy that can be easily transformed into a lightweight ontology.
3. Ontologies will be built cheaper and faster because reengineering knowledge aware resources is less time and resource consuming than acquiring knowledge in a domain, reaching consensus, and formalizing it.

1.1. WP5 Objectives and Main Tasks

In this context, the main objectives of WP5 are:

- 1 To create the NeOn methodology that support the collaborative aspects of ontology development, and the reuse and the dynamic evolution of networked ontologies in distributed environments, in which contextual information is introduced by developers (domain experts, ontology practitioners) at different stages of the ontology development process.
- 2 To create a rigorous, sound NeOn methodology for the development of large scale Semantic Web applications that supports the reference architecture and the service oriented infrastructure developed in WP6.

¹ An ontology network or a network of ontologies is a collection of ontologies together through a variety of different relationships such as mapping, modularization, version, and dependency relationships [70].

- 3 To provide qualitative and quantitative experimental evidence of how by following the NeOn methodologies the system development improves.

These objectives will be achieved through investigating the following tasks:

- Task 5.3. Identification and definition of the *development process* and *life cycle for networks of ontologies*. Results of these researches were included in D5.3.1 [111].
- Task 5.4. The NeOn methodology for building collaboratively ontology networks will include methods, techniques and tools for carrying out the activities identified and defined in the ontology network development process. Research results are presented in this deliverable.
- Task 5.5. The NeOn methodology for development of large-scale Semantic Web applications from the initial phases (requirement analysis) of the development process until the stage prior to the implementation. To ease the use of the NeOn reference architecture and the NeOn software components, a set of developer-oriented reference specifications will be defined. These specifications will serve as skeleton for adapting the selected components and for developing new complex semantic-based software components and semantic applications.
- Task 5.6. Experimentation with NeOn methodologies. In this task experiments, methods, and metrics are proposed for evaluating the main outcomes produced in this WP. The goal is to provide qualitative and quantitative evidence that with the NeOn methodologies ontologies and systems are built faster and better.

1.2. Deliverable Main Goals and Contributions

The main goal of this deliverable is to present the first version of a methodology for building networks of ontologies. The principles that guide the construction of such a methodology are:

1. The methodology should be general enough in the sense that it should help software developers and ontology practitioners to build networks of ontologies with the NeOn toolkit and with other widely used platforms such as Protégé or Top Braid Composer.
2. The methodology should define each process or activity precisely; state clearly its purpose, its inputs and outputs, the actors involved, when its execution is more convenient, and the set of methods, techniques and tools to be used for executing it.
3. To facilitate a promptly assimilation by software developers and ontology practitioners, we present the methodology in a prescriptive way none oriented to researchers. We also include examples on how to use the methodology in different use cases.

The scope of this deliverable is limited to the following:

1. Ontology specification activity.
2. Reuse and reengineering of non ontological resources. By non ontological resource we mean: a knowledge aware resource whose semantics has not been formalized yet by means of an ontology. Elements in this set are: glossaries, dictionaries, lexicons, classification schemes and taxonomies, and thesauri.
3. Reuse of ontological resources. By ontological resources we mean: a set of elements extracted from a set of available ontologies in order to solve a need. Elements from this set can be: ontologies, ontology modules, ontology statements or ontology design patterns. In this deliverable we analyze: general or common ontologies, domain ontologies, ontology statements, and ontology design patterns.

1.3. Deliverable Structure

The deliverable is structured as follows:

- ❑ Chapter 2 deals with the state of the art on methodological issues in Ontology Engineering. We briefly present the most well known methodologies and we compare them according to the following features: critical dimensions within NeOn (that is, collaboration, dynamics and context), degree of coverage of the activities included in this deliverable, and whether methodologies are targeted to software developers and ontology practitioners, or to ontology researchers.
- ❑ Chapter 3 explains the research methodology followed for creating the NeOn methodology for building ontology networks, and the general framework for describing such NeOn methodology.
- ❑ Chapter 4 presents: (1) an overview of the different scenarios for building ontologies presented in D5.3.1 [111], (2) an analysis on how argumentation and collaboration issues are related to the different scenarios, and (3) an explanation of when it is better to develop a single ontology and when an ontology networks.
- ❑ Chapter 5 presents the proposed methodological guidelines for carrying out the ontology specification activity, and includes three examples on how to carry out this activity in different use cases from the NeOn project and the SEEMP project.
- ❑ Chapter 6 explains the proposed methodological guidelines for the non ontological resource reuse and reengineering processes.
- ❑ Chapter 7 presents the proposed methodological guidelines for the ontological resource reuse, distinguishing between reusing general or common ontologies, domain ontologies as a whole or ontology statements.
- ❑ Chapter 8 describes the proposed methodological guidelines for carrying out ontology design patterns reuse, focused on a user non expert in design patterns.
- ❑ Chapter 9 presents the conclusions and future work.

1.4. Relation with D5.3.1 and D5.6.1 and the Rest of WPs within the NeOn Project

The relation between this deliverable and the rest of the work done in WPs in the NeOn project is briefly described below:

- ❑ Methodological guidelines for a subset of the process or activities defined in the NeOn Glossary [111] are included in this deliverable.
- ❑ For some of the processes or activities described in this deliverable and for others in the NeOn Glossary we planned experiments, which are described in D5.6.1.
- ❑ D2.2.1 [98] from WP2 has been reviewed and its content considered in the proposed guidelines for reusing and reengineering.
- ❑ D2.5.1 [94] from WP2 has been reviewed and its content considered in the proposed guidelines for reusing ontology design patterns.
- ❑ To operationalize a methodology it is desirable to have tools that reflect and support all processes and activities of the methodology, and guide users step by step through the ontology engineering process. For this reason, although not included in this deliverable, we are identifying which NeOn plug-ins described in deliverable D6.10.1 from WP6 give support to the

activities included in the methodology. Such identification will be included in the next version of the deliverable.

2. State of the Art on Methodologies

A series of existing methods and methodologies for developing ontologies from scratch have been reported in [59] and can be summarized as follows: in 1990, some general steps and some interesting points about the Cyc ontology development were published. Some years later, in 1995, on the basis of the experience gathered in developing the Enterprise ontology and the TOVE (TOronto Virtual Enterprise) project ontology both in the domain of enterprise modeling, the first guidelines were proposed. In 1996, a method to build an ontology in the domain of electrical networks as part of the Esprit KACTUS project was presented. The METHONTOLOGY methodology [59] appeared simultaneously. In 1997, a new method was proposed for building ontologies based on the SENSUS ontology. Then some years later, in 2001, the On-To-Knowledge methodology appeared within the project with the same name. One of the main limitations of all the aforementioned approaches is that they do not consider collaborative and distributed development of ontologies [59]. In fact, the first method that included a proposal for collaborative construction was Co4 [41, 42]. This method includes a protocol for agreeing new pieces of knowledge with the rest of the knowledge architecture, which has been previously agreed upon. After this, in 2004, the DILIGENT methodology [90] appeared which is intended to support domain experts in a distributed setting to engineer and evolve ontologies.

From the aforementioned methods and methodologies, in this section we include chronologically a description of three of them (METHONTOLOGY, On-To-Knowledge and DILIGENT). A detailed explanation of all of them can be found in [59].

Before that, we present definitions from IEEE for the terms methodology, method, technique, process, activity, and task.

At the end of this chapter, we compare METHONTOLOGY, On-To-Knowledge and DILIGENT with respect to the following characteristics:

- ❑ Critical dimensions within NeOn (that is, collaboration, dynamics and context).
- ❑ Degree of coverage of the processes or activities included in this deliverable (that is, ontology specification, non ontological resource reuse, non ontological resource reengineering, reusing ontological resources, and reusing ontology design patterns) by providing detailed guidelines.
- ❑ If they are targeted to software developers and ontology practitioners, and not to ontology researchers.

2.1. Definitions for Methodology, Method, and Technique

Throughout literature, the terms *methodology*, *method*, *technique*, *process*, *activity*, etc. are used indiscriminately [76]. To make clear the use of these terms, we have adopted the IEEE definitions for such terms in this deliverable².

The IEEE [19] defines **methodology** as “a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed” [6]. Methods and techniques are parts of methodologies. A **method** [19] is a set of “orderly processes or procedures used in the engineering of a product or performing a service” [6]. A **technique** [5] is “a technical and managerial procedure used to achieve a given objective” [4]. De Hoog [76] explores relationships between methodologies and methods. According to him, methodologies and methods are not the same because “methodologies refer to knowledge about

² This section is a summary taken from [59]

methods". Methodologies state "what", "who" and "when" a given activity should be performed. Greenwood [66] also explores the differences between methods and techniques. A method is a general procedure while a technique is the specific application of a method and the way in which the method is executed. Usually, there are several techniques for applying a given method.

Methods and techniques are strongly related because both are used to carry out tasks inside the different processes of which a methodology consists of. The IEEE defines a **process** [3] as a "function that must be performed in the software life cycle. A process is composed of activities". An **activity** [3] is "a constituent task of a process". Another definition of **activity** [2] is a defined body of work that is to be performed, including its required input and output information. A **task** is the smallest unit of work subject to management accountability. "A task [3] is a well-defined work assignment for one or more project members. Related tasks are usually grouped to form activities".

The relationships between the aforementioned definitions are summarized in Figure 1 [59], where we can see that a methodology is composed of methods and techniques. Methods are composed of processes and are detailed with techniques. Processes are composed of activities. Finally, activities are made up of groups of tasks.

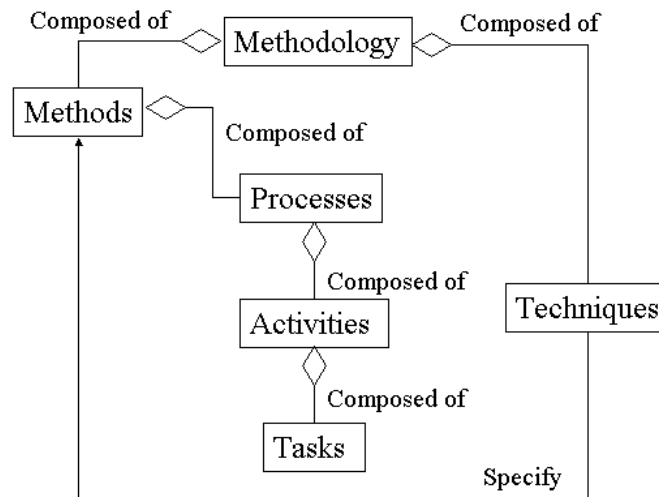


Figure 1. Graphical Representation of Terminological Relationships in Methodologies [59]

2.2. METHONTOLOGY

METHONTOLOGY methodology [59, 48, 19] was developed within the Ontology Engineering Group at Universidad Politécnica de Madrid. This methodology enables the construction of ontologies at the knowledge level.

METHONTOLOGY identifies the set of activities to be carried out. Such set is based on the main activities identified by the software development process [1] and used in Knowledge Engineering methodologies [120, 62].

This methodology includes: the identification of the ontology development process, a life cycle based on evolving prototypes, and techniques to carry out each activity in the management, development-oriented, and support activities.

To give technological support to METHONTOLOGY, ODE [19] and WebODE [15] were built within the same group at Universidad Politécnica de Madrid. Other ontology tools and tool suites can also be used to build ontologies following this methodology, for example, the NeOn Toolkit, Protégé,

etc. METHONTOLOGY has been proposed³ for ontology construction by the Foundation for Intelligent Physical Agents (FIPA), which promotes inter-operability across agent-based applications.

METHONTOLOGY proposes an ontology building life cycle based on evolving prototypes because it allows adding, changing, and removing terms in each new prototype.

For each prototype, METHONTOLOGY proposes to begin with the scheduling activity that identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion. After that, the ontology specification activity starts and at the same time several activities begin inside the management (control and quality assurance) and support processes (knowledge acquisition, integration, evaluation, documentation, and configuration management). All these management and support activities are performed in parallel with the development activities (specification, conceptualization, formalization, implementation and maintenance) during the whole life cycle of the ontology.

In the case of the **ontology specification activity**, METHONTOLOGY proposes the use of competency questions or intermediate representations for describing the requirements that the ontology should fulfill. However, this methodology does not provide detailed guidelines for carrying out this activity.

Regarding the knowledge acquisition activity, METHONTOLOGY proposes the use of techniques taken from the knowledge engineering field.

Once the first prototype has been specified, the conceptual model is built within the ontology conceptualization activity. This is like assembling a jigsaw puzzle with the pieces supplied by the knowledge acquisition activity, which is completed during the conceptualization. For carrying out this conceptualization activity, METHONTOLOGY provides detailed guidelines.

Then, the formalization and implementation activities are carried out. If some lack is detected after any of these activities, we can return to any of the previous activities to make modifications or refinements. When tools like the WebODE ontology editor are used, the conceptualization model can be automatically implemented into several ontology languages using translators. Consequently, formalization is not a mandatory activity in METHONTOLOGY.

Figure 2 shows the ontology life cycle proposed in METHONTOLOGY, and summarizes the previous description. Note that the activities inside the management and support processes are carried out simultaneously with the activities inside the development process.

³ <http://www.fipa.org/specs/fipa00086/> (last access, January 16, 2008)

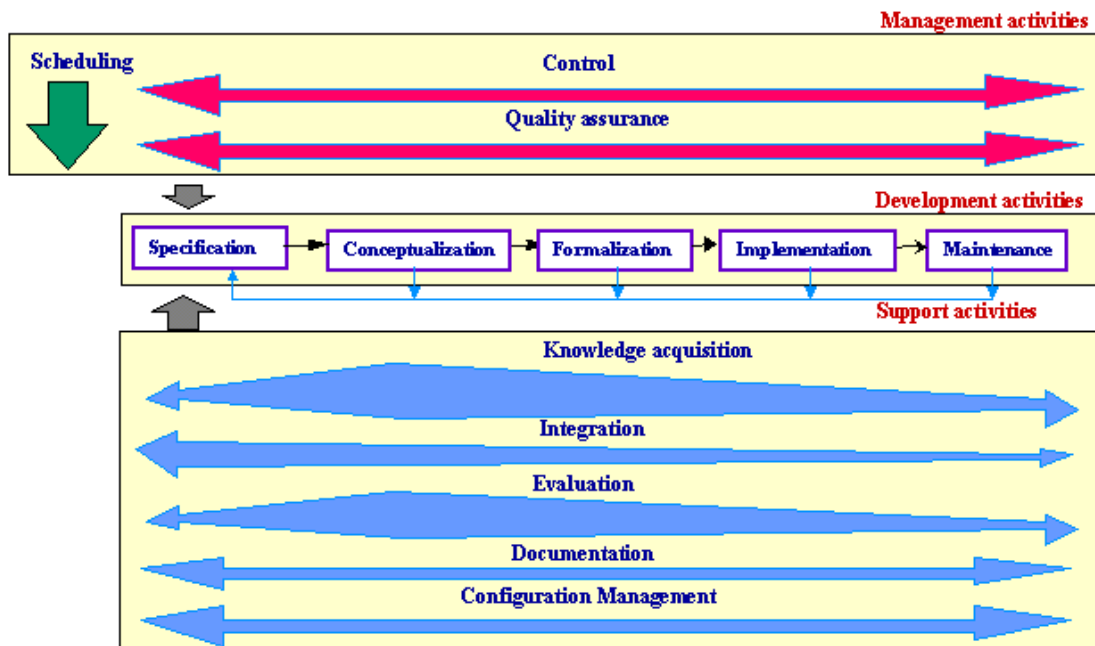


Figure 2. METHONTOLOGY Ontology Life Cycle

Related to the support activities, Figure 2 also shows that the knowledge acquisition, integration and evaluation are greater during the ontology conceptualization, and that it decreases during formalization and implementation. The reasons for this greater effort are:

- ❑ Most of the knowledge is acquired at the beginning of the ontology construction.
- ❑ The integration of other ontologies into the one we are building is not postponed to the implementation activity. Before the integration at the implementation level, the integration at the knowledge level should be carried out.
- ❑ The ontology conceptualization must be evaluated accurately to avoid propagating errors in further stages of the ontology life cycle.

The relationships between the activities carried out during the ontology development are called *intra-dependencies*, or what is the same, they define the ontology life cycle.

METHONTOLOGY also considers that the activities performed during the development of an ontology may involve performing other activities in other ontologies already built or under construction [47]. Therefore, METHONTOLOGY considers not only *intra-dependencies*, but also *inter-dependencies*. *Inter-dependencies* are defined as the relationships between activities carried out when building different ontologies. Instead of talking about the life cycle of an ontology, we should talk about crossed life cycles of ontologies. The reason is that, frequently before integrating an ontology in a new one, the ontology to be reused is modified or merged with other ontologies of the same domain.

The idea of integrating an ontology in a new one is related to the ***reuse of existing ontologies***. In this case, METHONTOLOGY includes the list of activities to be carried out during the ontology reuse, but does not provide detailed guidelines for such activities. Furthermore, METHONTOLOGY does not consider different levels of granularity during the reuse of ontologies (as for example, ontology statements).

When an ontology to be reused has to be modified, METHONTOLOGY proposes to carry out the ***ontology reengineering activity***. However, for this activity the methodology only mentions the main activities to be carried out, but without giving detailed guidelines. Such proposed activities are shown in Figure 3.

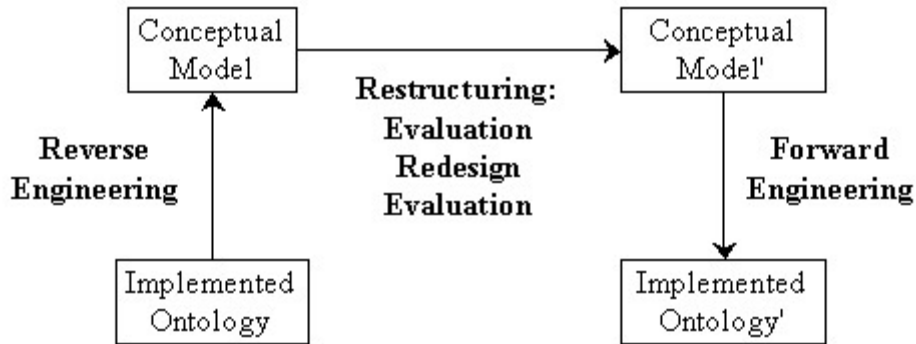


Figure 3. Ontology Reengineering Activities [61]

In the context of reusing and reengineering, METHONTOLOGY does not consider the **reuse and reengineering of non ontological resources**, neither the **reuse of ontology design patterns**.

Taking into account the important dimensions considered in the NeOn project, we can say that METHONTOLOGY does not mention anything about **collaboration** and **context**. Although some mention about the **dynamic** dimension is made, no detailed guidelines about how to manage different versions are given.

The main METHONTOLOGY contributions to the area were:

- Identification of the ontology development process.
- Identification of the life cycle.
- Detailed guidelines for building ontologies from scratch.

However, its main limitation is that the methodology is not **targeted to software developers and ontology practitioners**, but towards ontology engineers and researchers.

Finally, it is important to mention that this methodology enables the construction of ontologies at the knowledge level.

2.3. On-To-Knowledge

The aim of the On-To-Knowledge project [108] was to apply ontologies to electronically available information for improving the quality of knowledge management in large and distributed organizations. Some of the partners of this project were the Institute AIFB of the University of Karlsruhe, the Vrije Universiteit of Amsterdam, and British Telecom. In this project, they developed a methodology and tools for intelligent access to large volumes of semi-structured and textual information sources in intra-, extra-, and internet-based environments. The methodology includes a methodology for building ontologies to be used by the knowledge management application. Therefore, the **On-To-Knowledge methodology** for building ontologies proposes to build ontologies taking into account how they are going to be used in further applications.

Another important characteristic is that On-To-Knowledge proposes ontology learning for reducing the efforts made to develop the ontology.

The methodology also includes the identification of goals to be achieved by knowledge management tools, and is based on an analysis of usage scenarios [108].

The processes proposed by this methodology are shown in Figure 4 and can be summarized as follows:

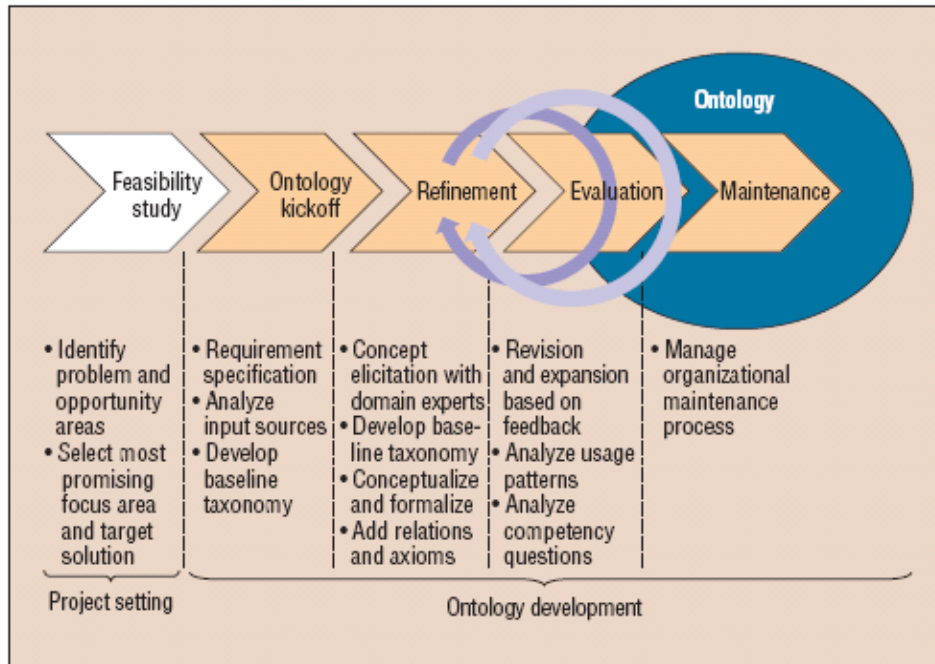


Figure 4. On-To-Knowledge Ontology Life Cycle [108]

Process 1. Feasibility study. On-To-Knowledge adopts the kind of feasibility study described in the CommonKADS methodology [102]. According to On-To-Knowledge, the feasibility study is applied to the complete application and, therefore, should be carried out before developing the ontologies. In fact, the feasibility study serves as a basis for the kickoff process.

Process 2. Kickoff. The result of this process is the ontology requirements specification document that describes the following: the domain and goal of the ontology; the design guidelines (for instance, naming conventions); available knowledge sources (books, magazines, interviews, etc.); potential users and use cases as well as applications supported by the ontology. Another outcome of this process is a semi-formal draft description of the ontology.

On-To-Knowledge proposes competency questions (CQs) [67] for carrying out the **ontology specification activity**, however not detailed guidelines are provided for this activity.

CQs can be useful to elaborate the requirements specification document. The requirement specification should lead the ontology engineer to decide about the inclusion or exclusion of concepts in the ontology, and about their hierarchical structure. In fact, this specification is useful to elaborate a draft version containing few but seminal elements. This first draft is called “baseline ontology”. The most important concepts and relations are identified on an informal level.

In the kickoff process developers should look for potentially **reusable ontologies** already developed. Although this methodology mentions the identification of potential ontologies to be reused, it does not provide detailed guidelines for identifying such ontologies neither for reusing them. Apart from that, this methodology does not explicitly mention guides for the **reuse and reengineering of non ontological resources**, neither for the **reuse of ontology design patterns**.

Process 3. Refinement. The goal here is to produce a mature and application oriented “target ontology” according to the specification given in the kickoff process. This refinement process is divided into two activities:

- *Activity 1: Knowledge elicitation process with domain experts.* The baseline ontology, that is, the first draft of the ontology obtained in process 2, is refined by means of interaction with domain experts. When this activity is performed, axioms are identified and modeled.

During the elicitation, concepts are gathered on one side and terms to label the concepts on the other. Then, terms and concepts are mapped.

The On-To-Knowledge methodology proposes the use of intermediate representations to model the knowledge. In this aspect, it follows METHONTOLOGY's basic ideas. If several experts participate in the building of the ontology, it is necessary to reach an agreement. A complementary way to enrich the ontology is to use it as seed in an ontology learning process.

- *Activity 2: Formalization.* The ontology is implemented using an ontology language. Such language is selected according to the specific requirements of the envisaged application.

To carry out the formalization, On-To-Knowledge recommends the use of the OntoEdit ontology editor, which automatically generates the ontology code in several languages. Other ontology editors that perform similar functions can be also used.

Process 4. Evaluation. The evaluation process serves as a proof of the usefulness of the developed ontologies and their associated software environment. The product obtained is called ontology based application. During this process two activities are carried out:

- *Activity 1: Checking the requirements and competency questions.* The developers check whether the ontology satisfies the requirements and "can answer" the competency questions.
- *Activity 2: Testing the ontology in the target application environment.* Further refinement of the ontology can arise in this activity.

This evaluation process is closely linked to the refinement process. In fact, several cycles are needed until the target ontology reaches the envisaged level.

Process 5. Maintenance. It is important to clarify who is responsible for the maintenance and how this should be carried out. On-To-Knowledge proposes to carry out ontology maintenance as part of the system software.

Such maintenance is related with the **dynamic** dimension within the NeOn project. In this case, this methodology proposes to create any new version after testing possible effects to the application. However, no guidelines are provided about how to manage different versions neither when to create new versions.

Related to the other important dimensions considered in the NeOn project, On-To-Knowledge does not consider neither **collaboration** nor **context**.

Finally, it is important to mention that ontologies developed with this methodology are highly dependent of the application. However, the methodology is not explained **targeted to software developers and ontology practitioners**.

2.4. DILIGENT

The **DILIGENT methodology** [90] is intended to support domain experts in a distributed setting to engineer and evolve ontologies. This methodology is focused on **collaborative** ontology engineering, and the central issue is to keep track of the change arguments. However, the notion of **context** is not considered by the methodology.

The ontology development process proposed by this methodology includes the following five main activities [40, 91]:

1. **Build.** In the build phase, domain experts, users, knowledge engineers and ontology engineers collaboratively create an initial version of the ontology. The team involved in building the initial ontology should be relatively small, in order to more easily find a small and consensual first version

of the shared ontology. Completeness of the initial shared ontology with respect to the domain is not required.

For building this initial version of the ontology, DILIGENT does not propose to carry out the **ontology specification activity** neither to take into account **reuse and reengineering of existing knowledge resources**.

2. Local adaptation. Once the shared ontology is made available, users can start using it and locally adapting it for their own purposes. In their local environment they are free to change the local copy of the shared ontology. Local changes do not affect other users of the ontology. All local changes of the shared ontology are collected by a central control board. They are seen as change requests to the shared ontology.

3. Analysis. During this phase, the control board analyzes all changes that were done in the previous phase by the users or stakeholders to their local copies of the shared ontology. It then decides which of the local changes will be introduced in the next version of the shared ontology. For this purpose, the control board tries to identify similarities in users' ontologies because not all user ontologies should be merged. Instead, it is the goal of this phase to develop a core shared ontology because otherwise its size will grow fast and it will become unmaintainable.

Regarding **dynamic** dimension, DILIGENT proposes the creation of different versions of the ontology, but does not provide guidelines on how to manage such versions neither on when to create different versions, nor how such changes can affect to the different versions.

4. Revision. Based on the previous analysis phase, a new revision of the shared ontology is created and distributed. Regular revisions of the shared ontology are required, in order to avoid a larger divergence of the local ontologies from the shared ontology. Thus, a balanced decision has to be found. It should take into account the different needs of the users and their evolving requirements.

5. Local update. In the last step, users of the shared ontologies update their local copies to the latest revision of the shared ontology. The shared ontology may contain several of the changes that were introduced in the local adaptation phase, but other changes may be missing. Because the control board tries to balance the different needs of users, it will not always take over changes as they are. Thus, even if a previous change made it into the new revision of the shared ontology, it may contain differences. Nevertheless, the user should reuse the new concepts instead of using their previously locally defined concepts in order to benefit from the further development of the shared ontology.

All in all, the DILIGENT methodology proposes an ontology life cycle model that is based on the idea of evolving prototypes. The life cycle model is shown in Figure 5.

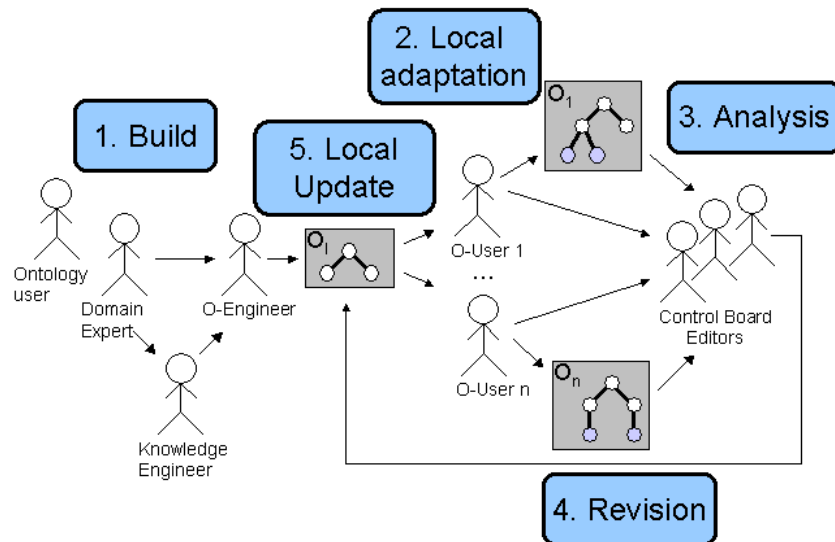


Figure 5. Life Cycle Model of the DILIGENT Methodology [40]

Central to the DILIGENT methodology is an argumentation framework. It facilitates discussions about the design rationale of changes that are introduced in the different phases of the life cycle. Especially in the analysis and revision phase, the exchanged arguments help the control board in understanding the reasons for specific changes [40].

But in the same way, the argumentation model may also be used by the control board for communicating the reasons of a decision to the users of the shared ontology. Otherwise, it would be difficult for users to understand why e. g. one change made it into the new revision of the shared ontology and another change did not. More details about the DILIGENT argumentation framework and how it facilitates collaborative ontology engineering are available in [38].

Finally, due to DILIGENT being intended to support domain experts, it is ***not targeted to software developers and ontology practitioners***.

2.5. Comparison of Presented Methodologies

In general we can say that METHONTOLOGY and On-To-Knowledge are up to now the most complete methodologies for building ontologies from scratch. They mainly include guidelines for single ontology construction from the ontology specification to the implementation.

Table 1 summarizes the presented methodologies (METHONTOLOGY, On-To-Knowledge and DILIGENT) according to the following characteristics:

- ❑ Critical dimensions within the NeOn project: collaboration, context and dynamics.
- ❑ Degree of coverage of the process or activities included in this deliverable by means of providing detailed guidelines.
- ❑ Targeted to software developers and ontology practitioners in general and not towards ontology researchers.

	METHONTOLOGY	On-To-Knowledge	DILIGENT
NeOn Dimensions			
Collaboration	Not mentioned	Not mentioned	Treated
Context	Not mentioned	Not mentioned	Not mentioned
Dynamic	Mentioned, but not treated	Mentioned, but not treated	Mentioned, but not treated
Detailed Guidelines for Processes and Activities			
Ontology Specification	Not provided Only Competency Questions are proposed	Not provided Only Competency Questions are proposed	Not provided In fact, this activity is not proposed by the methodology
Reusing Non Ontological Resources	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned
Reengineering Non Ontological Resources	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned
Reusing Ontologies	Not provided Only a list of activities to be carried out is proposed	Not provided Only recommendation of identifying ontologies to be reused is given	Not provided, neither explicitly mentioned
Reusing Ontology Design Patterns	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned
Audience			
Targeted to Software Developers and Ontology Practitioners	Targeted to ontology engineers and researchers	Not targeted to ontology engineers and researchers	Intended to domain experts and users

Table 1. Summary of Conclusions

Regarding **NeOn dimensions**, in the **collaboration** dimension, none of the analyzed methodologies consider distributed ontology engineering among heterogeneous and geographically distributed groups of domain experts and ontology practitioners. DILIGENT does it, but it only provides a rich argumentation framework in order to quickly proceed with building a single ontology and tracking all relevant discussions about the conceptualization activity [40]. In the **context** dimension, none of the presented methodologies treat with it. Finally, all analyzed methodologies mention the importance of versioning, and the problematic of management different versions. However, none of them provide guidelines for treating the **dynamic** and evolution of the ontology.

Table 1 shows that none of the analyzed methodologies provide **detailed guidelines for the process or activities** included in this deliverable, which are ontology specification, non ontological resource reuse, non ontological resource reengineering, ontological resources reuse, and ontology design patterns reuse. Based on this, we can say that the analyzed methodologies are more descriptive than prescriptive, because they do not provide instructions to carry out processes or activities.

As a final comment, none of the analyzed methodologies are described **targeted to software developers and ontology practitioners**.

Our aim within the NeOn project is to create the *NeOn methodology for building ontology networks* covering the drawbacks presented in the three analyzed methodologies, and benefiting from the advantages included in such methodologies, with respect to the aforementioned characteristics.

Concretely, regarding NeOn dimensions, the NeOn methodology will include the benefits provided by DILIGENT about collaboration. Furthermore, we will take into account the proposal given by METHONTOLOGY and On-To-Knowledge about the use of competency questions for the ontology

specification activity in the proposed methodological guidelines for this activity presented here. With respect to the reuse of ontologies, we will consider as starting point the list of activities proposed by METHONTOLOGY; we will improve and extend them to propose the corresponding methodological guidelines in the NeOn methodology.

3. Research Methodology

In this chapter, we present the research methodology used for building the NeOn methodology as well as the main requirements that guide its development.

3.1. General Framework for Describing the NeOn Methodology

For building the methodology we will use a “*divide and conquer*” strategy. That is we decompose the general problem to be solved in different subproblems. For each subproblem, we provide different strategies and alternatives to find the solution. To obtain the solution to the general problem, that is, the development of an ontology network, the solutions to the different subproblems are combined. In our case, the subproblems are the processes and activities identified in the 9 scenarios presented in section 4.1 and described in [149].

For obtaining the methodological guidelines associated to each process or activity, we grounded in the following approaches, as presented graphically in Figure 6.

- *Existing methodologies and methods.* In this case, we used METHONTOLOGY, On-To-Knowledge, DILIGENT and existing methods in order to provide guidelines for carrying out a process or an activity. This is the case of non ontological resource reengineering that had as a starting point the idea and activities proposed in the ontology reengineering method [61] presented in METHONTOLOGY.
- *Existing practices and previous experiences.* NeOn consortium members have built a lot of ontologies in different domains across several European and National funded projects. 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 complete methodological guidelines for each process or activity. As an example, we can mention the ontology specification activity, whose guidelines are based on previous experiences in the SEEMP project (FP6-27347).
- *Existing NeOn tools.* In this case, we used technology being developed in WP1 to WP4 within the NeOn project in order to provide guidelines for carrying out a specific process or activity. As an example, we can mention the ontology statement reuse process, whose guidelines are based on the use of the Watson [34, 35] NeOn plug-in in two different use cases.

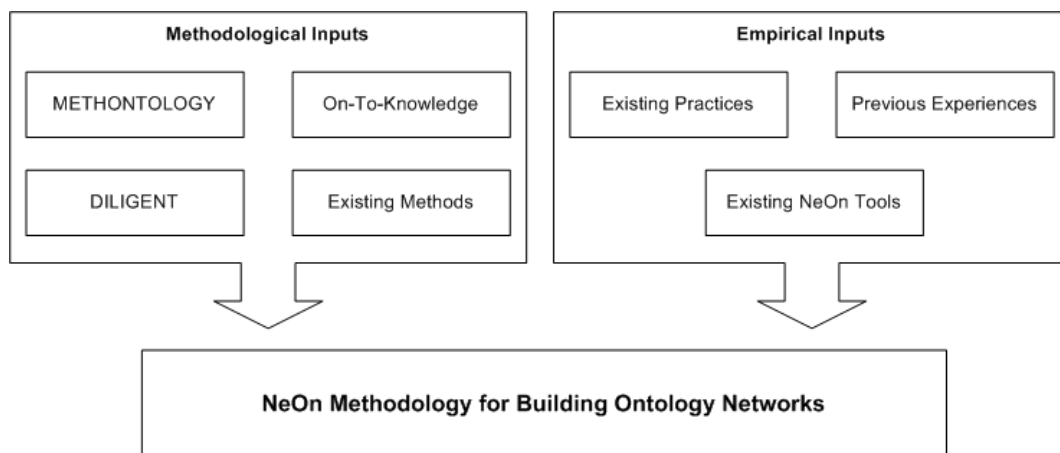


Figure 6. Inputs taken into account for obtaining the NeOn Methodology

Thus, the NeOn methodology for building ontology networks is based on a set of 9 scenarios, which can be combined between them. Eight out of nine were described in D5.3.1 [111]. A new scenario for building ontology networks by reusing ontology design patterns has been identified here. Each scenario is also decomposed in different processes or activities, and for each process or activity detailed methodological guidelines are provided. In this sense, the deliverable is written with a process or an activity centric approach, and in a more prescriptive way than prescriptive one.

Processes and activities included in this deliverable cover in a complete way, scenarios 2, 3 and 7, and in a partial manner scenario 1.

In this deliverable, we include a set of chapters describing methodological guidelines for carrying out different processes or activities in a subset of the identified scenarios. The activities included here are: ontology specification, non ontological resource reuse, non ontological resource reengineering, ontological resource reuse, and ontology design patterns reuse. In the case of non ontological resource reuse and reengineering, we describe them in a unique chapter because of non ontological resource reengineering can not occur without the non ontological resource reuse.

It is important to mention here, that based on the terminology included in section 2.1, some activities included in the NeOn Glossary [111] can be considered as processes composed of activities. This is the case of non ontological resource reuse, non ontological resource reengineering, and ontological resource reuse, which are now processes composed of a set of activities⁴. Activities can be divided into zero or more tasks. Tasks, which are the smallest unit of works, are used to decompose activities and provide more detailed information about the activities. Within this deliverable, we use this terminology, which is also shown in Figure 7.

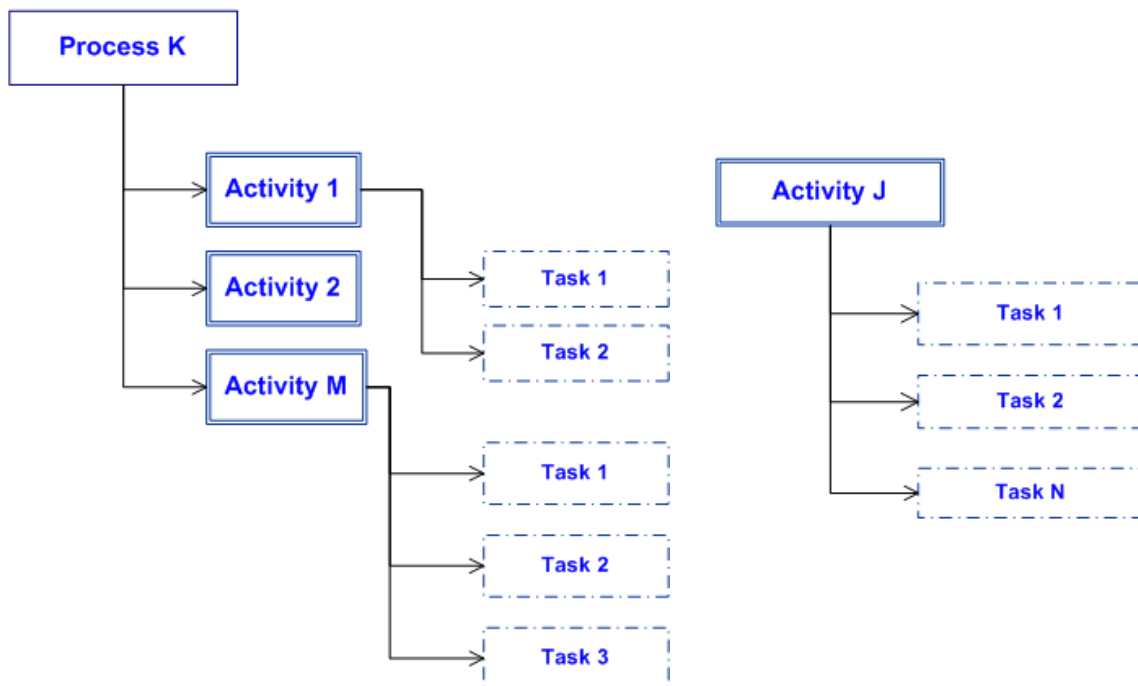


Figure 7. Process, Activities and Tasks

For describing each process and activity included in the NeOn methodology presented here, we use the following template:

- Introduction.

⁴ The next version of the NeOn Glossary of Activities will be called NeOn Glossary of Processes and Activities.

- ❑ State of the art, including methods, techniques, and tools, and conclusion about related works.
- ❑ Proposed detailed guidelines for carrying out the process or the activity, including:
 - Definition, taken from the NeOn Glossary of Activities [111] or new ones updated in this deliverable.
 - Goal, explaining the main objective intended to achieve by the process or the activity.
 - Input, which includes the resources needed for carrying out the process or the activity.
 - Output, which includes the results obtained after carrying out the process or the activity.
 - Who, which identifies people or teams involved in the process or the activity.
 - When, explaining in which moment the process or the activity should be carried out.
 - How, including 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 inputs, outputs and actors involved. Additionally, methods, techniques and tools supporting the process or the activity are proposed.

For each process and activity included in this deliverable, we provide a *filling card* including all the aforementioned information, by the exception of the “how”. The use of such filling cards allows us to explain the information of each process and activity in the NeOn methodology in a practical and easy way. Each filling card follows the *filling card* template shown Table 2.

Process or Activity Name	
<i>Definition</i>	
<i>Goal</i>	
<i>Input</i>	<i>Output</i>
<i>Who</i>	
<i>When</i>	

Table 2. Template for Process and Activity Filling Card

- Examples explaining the proposed guidelines using previous experiences and/or NeOn use cases, whenever it has been possible.

3.2. Conditions for the NeOn Methodology for Building Ontology Networks

According to [87], any methodology must fulfill a set of conditions that can be group into two main types, that is, the necessary and sufficient conditions. The **necessary conditions** (also called formal conditions) of a methodology are independent of the domain where the methodology is applied. The **sufficient conditions** (also called material conditions) of a methodology are specific to each methodology and are determined by factors such as domain where the methodology is applied, cases, situations or problems it deals with, characteristics of the material (economic, technological, etc.), human or temporal resources, etc.

In this section we present the necessary and sufficient conditions taken into account in the development of the NeOn methodology for building ontology networks. We based on Paradela's conditions [87], adapting them whenever it was necessary.

3.2.1. Necessary Conditions

- **Generality.** A methodology should be general enough and should not be driven to solve ad-hoc cases or problems.

In our case, the NeOn methodology treats the development of ontology networks in general, by means of proposing different scenarios for building networks of ontologies.

- **Completeness.** A methodology must consider all the cases presented and propose solutions to all of them.

In our case, the NeOn methodology for building ontology networks will deal with the 9 scenarios, from which 8 were identified in [111] and one, in the present deliverable. This first version only includes a subset of the possible cases that are, scenario1, scenario 2, scenario 3, and scenario 7.

- **Effectiveness.** A methodology should solve adequately the proposed cases that have a solution, with independency of the person that uses it. So, the methodology should be more prescriptive than descriptive.

In our case, we will describe the NeOn methodology in a simple way, and any person (being a software developer or an ontology practitioner) will able to understand and use it with no special effort.

- **Efficiency.** A methodology must be efficient, that is, be able to achieve its objective/goal. This means that the methodology should allow the construction of ontologies with fewer resources (time, money, etc.) and with better quality.

In our case, whenever it is possible, we describe and carry out experiments using the methodology with the goal of confirming its efficiency.

- **Consistency.** A methodology must produce the same set of results (semantically speaking) for the same problem, independently of who carries it out.

In our case, the NeOn methodology identifies which should be the outputs of the different activities involved in the development of ontology networks. Semantically speaking, the same set of outputs will be obtained after applying the methodology for a given case.

- **Finiteness.** The number of the elements that compose a methodology and the number of activities must be finite, i.e., consuming a reasonable period of time.

In our case, the NeOn Glossary of Activities [111] includes the initial and finite set of activities involved in the methodology. The number of elements used to describe the process or the activities are also finite.

- **Discernment.** A methodology must be composed of a small set of structural, functional and representational components.

In our case:

- Regarding structural components, the methodology provides a set of scenarios for building ontology networks. Such scenarios are a mix between heterarchical and hierarchical structure.
 - Regarding functional components, the methodology includes processes, activities, tasks, inputs, outputs and restrictions.
 - Finally, with respect to representational components, the methodology provides a graphical representation for the scenarios and for describing each process or activity.
- **Environment.** Methodologies can be classified into scientific and technological ones. In scientific methodologies ideas are validated, and in technological ones artefacts are built. A technological methodology must consider the life cycle of the product that is guiding its development.

The NeOn methodology for building ontology networks can be considered as a technical one, because the main result after applying it should be a technological product, that is, an ontology network. Thus, it is needed to establish the life cycle for the ontology network.

The first version of the collection of ontology network life cycle models and guidelines for establishing a particular ontology network life cycle were included in D5.3.1 [111].

- **Transparency.** A methodology must be like a white box, allowing to know in every moment the active processes or activities that are being performed, who is performing them, etc.

In our case, we explicitly define the actors, inputs, and outputs of each activity covered by the methodology.

- **Essential Questions.** The following six questions: “what”, “who”, “why”, “when”, “where”, and “how” must be considered for each activity included in the methodology.

In our case, questions about why and where are not covered in the methodology. However, the rest of the questions are answered in detail in the methodology: the NeOn Glossary of Activities already presented in D5.3.1 [111] explains “what” each activity involved in the methodology refers to, and the methodological guidelines for the processes or activities included in this deliverable answer the rest of the covered questions.

3.2.2. Sufficient Conditions

- **Domain or Scope.** In our case, the NeOn methodology is for developing ontology networks, with special emphasis in the existence of multiple ontologies in ontology networks, the dynamic dimension treating the ontology evolution, the context dimension, and the collaborative ontology development.

- **Perspectives.** A methodology must facilitate its application following different approaches.

In our case, the methodology provides different scenarios for building ontology networks and allows combining them in different ways.

- **Understanding.** A methodology must be ease to understand and to learn in order to facilitate its success and its generalized use.

The NeOn methodology for building ontology networks is being explained with simple descriptions and graphical representations to be easily understood by both ontology practitioners and software developers in general.

- **Usability.** The degree of difficulty in using the methodology must be minimal.

The methodology is presented using a software engineering approach with different levels of complexity to facilitate a promptly assimilation.

- **Grounded on existing practices.** The NeOn methodology grounds on existing methodologies in the Software and Ontology Engineering fields.

- **Flexibility.** The NeOn methodology allows the adaptation to concrete needs and users, and will also allow the inclusion of new processes or activities involved in the development of ontology networks.

- **Tool-Independent.** A methodology must be independent of the existing technology.

The NeOn methodology is being developed with the aim of being tool independent. Of course, it will have a close relation with the NeOn toolkit and its plug-ins, but it can also be used with other tools as Protégé, Top Braid Composer, etc.

4. NeOn Methodology for Building Ontology Networks

As we mentioned before, the 1990s and the first years of this new millennium have witnessed the growing interest of many practitioners in approaches that support the creation and management as well as the population of single ontologies built from scratch. There are some methodological approaches (e.g., METHONTOLOGY [59], On-To-Knowledge [108], and DILIGENT [90]) that help develop ontologies from scratch. All these approaches have provoked a step forward by having transformed the art of constructing single ontologies into an engineering activity.

The development of ontologies in different international and national projects have revealed that there are different alternative ways or possibilities to build ontologies. Just to name a few of them, in the Esperonto⁵ project ontologies were built from scratch; in Knowledge Web⁶ the aligning and versioning of ontologies was treated as well as the use of best practices or patterns, related to W3C activities; in the SEEMP⁷ project the development of ontologies is based on the reuse of non ontological resources; the SEKT⁸ project was focused on argumentative development of ontologies using the DILIGENT methodology; in the UMLS Project [32] the experiences gained while transforming the UMLS[®] Semantic Network into OWL ontology are described; within the UK PRODIGY and Drug Ontology Projects [76] the transformation of tangled hierarchies, as e.g. such derived from ambiguous "broader than / narrower than" thesauri in library science, into formal ontologies is described, etc. Thus, it is not premature to affirm that a new ontology development paradigm is starting, whose emphasis is on the reuse and possible subsequent reengineering of knowledge aware resources, the collaborative and argumentative ontology development, and the building of ontology networks⁹, as opposed to custom-building new ontologies from scratch. In order to support and promote such reuse-based approach, new methods, techniques, and tools are needed.

The following briefly presents:

- ❑ An overview of the different ways or scenarios for building ontologies, presented in D5.3.1 [111], that will be reviewed and extended in D5.3.2.
- ❑ An analysis of how argumentation and collaboration issues are related to the different scenarios.
- ❑ An explication of when ontologies become ontology networks.

4.1. Scenarios for Building Ontology Networks

Based on the analysis of the three NeOn use cases, on the different studies carried out to revise the state of the art on ontology development, and on the building of ontologies in different international and national projects, we have detected that there are alternative ways or possibilities to build ontologies and ontology networks. These ways can be seen as different scenarios in the NeOn methodology for building ontology networks.

⁵ <http://www.esperonto.net>

⁶ <http://knowledgeweb.semanticweb.org>

⁷ <http://www.seemp.org/>

⁸ <http://www.sekt-project.com/>

⁹ An ontology network or a network of ontologies is defined as a collection of ontologies (called networked ontologies) related together through a variety of different relationships such as mapping, modularization, version, and dependency relationships [70]

Figure 8 presents the set of 9 identified NeOn scenarios for building ontology networks: 8 of these scenarios were briefly described in [111] and one of them (scenario 7) has been recently identified. They are the following ones:

- ❑ *Scenario 1*: Building ontology networks from scratch without reusing existing knowledge resources.
- ❑ *Scenario 2*: Building ontology networks by reusing and reengineering non ontological resources.
- ❑ *Scenario 3*: Building ontology networks by reusing ontological resources.
- ❑ *Scenario 4*: Building ontology networks by reusing and reengineering ontological resources.
- ❑ *Scenario 5*: Building ontology networks by reusing and merging ontological resources.
- ❑ *Scenario 6*: Building ontology networks by reusing, merging and reengineering ontological resources.
- ❑ *Scenario 7*: Building ontology networks by reusing ontology design patterns.
- ❑ *Scenario 8*: Building ontology networks by restructuring ontological resources.
- ❑ *Scenario 9*: Building ontology networks by localizing ontological resources.

The activities of knowledge acquisition and elicitation, documentation, configuration management, evaluation and assessment (as shown at the bottom of Figure 8) should be carried out during the whole ontology network development.

It is worth mentioning that these scenarios can be combined in different ways. For instance, scenario 2 (reusing and reengineering non ontological resources) can be combined with scenarios 3-8; and scenario 9 (localizing ontologies) can be carried out or not with scenarios 1-8. Although we think this set of scenarios covers the most plausible ways for building ontology networks, it can not be considered exhaustive.

From this set of scenarios, we can say that scenario 1 is the most typical one for building ontologies and ontology networks from scratch without reusing existing knowledge resources. However, more and more ontology developers build ontologies and ontology networks by means of reusing existing knowledge resources. The NeOn methodology distinguishes scenarios involving reuse¹⁰ of ontological resources from those involving reuse and reengineering of non ontological resources. In this version of the methodology, we have included a new scenario (scenario 7) for the reuse of ontology design patterns (ODPs), because ODPs are key elements during the ontology design and, therefore, deserve to be treated in a different scenario from that involving the other types of ontological resources.

In Figure 8, processes and activities to be carried out are represented by coloured circles or by rounded boxes. Directed arrows with numbered circles associated represent the different scenarios presented in this section. The figure also shows (as dotted boxes) existing knowledge resources to be reuse; and possible outputs (implemented ontology networks and alignments) that result from the execution of some of the presented scenarios.

¹⁰ *Reuse* in software engineering is defined [1] as “the use of an asset in the solution of different problems”, where an asset is “an item, such as design, specifications, source code, documentation, test suites, manual procedures, etc., that has been designed for use in multiple contexts”. Analogously, we can define *knowledge reuse* as the use of any knowledge resource¹⁰ (ontological and non ontological resources) in the solution of different problems, as for example, the building of new ontologies or the development of ontology-based applications.

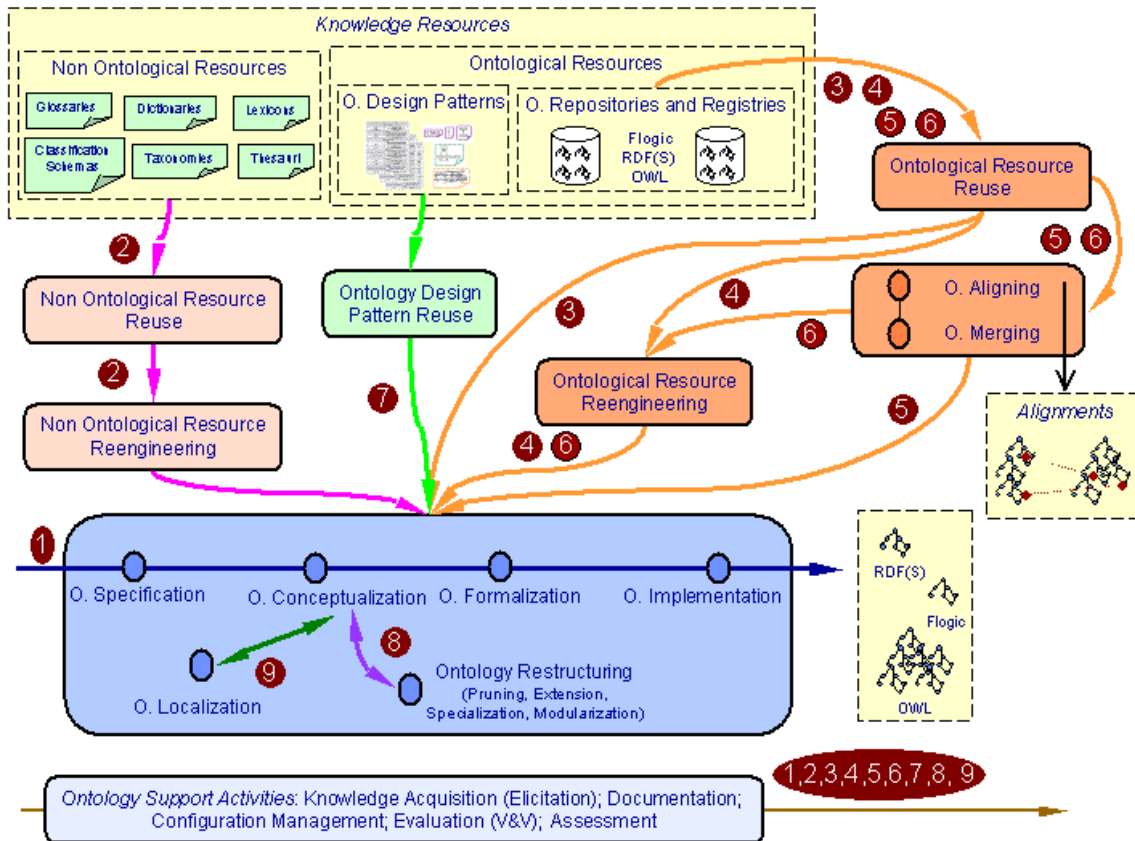


Figure 8. Scenarios for Building Ontology Networks

From the set of 9 scenarios shown in Figure 8, in this deliverable we provide guidelines for different activities involved in scenarios 1, 2, 3, and 7. The processes and activities considered in this deliverable are: ontology specification, non ontological resource reuse, non ontological resource reengineering, ontological resource reuse, and ontology design patterns reuse.

4.2. Argumentation and Collaboration in NeOn Scenarios

In general, one can distinguish two different cases in which argumentation plays an important role in enabling collaboration between the participants of an ontology engineering project:

- First, there are activities during which argumentation data is actively created, e.g. by discussions between the participants. In this case, the argumentation framework has the role of structuring the discussion process, helping in systematically exploring possible solutions and capturing the pro and contra arguments. Argumentation support is then a means of having more efficient discussion and decision taking processes (e. g. like it is the objective of the DILIGENT argumentation framework described in section 2.1.3).
- Second, there are activities where previously recorded discussions are used for understanding the design rationale of elements in the ontology network. For example, in the DILIGENT methodology the argumentation data created during the local adaptation of an ontology is used by the control board during the analysis and revision activity (cf. section 2.1.3). In this case, recorded discussions are part of the ontology documentation.

The most important activities of an ontology engineering project during which discussion data may be actively created are the ontology specification, ontology conceptualization, ontology

formalization and ontology implementation phases. All four activities require reaching a consensus between the participants about the requirements of the ontology network and how they should be implemented. But the recorded discussions may also be used for understanding the decisions made during previous activities (e.g. during ontology formalization one has to understand the decisions from the ontology conceptualization activity).

Obviously, reaching a consensus or explaining decisions to collaborators is not only needed during building an ontology from scratch. It may also be needed during reusing or reengineering ontological and non-ontological resources, or during the alignment with other ontologies.

Discussions are an important part of the ontology documentation, which should also refer to explanatory comments generated during the entire ontology building process. The recorded discussions help in keeping track of the design rationale all the way through the ontology engineering process, and keeping the design rationale up to date by amending it with additional arguments.

Recording discussions makes it easier to resume a previous activity in the ontology engineering process, if it turns out that a decision taken during that activity is underspecified or not appropriate. In this case, the discussion that led to the decision may be easily resumed because all stakeholders that participated in the decision taking process are identified by the recorded discussion. Resuming a discussion may additionally be useful during the maintenance phase of an ontology, e.g. if there were changes to the requirements that affected the decision.

In general, supporting the argumentation process is important in each situation where either several users collaboratively decide an issue or where a user by himself creates an ontology element that should be later used as input for the activity to be developed by another user. In the latter case, the collaboration is facilitated by enhanced and more complete ontology documentation.

4.3. When do Ontologies become Ontology Networks?

When software developers and ontology practitioners decide to use ontologies for solving a particular problem, the first activity to be carried out is an environment and feasibility study. This allow them to decide whether ontologies should be developed or not for the specific problem, and if so, they have to decide if for their problem it is better to build a single ontology, a set of interconnected single ontologies, or an ontology network.

- We have a *single ontology* when an ontology has not any type of relationship (domain dependent or independent) with other ontologies.
- We have a set of interconnected single ontologies if some kind of domain dependent ad-hoc relation exists between them.
- We have an ontology network, if there is a requirement or it is advisable to express: (a) metarelationships between the ontology to be developed and other existing ontologies available in the web, or (b) metarelationships between the ontology to be developed and its components. Examples of these metarelationships are:
 - *priorVersionOf*: if the ontology to be developed is a new version of an existing one.
 - *useImports*: if the ontology is importing any other ontology due to the fact that it consists of different knowledge domains
 - *extendingBy*: if the ontology is extending an existing one.
 - *composedbyModules*: if the ontology to be developed is composed of a number of modules.

- *haveMapping*: if some ontology components have mappings with other existing ontologies.

Thus, to create ontology networks what is needed is a set of defined metarelations between ontologies and between ontologies and their elements. These metarelations, such as “*priorVersionOf*”, “*useImports*”, “*isIncompatibleWith*”, are included in D1.1.2 [69].

In this case, the ontology to be developed is in constant relation with others in the network, what permits a fluent knowledge sharing and an easy enrichment of the network. Furthermore, ontology networks favour knowledge growth in the Internet, and thus, its sharing and spreading.

For clarifying what is the difference between an ontology and an ontology network, we provide here some examples.

- An isolated ontology O_1 is a single ontology, as it is shown in Figure 9 (a). If we have n single ontologies related among them by means of ad-hoc relations between concepts included in such ontologies, such set is considered as a set of interconnected single ontologies, as in Figure 9 (b) shows.
- If a new version (O_2) of O_1 is developed and the explicit metarelation “*priorVersionOf*” between O_1 and O_2 is established, then a network of ontologies has been created (shown Figure 9 (c)).
- Figure 9 (d) presents the ontology network associated to the interconnected single ontologies presented in Figure 9 (b), in which it has been explicitly expressed that ontology A imports ontologies B and C.

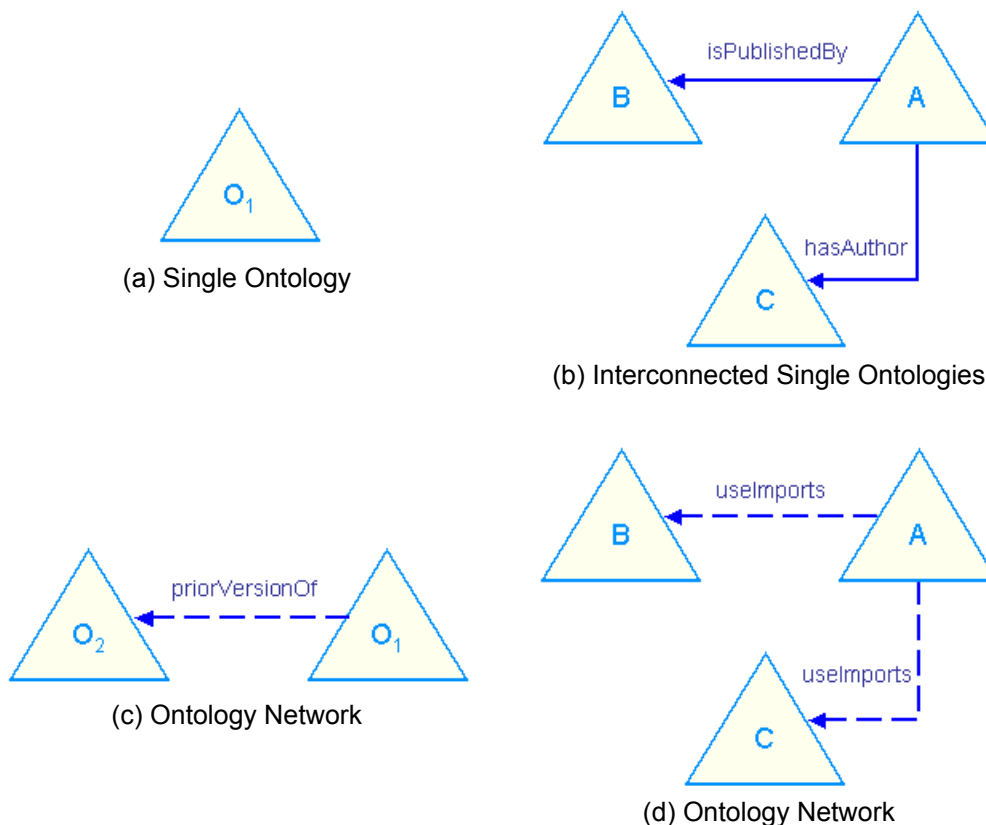


Figure 9. Simple Graphical Examples of Single Ontologies and Ontology Networks

In summary, the scenarios presented in section 4.1 permit software developers and ontology practitioners to build single ontologies, interconnected single ontologies and ontology networks [70]. If software developers and ontology practitioners explicitly define metarelationships such as mapping, modularization, version, and dependency, between a set of ontologies and/or between an ontology and their components, then, an ontology network has been created.

However, there are at least three key aspects when using such scenarios:

- ❑ The identification of when an ontology network is better than a single ontology or a set of interconnected ontologies.
- ❑ The development of ontology networks is a more complex process, which has some specific features different from those for building single ontologies.
- ❑ The impact of the evolution of components in an ontology network, which is greater than in a single ontology.

5. Ontology Specification

Ontology Specification is defined in [111] as a collection of requirements that the ontology should fulfill. The output of this activity is the ontology requirements specification document (ORSD) that includes the purpose, level of formality and scope of the ontology, target group and intended uses of the ontology, and a set of requirements, which are those needs that the ontology to be built should cover.

In this chapter we present a brief introduction to the existing methods, techniques and tools for ontology specification. We also propose the NeOn methodological guidelines for carrying out the activity.

5.1. State of the Art

5.1.1. Methods

In this section we present existing general methods for carrying out the ontology specification activity.

METHONTOLOGY [59, 48, 19] proposes the goals of the ontology specification activity, however it does not propose any method for carrying out the activity.

Grüninger and Fox methodology [68], On-To-Knowledge methodology [108], and the Unified methodology [114] proposed the following steps for obtaining the so-called ontology requirements specification document:

- Identify the purpose of the ontology to be developed.
- Identify the intended uses and users of the ontology to be developed.
- Identify the set of ontology requirements that the ontology should satisfy after being formally implemented.

5.1.2. Techniques

There are different techniques that can be applied for collecting requirements. Examples of these techniques are: brainstorming, joint application development (JAD) [93], exploit scenarios and use cases using templates, interviews with users and domain experts, and competency questions.

Most of the existing methodologies or methods [68, 108, 77, 78, 114] and guides [86] for developing ontologies suggest to identify **competency questions** as a technique for establishing the ontology requirements. Competency questions (CQs) were proposed for the first time in [68]. They are defined as natural language questions that the ontology to be built should be able to answer.

The following presents how the different aforementioned methodologies and guides propose to carry out the ontology specification activity using competency questions.

- *Grüninger and Fox's methodology* [68, 59] is inspired by the development of knowledge based systems using first order logic. This methodology proposes identifying intuitively the main motivating scenarios, that is, possible applications in which the ontology will be used. Such scenarios describe a set of the ontology requirements that the ontology should satisfy after being formally implemented. In particular, such scenarios may be presented by industrial partners regarding problems they encounter in their enterprises. The motivating scenarios often

have the form of story problems or examples which are not adequately addressed by existing ontologies. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems. These solutions give a first idea of the informal intended semantics of the objects and relations that will later be included in the ontology.

Given the set of informal scenarios, a set of informal competency questions is identified to determine the scope of the ontology. Informal competency questions are those written in natural language to be answered by the ontology once the ontology is expressed in a formal language. These questions and their answers are both used to extract the main concepts and their properties, relations and formal axioms of the ontology. Competency questions and their responses play the role of a type of requirement specification against which the ontology can be evaluated.

Ideally, competency questions should be defined in a stratified manner, with higher level questions requiring the solution of lower level ones. It is not a well-designed ontology if all competency questions have the form of simple queries, that is, if the questions cannot be decomposed or composed into more specific or general questions, respectively. There should be questions that use the solutions to such simple queries. Specific competency questions can be composed into more general questions that are answered by composing answers associated to specific competency questions.

- *On-To-Knowledge methodology* [108, 59] mentions that competency questions can be useful to elaborate the requirements specification document. The requirement specification should lead the ontology engineer to decide about the inclusion or exclusion of concepts in the ontology, and about their hierarchical structure.
- *The Unified methodology* [114] proposes to identify the purpose of the ontology, in particular, identify and characterise the range of intended users, identify the uses for the ontology, identify (fairly general) motivating scenarios and competency questions, and produce a user requirements document for the target software system. After that, the methodology recommends to decide how formal the ontology needs to be. This is determined in large part by purpose and users of the ontology. Finally, this methodology proposes to identify the scope of the ontology by means of (a) creating the detailed scenarios that arise in the applications, or (b) using brainstorming to do a more thorough and accurate job of scoping.

Besides competency questions, this methodology allows the ontology developer to use other techniques for the gathering of ontology requirements, such as defining the detailed motivating scenarios, brainstorming and trimming.

- *The EXPLODE methodology* [77, 78] integrates ideas from the eXtreme Programming methodology, and it is particularly suitable for dynamic and open environments thanks to its focus on immediate feedback and evaluation. This methodology proposes to fetch the requirements of the system and to define the competency questions.
- *The “Ontology Development 101” guide* [86] proposes to determine the domain and scope of the ontology by answering a set of basic questions (“What is the domain that the ontology will cover?”, “For what are we going to use the ontology?”, “Who will use and maintain the ontology?”, etc.) and by identifying the ontology competency questions.

5.1.3. Tools

After analysing the state of the art, we realized that only one tool exists for supporting the creation of ontology requirements. The tool is called **OntoKick** [112] and allows the creation of the requirement specification document and the extraction of relevant structures for the building of the semi-formal ontology description.

OntoKick is an OntoEdit plug-in for supporting the collaborative generation of requirement specifications for ontologies. OntoKick allows the description of important aspects of the ontology, such as: domain and goal of the ontology, design guidelines, available knowledge sources (e.g. domain experts, reusable ontologies etc.), potential users, use cases, and applications supported by the ontology. This tool uses competency questions (CQ) to define requirements for an ontology. Each CQ defines a query that the ontology should be able to answer and, therefore, it defines explicit requirements for the ontology. OntoKick takes further advantage of using CQs to create an initial version of the semi-formal description of the ontology. Based on the assumption that each CQ contains valuable information about the domain of the ontology, OntoKick extracts relevant concepts and relations. Furthermore, OntoKick establishes and maintains links between CQs and concepts derived from them. This allows for better traceability of the origins of concept definitions in later stages.

5.1.4. Conclusion

As a conclusion we can mention that most of the analyzed methodologies propose simple methods for carrying out the ontology specification activity. The methods consist of high level steps that can be summarized as follows: identify purpose, uses and users for the ontology to be developed, and identify the set of requirements the ontology to be developed should fulfill. However, these methodologies do not provide detailed guidelines explaining how to carry out each step.

For gathering ontology requirements, most of the analyzed methodologies propose as a technique the use of competency questions.

Finally, we can mention that only one tool, called OntoKick, exists for helping people in the ontology specification activity.

5.2. Proposed Guidelines for Ontology Specification

As we mentioned before, the goal of the ontology specification is to state why the ontology is being built, what its intended uses are, who the end-users are, and what the requirements the ontology should fulfill are. For specifying the ontology requirements we will use the competency questions techniques proposed in [68]. Before identifying the set of competency questions, we will identify the purpose and scope of the ontology, its level of formality, its intended uses and its intended users.

The NeOn methodology proposes the filling card, presented in Table 3, for the ontology specification activity, including the definition, goal, inputs and outputs, who carries out the activity and when the activity should be carried out.

Ontology Specification	
<i>Definition</i>	
<p><i>Ontology Specification</i> refers to the activity of collecting the requirements that the ontology should fulfill, e.g. reasons to build the ontology, target group, intended uses, possibly reached through a consensus process.</p>	
<i>Goal</i>	
<p>The specification activity states why the ontology is being built, what its intended uses are, who the end-users are, and what the requirements the ontology should fulfill are.</p>	
<i>Input</i>	<i>Output</i>
<p>A set of ontological needs.</p>	<p>Ontology Requirements Specification Document (ORSD).</p>
<i>Who</i>	
<p>Software developers and ontology practitioners, who form the ontology development team (ODT), in collaboration with users and domain experts.</p>	
<i>When</i>	
<p>This activity must be carried out in parallel with the knowledge acquisition activity.</p>	

Table 3. Ontology Specification Filling Card

The tasks for carrying out the ontology specification activity can be seen in Figure 10. The result of this activity is the Ontology Requirements Specification Document (ORSD).

The NeOn methodology proposes a template for writing the ORSD that have the following slots, and that is shown in Table 4:

- ❑ Ontology Purpose, which includes the ontology aims.
- ❑ Ontology Scope, which includes the ontology coverage and granularity.
- ❑ Ontology Level of Formality, which includes the degree of formality of the ontology.
- ❑ Ontology Intended Users, which includes the main intended users for the ontology.
- ❑ Ontology Intended Uses, which includes the main scenarios in which the ontology will be used.
- ❑ Groups of Competency Questions (CQs) and their answers, including priorities.
- ❑ Pre-Glossary of Terms with their Frequencies.

Ontology Requirements Specification Document Template	
1	Purpose
	<i>“Software developers and ontology practitioners should include in this slot the purpose of the ontology”</i>
2	Scope
	<i>“Software developers and ontology practitioners should include in this slot the scope of the ontology”</i>
3	Level of Formality
	<i>“Software developers and ontology practitioners should include in this slot the level of formality of the ontology”</i>
4	Intended Users
	<i>“Software developers and ontology practitioners should include in this slot the intended users of the ontology”</i>
5	Intended Uses
	<i>“Software developers and ontology practitioners should include in this slot the intended uses of the ontology”</i>
6	Groups of Competency Questions
	<i>“Software developers and ontology practitioners should include in this slot the groups of competency questions and their answers, including priorities for each group”</i>
7	Pre-Glossary of Terms
	Terms
	<i>“Software developers and ontology practitioners should include in this slot the list of terms included in the CQs and their frequencies”</i>
	Objects
	<i>“Software developers and ontology practitioners should include in this slot a list of objects and their frequencies”</i>

Table 4. Template for the OSRD

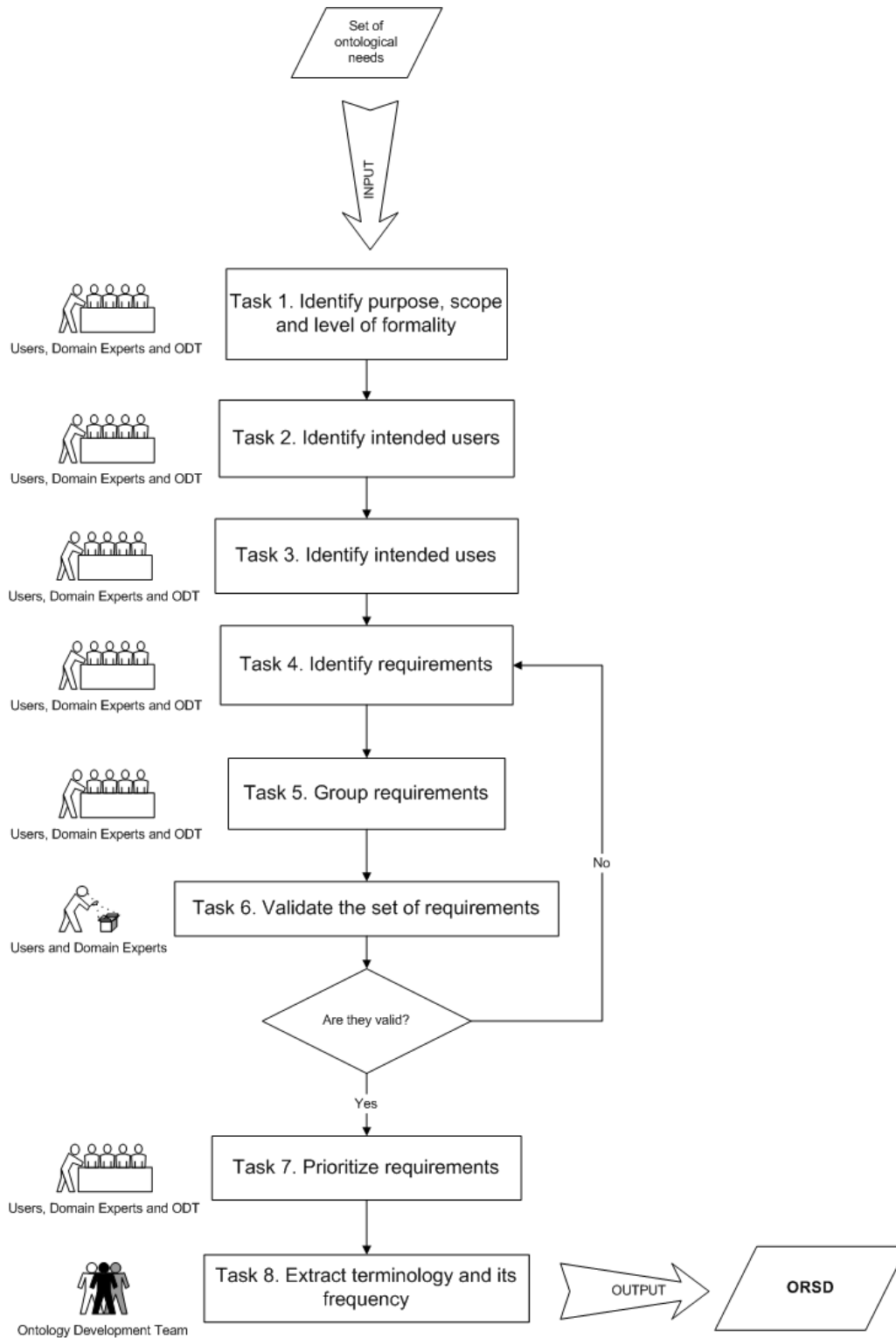


Figure 10. Tasks for Ontology Specification

The tasks for carrying out the ontology specification activity are explained in detail in the following:

Task 1. Identify purpose, scope and level of formality.

The objective of this task is to obtain the main goal or aim of the ontology, its coverage and granularity. The degree of formality to be used to codify the ontology should be also identified. This degree of formality ranges from informal natural language to a rigorous formal language. Users,

domain experts and the ontology development team carry out this task taking as input a set of ontological needs for obtaining the purpose, scope and level of formality of the ontology, using techniques as physical or virtual interviewers between them.

The task output is the purpose, scope and level of formality of the ontology, which will be included in the corresponding slots of the OSRD template.

Task 2. Identify intended users.

The goal of this task is to establish which are the main intended users of the ontology. Users, domain experts and the ontology development team carry out this task taking as input a set of ontological needs for identifying the intended users, using techniques as physical or virtual interviewers between them.

The task output is a list with the intended users, which will be included in the corresponding slot of the OSRD template.

Task 3. Identify intended uses.

The goal of this task is to obtain the main ontology intended uses, that is, in which kind of scenarios the ontology will be used. Users, domain experts and the ontology development team carry out this task taking as input a set of ontological needs for identifying the intended uses, using techniques as physical or virtual interviewers between them.

The development of an ontology is motivated by scenarios related to the application that will use the ontology. The task output is a list of intended uses in the form of scenarios. Such scenarios describe a set of the ontology's requirements that the ontology should satisfy after being formally implemented. The scenarios can be described in natural language or expressed in UML as use cases. The list of scenarios will be included in the corresponding slot of the OSRD template.

Task 4. Identify requirements.

The goal of this task is to obtain the set of requirements or needs that the ontology should fulfill. Users, domain experts and the ontology development team carry out this task taking as input a set of ontological needs for identifying the ontology requirements, using techniques as writing the requirements in natural language in the form of the so-called competency questions (CQs) and tools as mind map tools, excel, and collaborative tools.

The output of this task is a list of competency questions written in Natural Language and a set of answers for the CQs.

Different approaches for identifying competency questions can be applied, such as:

- ❑ Top-Down: Complex questions are decomposed in simple ones.
- ❑ Bottom-Up: Simple questions that are organised to form complex ones.
- ❑ Middle out: Mix approach between top-down and bottom-up.

Regarding the recommended tools, we can mention that MindMap tools allow to represent mind maps [26]. These mind maps are diagrams used to represent words, ideas, tasks or other items linked to and arranged radially around a central key word or idea. They are used to generate, visualize, structure and classify ideas. In general, a mind map provides information about a topic that is structured in a tree. Each branch of the tree is typically named and associatively refined by its subbranches. Icons and pictures as well as different colors and fonts might be used for illustration based on the assumption that our memory performance is improved by visual aspects. Many people from academia and industry are familiar with mind maps, and for this reason we think that this recommendation will be very useful for software engineering and ontology practitioners forming the ontology development team. Another advantage is that requirements visualization in form of a hierarchy is very intuitive and easy to understand and manage.

If people are geographically distributed, wiki tools, such as Cicero¹¹ [38], can be used for identifying the requirements, in the form of CQs and associated responses.

Task 5. Group requirements.

The goal of this task is to group the list of CQs into several categories. Users, domain experts and the ontology development team carry out this task taking as input the list of CQs written in natural language (obtained in task 4) for obtaining different groups of CQs, using techniques as Card Sorting, when the grouping is done manually, and Clustering NL sentences or Information Extraction when the grouping is done automatically; and using tools as MindMap Tools or Cicero Tool (for distributed teams).

The task output is a set of groups including different CQs.

To group the requirements is useful for guiding the ontology development based on different ontology modules or based on prototypes involving different features of the ontology.

Competency questions are grouped in such a way that each group includes those questions that are relevant to a specific feature of the ontology.

For grouping the requirements we proposed a hybrid approach that combines:

- ❑ The analysis of the frequency of terms and the grouping of CQs based on those terms that have a higher frequency.
- ❑ The use of pre-established categories, such as time and date, units of measure, currencies, location, languages, etc.

Task 6. Validate the set of requirements.

The goal of this task is to identify possible conflicts between CQs, missing CQs, and contradictions in CQs. Users and domain experts carry out this task taking as input the set of grouped CQs for deciding if such CQs are valid or not.

The task output is a confirmation about the validity of the set of CQs.

For validating the identified CQs, the following criteria are proposed:

- ❑ *Correctness*. Inspired on [4, 36], we can say that a set of requirements is correct if each requirement refers to some features of the ontology to be developed. That is, any requirement is necessary.
- ❑ *Completeness*. In [121], a requirement specification is considered as complete if no requirement is omitted. Practically and adapting this consideration to the ontology engineering field, we can say that if users and domain experts review the requirements and confirm that they do not know more necessary requirements, then the set of requirements can be considered complete.
- ❑ *Consistent*. The set of requirements can be considered internally consistent if no conflicts exit between requirements. Conflicts can be between terms (different terminology is used in the requirements to refer to the same need) and between characteristics (two or more requirements refer to contradictory features of the ontology to be developed).
- ❑ *Verifiable*. Based on [36, 4], we can say that the set of requirements is verifiable if each requirement is verifiable. That is, a finite process with a reasonable cost exists to test that the final ontology satisfies each requirement. A necessary condition to have a verifiable requirement is that such a requirement should be unambiguous.
- ❑ *Understandable*. Each requirement must be understandable by users and domain experts.

¹¹ <http://cicero.uni-koblenz.de/wiki>

- ❑ *No Ambiguity*. Based on [36, 4], we can say that an ontology requirement is unambiguous if it has only one interpretation.
- ❑ *Conciseness*. Each and every requirement is relevant, and there are no duplicated or irrelevant requirements.
- ❑ *Realism*. Requirement meanings must make sense in the domain.
- ❑ *Modifiable*. Based on [36, 4], we can say that a set of requirements is modifiable if its structure and style allow to change issues in an easy, complete and consistent way.
- ❑ *Traceable*. Based on [36, 4], we can say that an ontology requirement is retraceable if its origin is known and it can be referred to in other documents during the ontology development. A necessary condition to have retraceable requirements is that such requirements should be referred in a unique way (normally using a kind of code).

Task 7. Prioritize requirements.

The goal of this task is to give different levels of priority to the different groups of CQs, and within each group to the identified requirements (in the form of CQs). Users, domain experts and the ontology development team carry out this task taking as input the groups of CQs written in natural language (obtained in task 5) for obtaining the priorities for each group and for each CQs within a group.

The task output is a set of priorities attached to each group of CQs and to each CQ in a group.

Priorities in CQs will be used for planning the ontology development.

This task is optional, but recommended. In fact, if no priorities are given to the groups of CQs, the ontology development will model all requirements at the same time.

Task 8. Extract terminology and its frequency.

The goal of this task is to extract from the list of CQs a pre-glossary to be used in the conceptualization activity. The ontology development team carries out this task taking as input the list of identified CQs and their answers for obtaining a list of the most used terms in them, using terminology extraction techniques and tools supporting such techniques.

From the requirements in form of competency questions, we extract the terminology (names, adjectives and verbs) that will be formally represented in the ontology by means of concepts, attributes and relations.

From the answers to the CQs we extract the objects in the universe of discourse that will be represented as instances.

5.3. Examples

In this section we include three different examples of how to use the proposed guidelines for the ontology specification activity and what results are expected.

The first example refers to the specification of the SEEMP reference ontology, by means of using the guidelines proposed in this deliverable. It is important to mention that the work done within the SEEMP project in the ontology specification activity has been one of the inputs to get preliminar guidelines for this activity. Such preliminar guidelines have been extended, improved, and proposed in this deliverable. Using the proposed guidelines presented here, we described the specification activity with the SEEMP reference ontology.

The remaining two examples instantiate the guidelines for the ontology specification proposed here in the invoice use case and in the nomenclature use case within the NeOn project. It is worth to mention that previous to this deliverable both uses cases presented the ontology specification in D8.3.1 [65] using preliminar guidelines for carrying out the activity.

5.3.1. SEEMP Reference Ontology Specification

The main objective of the SEEMP¹² project is to develop an interoperable architecture for public e-Employment services (PES). The resultant architecture will consist of: a Reference Ontology, the core component of the system, that acts as a common “language” in the form of a set of controlled vocabularies to describe the details of a job posting; a set of Local Ontologies, each PES uses its own Local Ontology, which describes the employment market in its own terms; a set of mappings between each Local Ontology and the Reference Ontology; and a set of mappings between the PES schema sources and the Local Ontologies. The SEEMP project relies on WSMO [45] that permits to semantically describe Web Services, ontologies and mediators. WSML [37] is the concrete language used in SEEMP for encoding those descriptions.

In this section we present the specification of the SEEMP Reference Ontology following the proposed guidelines of the NeOn Methodology. This specification is not intended to be exhaustive, but it just describes the most important points. A detailed and complete specification is described in [13]. Next we described the steps we followed:

Task 1. Identify purpose, scope and level of formality.

The development of the Reference Ontology is motivated by scenarios related to the application that will use the ontology. Such scenarios describe a set of the ontology requirements that the ontology should satisfy after being formally implemented. The motivating scenarios are described in [14]. In summary, the purpose of building the Reference Ontology is to provide a consensual knowledge model of the employment domain that could be used by public e-Employment services (PES), more specifically within the ICT (Information and Communication Technology) domain. Since SEEMP project relies on WSMO, the implementation language of the resultant ontology will be WSML.

Task 2. Identify intended users.

As it was mentioned before, the Reference Ontology will be the core component of the SEEMP platform; the peers on the SEEMP interoperate with each other from their local ontologies via the Reference Ontology. The analysis of the motivating scenarios described in [3], allowed us to identify the following intended users of the ontology:

- User 1. Candidate who is unemployed and searching for a job or searching another occupation for immediate or future purposes
- User 2. Employer who needs more human resources.
- User 3. Public or private employment search service which offers services to gather CVs or job postings and to prepare some data and statistics.
- User 4. National and Local Governments which want to analyze the situation on the employment market in their countries and prepare documents on employment, social and educational policy.
- User 5. European Commission and the governments of EU countries which want to analyze the statistics and prepare international agreements and documents on the employment, social and educational policy.

Task 3. Identify intended uses.

The analysis of the motivating scenarios described in [14], allowed us to identify the following main intended uses of the ontology:

- Use 1. Publish CV. Job seeker places his/her CV on the PES Portal.

¹² <http://www.seemp.org>

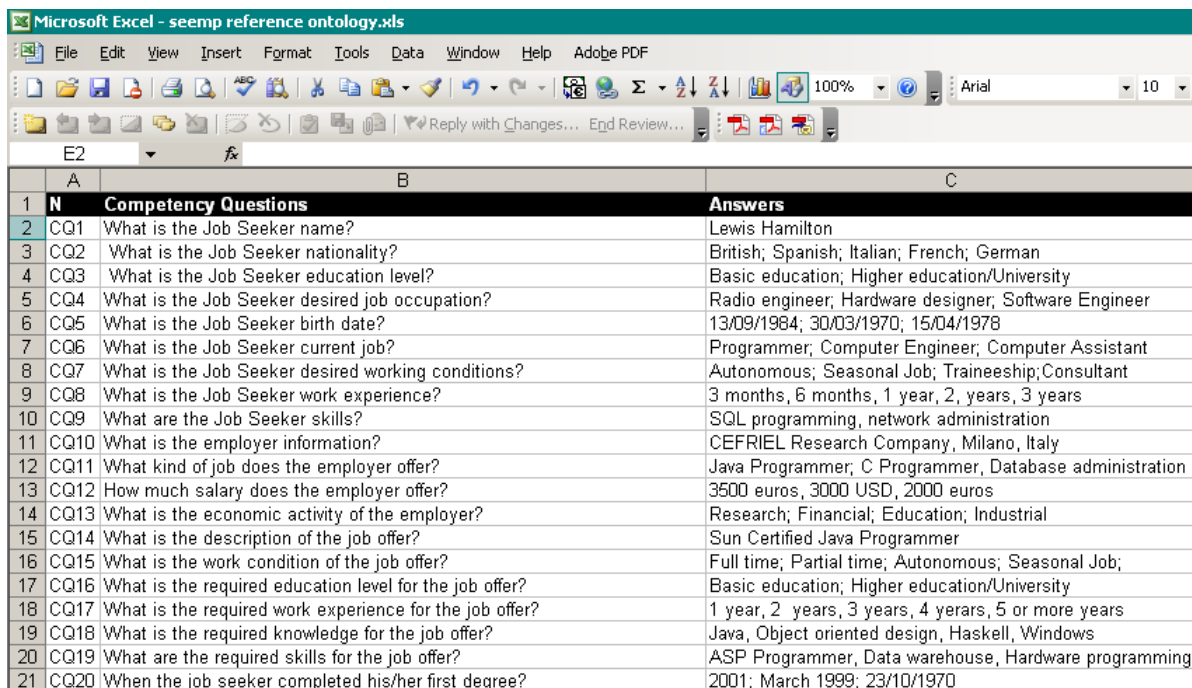
- Use 2. Publish Job Offer. An Employer places a Job Offer on the PES Portal.
- Use 3. Search for Job Offers. The Employer looks for candidates for the Job Offer through PES Portal.
- Use 4. Search for Employment information. Job Seeker looks for of general information about employment in a given location at the PES Portal.
- Use 5. Provide Job Statistics. The PES Portal provides employment statistics to the Job Seeker and Employer.

Task 4. Identify requirements.

For specifying the ontology requirements we used the competency questions techniques. We followed the *bottom up* approach for identifying them. Competency questions were stored in an *Excel file* and then rewritten in a mind map tool as appears in Figure 11 and Figure 12, respectively.

In total we identified sixty competency questions, which are described in detail in [14]. Examples of some competency questions are:

- What is the job seeker nationality?
- What is the job seeker desired job?
- What is the required work experience for the job offer?
- When did the job seeker complete his/her first degree?
- What is the job seeker education level?
- Is the offered salary given in Euros?



The screenshot shows a Microsoft Excel spreadsheet titled "seemp reference ontology.xls". The spreadsheet contains a table with two columns: "Competency Questions" and "Answers". The table lists 20 examples of competency questions and their corresponding answers.

	A	B	C
1	N	Competency Questions	Answers
2	CQ1	What is the Job Seeker name?	Lewis Hamilton
3	CQ2	What is the Job Seeker nationality?	British; Spanish; Italian; French; German
4	CQ3	What is the Job Seeker education level?	Basic education; Higher education/University
5	CQ4	What is the Job Seeker desired job occupation?	Radio engineer; Hardware designer; Software Engineer
6	CQ5	What is the Job Seeker birth date?	13/09/1984; 30/03/1970; 15/04/1978
7	CQ6	What is the Job Seeker current job?	Programmer; Computer Engineer; Computer Assistant
8	CQ7	What is the Job Seeker desired working conditions?	Autonomous; Seasonal Job; Traineeship; Consultant
9	CQ8	What is the Job Seeker work experience?	3 months, 6 months, 1 year, 2 years, 3 years
10	CQ9	What are the Job Seeker skills?	SQL programming, network administration
11	CQ10	What is the employer information?	CEFRIEL Research Company, Milano, Italy
12	CQ11	What kind of job does the employer offer?	Java Programmer, C Programmer, Database administration
13	CQ12	How much salary does the employer offer?	3500 euros, 3000 USD, 2000 euros
14	CQ13	What is the economic activity of the employer?	Research; Financial; Education; Industrial
15	CQ14	What is the description of the job offer?	Sun Certified Java Programmer
16	CQ15	What is the work condition of the job offer?	Full time; Partial time; Autonomous; Seasonal Job;
17	CQ16	What is the required education level for the job offer?	Basic education; Higher education/University
18	CQ17	What is the required work experience for the job offer?	1 year, 2 years, 3 years, 4 years, 5 or more years
19	CQ18	What is the required knowledge for the job offer?	Java, Object oriented design, Haskell, Windows
20	CQ19	What are the required skills for the job offer?	ASP Programmer, Data warehouse, Hardware programming
21	CQ20	When the job seeker completed his/her first degree?	2001; March 1999; 23/10/1970

Figure 11. Excerpt of the Competency Questions and Answers in an Excel File

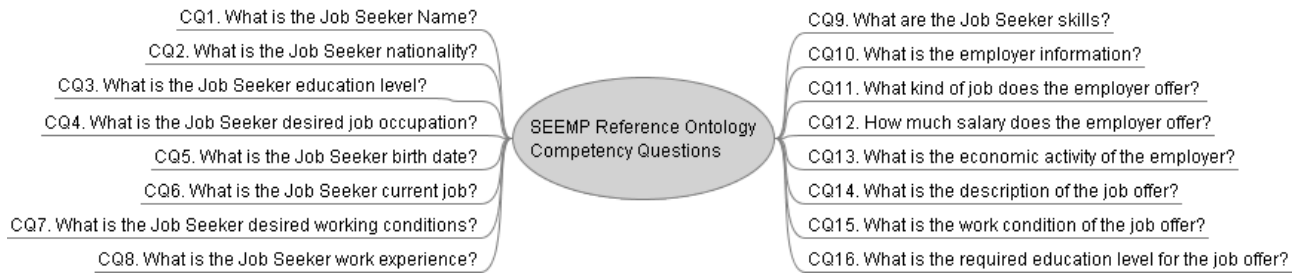


Figure 12. Excerpt of the Competency Questions in a Mind Map Tool

Task 5. Group requirements.

The sixty competency questions, described in [14], were manually grouped into five groups with the domain experts' help. Figure 13 shows the final 5 groups: Job Offer, Job Seeker, Currencies, Time and Date, and general ones. General competency questions are the result of the composition of simple queries into complex ones. The criteria for grouping the competency questions are based on the identified uses, the identified users and the domain expert suggestions. Figure 14 shows the 5 groups with some competency questions.

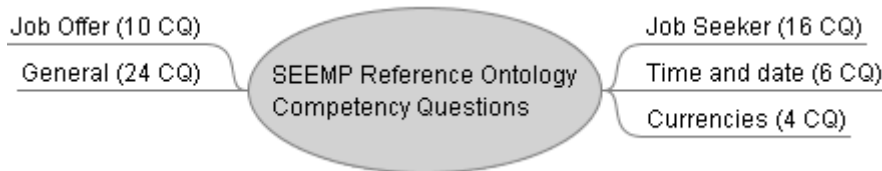


Figure 13. Competency Questions Groups

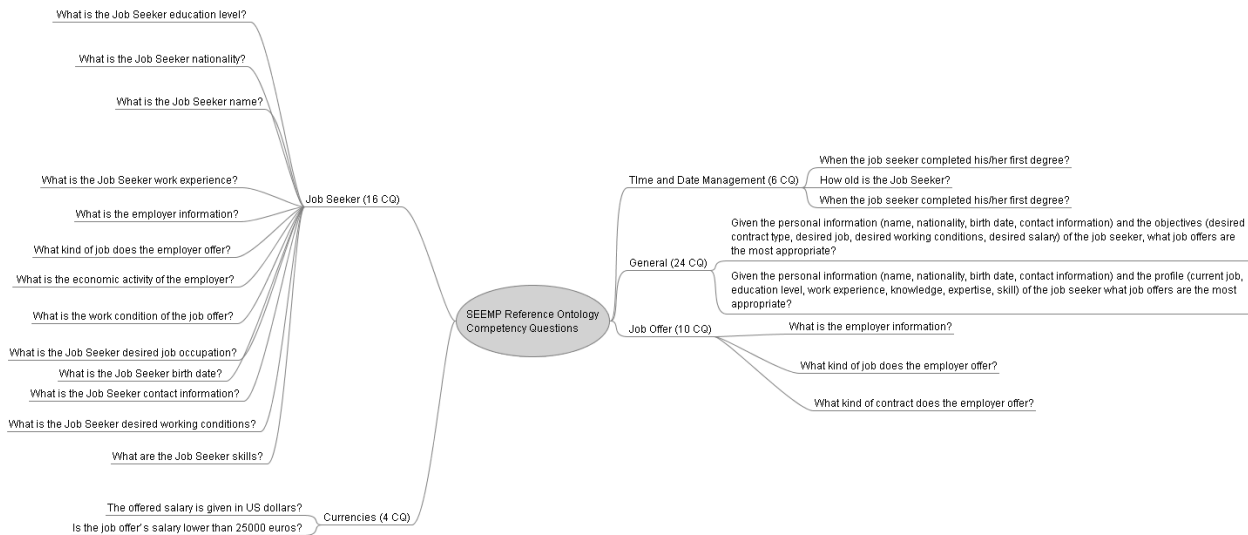


Figure 14. Competency Questions Groups in detail

Task 6. Validate the set of requirements.

During the overall process we received recommendations, suggestions and advices from the domain experts, and we iterated several times until we got the final approval by the end users. They used the following criteria for validating the competency questions:

- ❑ Correctness. Domain experts checked the correctness of each competency question, verifying that its formulation and answers were correct.

- Consistent. Domain experts also verified that the competency questions did not have any possible inconsistency. For example, a Job Seeker who does not speak English cannot find a job offer in England.

Task 7. Prioritize requirements.

Within the SEEMP Reference Ontology specification we did not carry out this step. This means the first version of the ontology must be able to represent the knowledge contained in all the competency questions.

Task 8. Extract terminology and its frequency.

From the competency questions, we manually extracted the terminology that will be formally represented in the ontology by means of concepts, attributes and relations. We identified the terms and the objects in the universe of discourse.

Examples of the terms related to job seeker are shown in Table 5.

Term	Frequency
Job Seeker	27
• CV	2
• Personal Information	3
Name	4
Gender	1
Birth Date	1
Address	1
Nationality	1
Contact (phone, fax, mail)	3
• Objective	3
Job Category	3
Activity Sector	3
Location	3
Work Condition	2
Contract type	2
Salary	3
• Education and training	3
• Work Experience	3
• Competencies	3
Knowledge	3
Abilities	3
Skills	3
• Publication	1
• Hobbies	1
• References	1

Table 5. Examples of Terminology and Frequency

Table 6 shows some examples of objects, which are instances of Nationality, Job Category, Education, Currency, Languages, and Activity Sector.

Nationality	Job Category	Education	Currency	Languages	Activity Sector
Austrian	Computer System Designer	Life Science	Euro	Austrian	Telecommunication
Belgian	Computer System Analyst	Mathematics	Krone	Belgian	Justice and Judicial
Cypriot	Programmer	Computer Science	Great British Pound	Cypriot	Public Security and law
Czech	Computer Engineer	Computer Use	Zlote	Czech	Manufacture of machine tools
Danish	Computer Assistant	Statistics	US Dollar	Danish	Research and Development
Estonian	Computer Equipment Operator	Physics	Franc	Estonian	Hardware Consultancy
Finnish	Industrial Robot Controller	Chemistry	Peso	Finnish	Software Consultancy and Supply
French	Telecommunication Equipment Operator	Earth Science		French	Data processing
German	Medical Equipment Operator	Network Administration		German	Database
Greek	Electronic Equipment Operator	Operating Systems		Greek	Publishing of Software
Hungarian	Image Equipment Operator	Informatics		Hungarian	Maintenance of computing machinery
Irish	Software Engineer	Programming Language		Irish	Government
Italian	Computer code recorder	Sports		Italian	Culture, Media, Design

Table 6. Examples of Objects

After following these tasks, the output of the Ontology Specification activity is the Ontology Requirements Specification Document. An excerpt of this document, which as been written for this deliverable, is shown in Table 7.

SEEMP Reference Ontology Requirements Specification	
1 Purpose	The purpose of building the Reference Ontology is to provide a consensual knowledge model of the employment domain that could be used by public e-Employment services (PES).
2 Scope	The ontology has to focus just on the ICT (Information and Communication Technology) domain. The level of granularity is directly related to the competency questions and terms identified.
3 Level of Formality	The ontology has to be implemented in WSML language
4 Intended Users	User 1. Candidate who is unemployed and searching for a job or searching another occupation for immediate or future purposes User 2. Employer who needs more human resources. User 3. Public or private employment search service which offers services to gather CVs or job postings and to prepare some data and statistics.

	<p>User 4. National and Local Governments which want to analyze the situation on the employment market in their countries and prepare documents on employment, social and educational policy.</p> <p>User 5. European Commission and the governments of EU countries which want to analyze the statistics and prepare international agreements and documents on the employment, social and educational policy.</p>		
5	Intended Uses		
	<p>Use 1. Publish CV. Job seeker places his/her CV on the PES Portal.</p> <p>Use 2. Publish Job Offer. An Employer places a Job Offer on the PES Portal.</p> <p>Use 3. Search for Job Offers. The Employer looks for candidates for the Job Offer through PES Portal.</p> <p>Use 4. Search for Employment information. Job Seeker looks for of general information about employment in a given location at the PES Portal.</p> <p>Use 5. Provide Job Statistics. The PES Portal provides employment statistics to the Job Seeker and Employer.</p>		
6	Groups of Competency Questions		
CQG1. Job Seeker (16 CQ)			
CQG2. Job Offer (10 CQ)			
CQG3. Time and Date (6 CQ)			
CQG4. Currencies (4 CQ)			
CQG5. General (24 CQ)			
7	Pre-Glossary of Terms		
	<table border="0" style="width: 100%;"> <tr> <td style="width: 50%;">Terms</td> <td style="width: 50%;">Frequency</td> </tr> </table>	Terms	Frequency
Terms	Frequency		

a.	Job Seeker	27
b.	CV	2
c.	Personal Information	3
d.	Name	4
e.	Gender	1
f.	Birth date	1
g.	Address	1
h.	Nationality	1
i.	Contact (phone, fax, mail)	3
j.	Objective	3
k.	Job Category	3
Objects		
Objects in the universe of discourse, which are instances of Job Category		
<ul style="list-style-type: none"> O1. Computer System Designer O2. Computer System Analyst O3. Programmer O4. Computer Engineer O5. Computer Assistant O6. Computer Equipment Operator O7. Industrial Robot Controller O8. Telecommunication Equipment Operator O9. Medical Equipment Operator O10. Electronic Equipment Operator O11. Image Equipment Operator 		

Table 7. Excerpt of SEEMP Reference Ontology Requirement Specification Document

5.3.2. Invoice Reference Ontology Specification

The invoice reference ontology designed in the scope of the Pharmaceutical case study aims to solve the lack of interoperability between invoice emitters and receivers. By using this invoice reference ontology integrated with the invoicing prototype which is currently being developed the end users will be able to map their invoices with the reference ontology and eliminate the interoperability problem that currently exists between both set of actors.

For the specification of the Invoice Reference Ontology we followed the proposed guidelines of the NeOn Methodology. Next we described the steps we followed:

Task 1. Identify purpose, scope and level of formality.

In [63] the purpose of the invoice reference ontology was identified and in deliverables [64, 65] this purpose was refined. The scope of the invoice reference ontology is to cover the most important languages/standards of the electronic invoicing domain. Electronic invoicing languages/standards like EDIFACT or UBL should be present along with other proprietary solutions and all the concepts needed for representing any invoice that can be emitted by a company. An important amount of emitters of invoices are small to medium enterprises which can not afford to use the previous standards due to the high cost of installing them into their small systems. Therefore the invoice reference ontology allows the users to add their proprietary solutions into it. An example of proprietary solution is the model and the terminology of the PharmaInnova model. Process

modeling concepts are also included in the reference ontology for representing the invoicing workflow. The language implementation selected is OWL.

Task 2. Identify intended users.

In order to get a fully usable ontology it is necessary to know who is going to finally use the ontologies developed. In order to address this problem, competency questions regarding the intended users of the ontology are mandatory. In [63] a complete description of the users of these applications is presented:

- U1. User of the invoicing application who is going to model a new invoice.
- U2. User who emits invoices.
- U3. User who receives invoices.
- U4. User who administrates the invoicing system.
- U5. Developers of invoicing applications

Task 3. Identify intended uses.

The development of the network of ontologies is motivated by scenarios related to the application that will use the ontology. Such scenarios describe a set of the ontology's requirements that the ontology should satisfy after being formally implemented. The motivating scenarios are described in [63].

Task 4. Identify requirements.

For specifying the ontology requirements we wrote a list of competency questions. This list has been included in [65]. The approach we followed for identifying the competency questions was first creating simple questions and by composition derives other complex questions. The approach followed to identify them was the top down one. The competency questions main topics were based on the requirements presented in [63]. Based on those requirements the competency question list was written and afterwards grouped in category groups. The tool for gathering the competency questions was *MS Word*. Examples of the competency questions are:

1. Is possible to identify the activity of one invoice emitter by looking at the invoice model?
2. How many different concepts are the different invoices emitted by e.g. wholesaler and a laboratory?
3. What are the differences between the model of the invoices emitted by e.g. wholesaler and a laboratory?
4. What concepts are mandatory for a wholesaler/provider/laboratory?
5. What is the language of this invoice?
6. Is possible to apply any special price to this invoice?
7. What product have we received?
8. What invoicing technologies are using the emitters of this invoice?
9. What is the language of this invoice?

Task 5. Group requirements.

The competency questions have been grouped into four groups. The criteria to group the competency questions have been based in the domain experts' advice and in the set of requirements from [63]. The competency questions groups are all dependent on the domain requirements. These groups have been divided in time dependent requirements, workflow specific domain requirements, specific domain requirements (like What product have we received?), etc.

- CQG1. Competency questions regarding the invoicing workflow
- CQG2. Competency questions regarding multilinguality
- CQG3. Competency questions regarding inference rules
- CQG4. Competency questions related to the receiver of invoices
- CQG5. Competency questions regarding the technology used by the emitters
- CQG6. Competency questions related to the emitter of invoices
- CQG7. Competency questions related to time and date management
- CQG8. Competency questions related to currencies
- CQG9. Composite ones (24 competency questions)

Task 6. Validate the set of requirements.

During the overall process we received recommendations, suggestions and advices from the domain experts and we iterated several times until we got the final approval by the end users.

Task 7. Prioritize requirements.

The ontology requirements were not prioritized. The domain experts and the knowledge engineers who designed the invoice reference ontology considered that the first version of the ontology should cover all the topics reflected by the competency questions.

Task 8. Extract terminology and its frequency.

The terminology used was mainly extracted from the standards we incorporated into the invoice reference ontology not from the competency questions. These standards like UBL contain terminology in order to represent a wide scope of the electronic invoicing domain and we incorporated this vocabulary into the ontology. Vocabulary was also extracted from the PharmaInnova invoice model. Therefore, this task was not considered.

The output of the ontology specification activity carried out with preliminar guidelines is described in [65] and the entire list of the competency questions can be found in its annex.

5.3.3. Semantic Nomenclature Reference Ontology Specification

Among the main objectives of the Semantic Nomenclature case study, we find: helping in the systematization of the creation, maintenance and keeping up-to-date drug-related information, and allowing an easy integration of new drug resources. In order to do that, the case study tackles the engineering of a pharmaceutical product reference ontology based on the use of networked ontologies to solve the particular case scenario of the nomenclature of products in the pharmaceutical sector in Spain. This reference ontology model is a compilation of the main terms and objects related to drugs, the general aspects of them and classify this pharmaceutical terms according to the ATC classification. Also, this reference ontology model is connected with the ontology models of the main databases which contain the information about the pharmaceutical products available in the Spanish market as Digitalis or BOTPlus. In the end, the reference ontology could be linked to the main medical vocabularies used in the world, and it should facilitate the integration of new resources or ontologies that would appear in the evolved scenario.

Next we described the tasks followed for the ontology specification in the Semantic Nomenclature case study based on the guidelines provided by the NeOn methodology in the context of NeOn WP5.

Task 1. Identify purpose, scope and level of formality.

The development of the Reference Ontology and the Nomenclature network ontology is motivated by scenarios presented to the end-user application that will use the ontology network. Such scenarios together with the ontology requirements are described in [64]. The ontology network

should satisfy these requirements after being formally implemented. Then, it should provide a consensual knowledge of the domain, and solve the lack of communication between stakeholders in the pharmaceutical sector. The purpose of the Nomenclature Network Ontology is to provide a complete reference model about all the knowledge around the pharmaceutical products based on the main pharmaceutical classification and models used in the sector.

Task 2. Identify intended users.

The Semantic Nomenclature reference ontology is the elemental component of the case study scenario. The analysis of these scenarios and of the pharmaceutical sector described in [64] allows us to identify the different intended users of the ontology:

User 1: Pharmacist. Pharmacists are the end-users of the ontology and navigate across the ontology searching for drug information.

User 2: GSCoP¹³ technician. GSCoP technicians navigate across the ontology network and search for more information or relations about a given concept (drug, active ingredient, etc.). Also, GSCoP technicians extract the latest information from different sources and update their BOTPlus database

User 3: Spanish Government. Spanish Government analysts study the situation of the pharmaceutical product information in the Spanish market or update the content.

Task 3. Identify intended uses.

The analysis of the motivating scenarios described in [64], allowed us to identify the following main intended uses of the ontology:

Use1. Search updated information about the characteristics of pharmaceutical products

Use2. Connect heterogeneous pharmaceutical models

Use3. Update pharmaceutical product information databases

Task 4. Identify requirements.

According to the previous steps and the scenarios described for the Semantic Nomenclature application in [64], the network of ontologies should satisfy these requirements. By the way, we adopted a bottom up approach for better understanding of the domain and identification of requirements. Competency questions were described in a *Word file* as appears in Figure 15.

We have identified sixty-one competency questions; they are described in detail in [65]. Examples of competency questions are:

CQ1. What is the drug commercial name? *Aspirina C (400/240MG 10 comprimidos Efervescentes)*

CQ2. What is the drug main active ingredient (molecule)? *Acido Acetilsalicilico*

CQ3. What is its Spanish national code? *7127291*

CQ4. What is the drug registration date? *01/09/1976*

¹³ GSCoP: General Spanish Council of Pharmacists

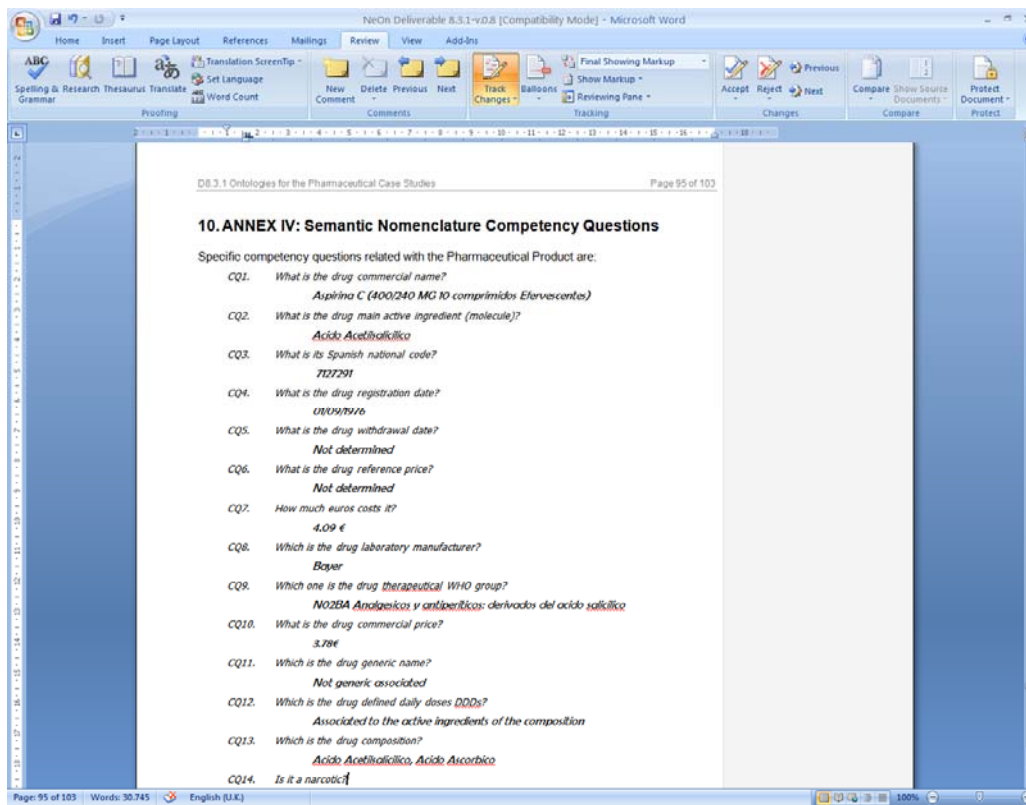


Figure 15. Excerpt of the Semantic Nomenclature Competency Questions

Task 5. Group requirements.

The sixty-one competency questions described in [65] were enumerated and later on grouped into different concept domain:

- CQG1. Pharmaceutical Product (29 competency questions)
- CQG2. Laboratory (4 competency questions)
- CQG3. Active Ingredient (12 competency questions)

The criteria for grouping competency questions are based on the identified uses and users of the ontology and also on domain expert suggestions. Then, competency questions in each group and between groups were composed into more general questions.

- CQG4. Composite ones (16 competency questions)

Moreover, we detected some questions in the previous groups related with date and time terms, and we identified another group for collecting some competency questions related with date and time terms for pharmaceutical products.

- CQG5. Date / Time

Figure 16 shows the final 5 groups and Figure 17 shows examples of CQs in the Laboratory and Pharmaceutical Product groups.

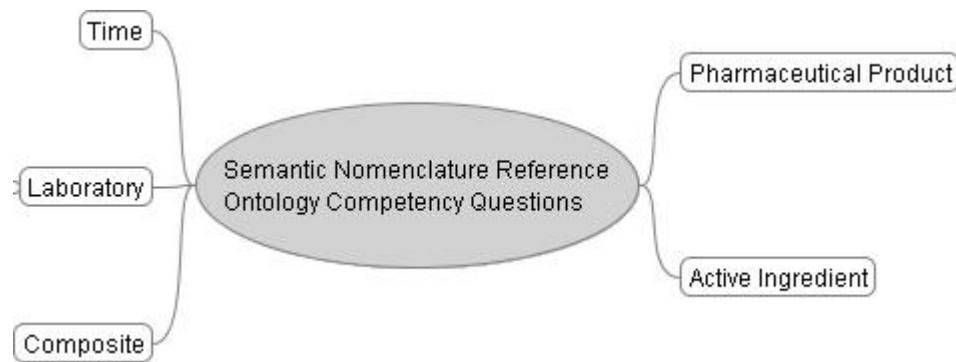


Figure 16. Semantic Nomenclature Competency Questions Groups

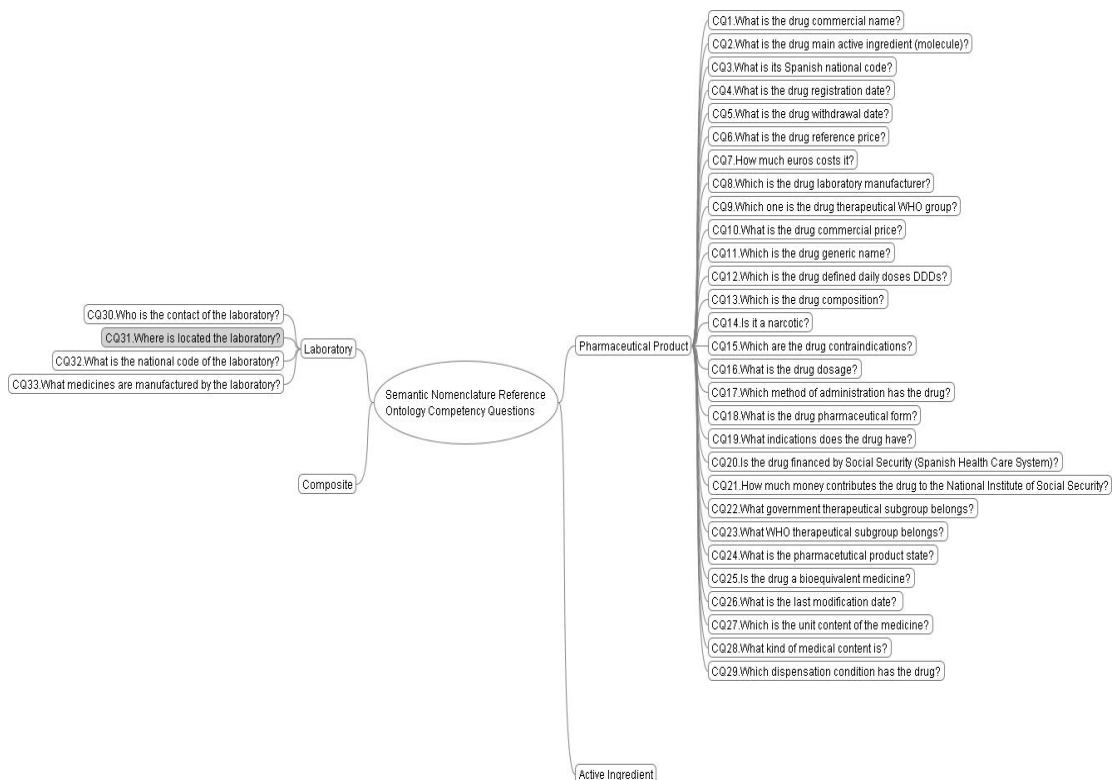


Figure 17. Examples of Competency Questions in Groups

Task 6. Validate the set of requirements.

During the Semantic Nomenclature case study requirements process we received recommendations, suggestions and advices from domain experts. These suggestions were useful when we described the set of competency questions, but during this specification we did not have a particular validation from domain experts. Due to this situation, we iterated, reviewed and refined the list of competency questions.

Task 7. Prioritize requirements.

In the Semantic Nomenclature case study, we did not carried out this step. In our case study we did not prioritize any requirement for the first version of the ontology network specification.

Task 8. Extract terminology and its frequency.

From the competency questions and their answers, we manually extracted the terminology that will be formally represented in the ontology by means of concepts, attributes and relations. We identified the terms and the objects in the universe of discourse.

Some examples of the terms related to the pharmaceutical product are shown in Table 8.

Term	Frequency
Drug	29
• Date (registration, withdrawal)	3
• Price (reference, commercial)	3
• Therapeutical Subgroup	3
• Dosage	1
• Composition	2
• Identification	2
• National Health financing	2
• Route of administration	1
• Units content	1
• Indications	2
• Status	1
• Pharmaceutical form	1

Table 8. Example of Terms and Frequency

The Semantic Nomenclature scenario has a high number of examples of objects which are instances of the main concepts of the ontology. Some examples of *Active Ingredient* instances are shown in Table 9.

Active Ingredients Objects
Ibuprofeno
Butibufeno
Penicilamina
Niflumico Acid
Galamina
Tetrazepam
Procaina
Ketamina
Clotiazepam
Oxitriptan

Table 9. Examples of Objects

The output of the Ontology Specification activity is the Ontology Requirements Specification Document. An excerpt of this document is shown in Table 10.

Semantic Nomenclature Reference Ontology Requirements Specification	
1	Purpose
	The purpose of building the Reference Ontology is to provide a network of ontologies for the pharmaceutical domain. This model is a compilation of the main terms and objects for this particular domain and could be used by health & pharmaceutical entities.
2	Scope
	The ontology has to focus just on the Spanish & European pharmaceutical domain. The level of granularity is directly related to the competency questions and terms identified.
3	Level of Formality

	The ontology has to be implemented in OWL																												
4	Intended Users																												
	<p>User 1: Pharmacist. Pharmacists are the end-users of the ontology and navigate across the ontology searching for drug information.</p> <p>User 2: GSCoP technician. GSCoP technicians navigate across the ontology network and search for more information or relations about a given concept (drug, active ingredient, etc.). Also, GSCoP technicians extract the latest information from different sources and update their BOTPlus database</p> <p>User 3: Spanish Government. Spanish Government analysts study the situation of the pharmaceutical product information in the Spanish market or update the content.</p>																												
5	Intended Uses																												
	<p>Use1. Search updated information about the characteristics of pharmaceutical products</p> <p>Use2. Connect heterogeneous pharmaceutical models</p> <p>Use3. Update pharmaceutical product information databases</p>																												
6	Groups of Competency Questions																												
	<p>CQG1. Pharmaceutical Product (29 competency questions)</p> <p>CQG2. Laboratory (4 competency questions)</p> <p>CQG3. Active Ingredient (12 competency questions).</p> <p>CQG4. Composed ones (16 competency questions).</p> <p>CQG5. Time / Date</p>																												
7	Pre-Glossary of Terms																												
	Terms																												
	<table border="1"> <thead> <tr> <th>Term</th> <th>Frequency</th> </tr> </thead> <tbody> <tr> <td><i>Drug</i></td> <td>29</td> </tr> <tr> <td>• <i>Date (registration, withdrawal)</i></td> <td>3</td> </tr> <tr> <td>• <i>Price (reference, commercial)</i></td> <td>3</td> </tr> <tr> <td>• <i>Therapeutical Subgroup</i></td> <td>3</td> </tr> <tr> <td>• <i>Dosage</i></td> <td>1</td> </tr> <tr> <td>• <i>Composition</i></td> <td>2</td> </tr> <tr> <td>• <i>Identification</i></td> <td>2</td> </tr> <tr> <td>• <i>National Health financing</i></td> <td>2</td> </tr> <tr> <td>• <i>Route of administration</i></td> <td>1</td> </tr> <tr> <td>• <i>Units content</i></td> <td>1</td> </tr> <tr> <td>• <i>Indications</i></td> <td>2</td> </tr> <tr> <td>• <i>Status</i></td> <td>1</td> </tr> <tr> <td>• <i>Pharmaceutical form</i></td> <td>1</td> </tr> </tbody> </table>	Term	Frequency	<i>Drug</i>	29	• <i>Date (registration, withdrawal)</i>	3	• <i>Price (reference, commercial)</i>	3	• <i>Therapeutical Subgroup</i>	3	• <i>Dosage</i>	1	• <i>Composition</i>	2	• <i>Identification</i>	2	• <i>National Health financing</i>	2	• <i>Route of administration</i>	1	• <i>Units content</i>	1	• <i>Indications</i>	2	• <i>Status</i>	1	• <i>Pharmaceutical form</i>	1
Term	Frequency																												
<i>Drug</i>	29																												
• <i>Date (registration, withdrawal)</i>	3																												
• <i>Price (reference, commercial)</i>	3																												
• <i>Therapeutical Subgroup</i>	3																												
• <i>Dosage</i>	1																												
• <i>Composition</i>	2																												
• <i>Identification</i>	2																												
• <i>National Health financing</i>	2																												
• <i>Route of administration</i>	1																												
• <i>Units content</i>	1																												
• <i>Indications</i>	2																												
• <i>Status</i>	1																												
• <i>Pharmaceutical form</i>	1																												
	Objects																												

Active Ingredient Objects
Ibuprofeno
Butibufeno
Penicilamina
Niflumico Acid
Galamina
Tetrazepam
Procaína
Ketamina
Clotiazapam
Oxtripán

Table 10. Excerpt of Semantic Nomenclature Reference Ontology Requirement Specification Document

5.4. Future Work

In this chapter we have presented detailed methodological guidelines for carrying out the ontology specification activity. Further work related with this activity includes: the implementation of a NeOn plug-in supporting this guidelines and the execution and evaluation of the experiments testing the usability and understandability of the guidelines (to be included in D5.6.2).

6. Non Ontological Resource Reuse and Reengineering

In this chapter we present current methods, techniques and tools for reusing and reengineering non ontological resources as well as preliminar NeOn guidelines to help software developers and ontology practitioners to carry out these processes.

6.1. Introduction

As we already mentioned in chapter 4, the NeOn Methodology presents 9 different scenarios for building networks of ontologies. One of these scenarios is *Building Ontology Networks by Reusing and Reengineering Non Ontological Resources*. In this scenario, software developers and ontology practitioners should analyse whether existing non ontological resources that contain already consensuated terminology (available in glossaries, dictionaries, lexicons, classification schemes, thesauri and folksonomies) can be reused to build an ontology network or not. If they decide that one or more non ontological resources are useful for the ontology development, then the *non ontological resource reengineering process* should be carried out, extending non ontological resources with specific domain knowledge, as opposed to custom-building new ontologies from scratch. The underlying principle is that reusing existing and already consensuated terminology allows saving time and money in the ontology development process, and promotes the application of good practices.

During the last years, the ontology engineering research community is being very active in the reuse and reengineering of non ontological resources for speeding up the ontology development process. In other words, software developers and ontology practitioners are realizing the benefits of not starting the ontology development process from scratch. This implies looking first for new methods, techniques, and tools for reusing and reengineering the consensuated terminology contained in available resources. Examples of projects that perform reuse and reengineering are: (1) the NeOn Project, in WP7 case study, Fisheries Ontologies were developed for use within the Fish Stock Depletion Assessment System (FSDAS) [28], by reusing resources available for the fisheries domain, e.g. the FIGIS database, (2) the SEEMP¹⁴ project in which a Reference Ontology has been built by reusing human resources management standards, and (3) SKOS¹⁵, the most used metamodel to reengineering thesauri, which is being used by several organizations to export their existing vocabularies in order to support their reuse by other communities, i.e. they are converting the vocabularies from a proprietary format into a consensuated format.

This chapter is organized as follows: section 6.2 presents the state of the art on methods, techniques and tools for reusing and reengineering non ontological resources Section 6.3 depicts the proposed typology of non ontological resources. Section 6.4 shows a high level overview of the NeOn methodology for reuse and reengineering non ontological resources, and section 6.5 and section 6.6 describe the details of the methodological guidelines proposed for reusing non ontological resources and for the reengineering of resources. Finally, section 6.7 presents the conclusions and future work.

¹⁴ <http://www.seemp.org>

¹⁵ <http://www.w3.org/2004/02/skos/>

6.2. State of the Art

A review of the state of the art regarding reuse and reengineering of non ontological resources shows that most of the analyzed research only focuses on the transformation process of these resources into ontologies, for instance, transformation of standards [85, 74], thesauri and lexicons [75, 98, 115, 121, 122], XML files [56], hierarchical classifications [58, 75], folksonomies [98], relational databases [17, 109], and spreadsheets [72] among others. These works only concentrate on the reengineering process of the type of non ontological resources (classifications, thesauri, lexicons, etc.) and of the implementation of non ontological resource (XML, spreadsheets, etc.).

In NeOn deliverable 2.2.1, Sabou et al. [98] distinguish two approaches for the resource transformation. The first one consists in transforming resource schema into an ontology schema, and then resource content into instances of the ontology. The second one transforms resource content into an ontology schema. For each one of the contributions presented below, we will comment which of the two approaches is followed.

6.2.1. Methods

In this section we present some of the methods we found in the literature related with the reuse and reengineering of non ontological resources. First, we introduce some research to transform classification schemes, folksonomies, lexica and thesauri into ontologies. Then, we show some works which deal with non ontological resource data sources as databases and XML files.

□ *Methods for transforming classification schemes into ontologies.*

The two main approaches for transforming classification schemes into ontologies are presented in [75, 74, 85]. Next we describe each one of them.

A method for deriving ontologies from hierarchical classifications is presented in [75, 74], in which a semi-automatic creation of the ontology is proposed. Both deal with classifications schemes such as the standard categorization for products and services, UNSPSC¹⁶. This classification scheme is in continuous evolution and holds thousand of categories, which makes it impractical to create and update the ontology manually. The method consists of the following steps:

- To pre-process and create a formal representation of the classification scheme.
- To derive classes from each category and set the relation among them according to a given context.
- To derive a class from each category and set a taxonomic relation among them.
- To generate the ontology in an ontology language (e.g. OWL).

This method follows the approach used for transforming resource content into an ontology schema. However, in [75, 74] no information is given about how to find the most suitable non ontological resource for the transformation.

The creation of a Human Resource Ontology reusing some existing widespread standards and classifications is presented in [85]. The method consists of the following steps:

- To discover potentially useful resources, by searching in general purpose search engines, domain-related web sites, and organizations. However, no detailed information is given about how to carry out the search.
- To select manually the resources without the usage of a pre-defined methodology.
- To extract and integrate the relevant fragments of the selected resources to the ontology.

¹⁶ <http://www.unspsc.org/>

This work follows the approach of transforming resource content into an ontology schema, and the resultant ontology is expressed in OWL.

❑ *Methods for transforming folksonomies into ontologies.*

The main approach for transforming folksonomies into ontologies is presented in NeOn deliverable D2.2.1 [98]. This approach proposes an algorithm for the semantic enrichment of folksonomies. It consists of: a) defining concepts for each tag (linking tags to ontology concepts) and b) discovering relations between all the possible pairs of tags.

This approach follows the one for transforming resource content into an ontology schema, and does not provide information about how to select the resource.

❑ *Methods for transforming lexica into ontologies.*

The two main approaches for transforming lexica into ontologies are presented in NeOn deliverable D2.2.1 [98]. They are focused on WordNet¹⁷.

The first approach consists of the following steps:

- To create a set of classes for each one of the main components of WordNet including classes for word, synset and sense, among others.
- To model all words, synsets and senses belonging to WordNet, as instances of the previously created classes.
- To code part of the semantics related to each instance by means of the URIs used to identify each instance.

This first approach is based on the one for transforming resource schema into an ontology schema, and then resource content into instances of the ontology. The resultant ontology is expressed in OWL.

The second approach aims at producing a formal specification of WordNet by means of an ontology. The reengineering process creates a class for each synset. It is also based on a set of assumptions on how to discover and map relations between WordNet components and ontology relations. The proposed steps are more focus on the learning process for discovering relations between words and synsets from the words textual definition (sense), than in a reengineering method. This second approach follows the one based on transforming resource content into an ontology schema. The resultant ontology is expressed in OWL as well. However, in [98] no information is given about how to find the most suitable non ontological resource for the transformation.

❑ *Methods for transforming thesauri into ontologies.*

The three main approaches for transforming thesauri into ontologies are presented in [115, 121, 122]. Next, we describe them.

The work described in [115, 122] provides a method for converting existing thesauri into ontologies with a set of guidelines. The method consists of the following steps:

- To find the most suitable thesauri for the conversion. However, no detailed information is given about how to carry out this step.
- To gather information about the thesaurus data source and conceptual model.
- To transform the source representation into an ontology language, trying to preserve the original structure.
- To augment class and properties with additional constraints as transitive or symmetric properties.

¹⁷ <http://wordnet.princeton.edu/>

This method follows the approach for transforming resource schema into an ontology schema, and then resource content into ontology instances. The resultant ontology is expressed in OWL or RDF(S).

The process for transforming the Art and Architecture Thesaurus (AAT) into an RDF(S) ontology is depicted in [121]. The method consists of the following steps:

- To convert the full AAT hierarchy into a hierarchy of concepts.
- To augment a number of concepts with additional slots and fillers.
- To add knowledge about the relation between possible values of fields and nodes in the knowledge base.

This method follows the approach centered on transforming the resource content into an ontology schema, and does not provide information about how to select the resource.

□ *Methods for transforming databases into ontologies.*

The two main approaches for transforming databases into ontologies are presented in [109, 17]. Next we outline them.

The first approach described in [109] is based in the semi-automatic generation of the ontology schema from the database relational model. This method consists of the following steps:

- To apply reverse engineering techniques, supervised by the ontology practitioner, to the database schema.
- To transform database records into ontology instances, having in this case the database content replicated in the ontology.

This first approach follows the one for transforming the resource schema into the ontology schema, and then resource content into instances of the ontology. This approach does not provide information about how to select the resource

The second approach [17] consists on creating mappings between legacy databases and existing ontologies. In that approach the ontology is not a mirror of the database schema. The framework proposed is composed of the R₂O mapping language and the ODEMapster processor. This second approach consists of the following steps:

- To discover semi-automatically mappings between the database and ontology elements.
- To express those mappings in a formal language, R₂O in this case.
- To evaluate and verify those mappings.
- To exploit those mappings for retrieving the data using ODEMapster.

In this approach the database content is not transferred into ontology instances, but accessed by exploiting those mappings for retrieving database data. Since this method starts from an existing ontology schema, it does not follow the approach of transforming the resource schema into the ontology schema, but the database content is transformed on demand into ontology instances.

□ *Methods for transforming XML files into ontologies.*

The main approach for transforming XML files into ontologies is described in [56]. This method proposes to create the ontology from the XML schema and then populate it with instances created from the XML data. The XSD2OWL (XML Schema Definition to OWL) mapping tool transforms the XML schema constructors to OWL ones according to a set of rules. The XML2RDF tool follows a structure-mapping model, where XML data instances are translated to RDF instances that instantiate the corresponding OWL construct. This method assumes that the most suitable XML resource is already found. This approach follows the one for transforming the resource schema into the ontology schema, and then resource content into instances of the ontology. However, in

[56] no information is given about how to find the most suitable non ontological resource for the transformation.

6.2.2 Techniques

We have not found any technique to carry out the non ontological resource reuse and reengineering processes except the use of reengineering patterns that has been mentioned in D2.5.1 [94]. Reengineering ontology design patterns are defined as transformation rules applied in order to create a new ontology (target model) from elements of a source model. The target model is an ontology, while the source model can be either an ontology or a non-ontological resource, e.g., a thesaurus concept, a data model pattern, an UML model, a linguistic structure, etc. D2.5.1 distinguishes between two types of reengineering patterns:

- Schema reengineering patterns which provide designers with rules for transforming a non-OWL DL metamodel
- Refactoring patterns which provide designers with rules for transforming an OWL DL source ontology into a new OWL DL target ontology.

This work presents a unique example of a schema reengineering pattern, which includes four rules to transform a knowledge organization system into SKOS¹⁸. These rules just identify the elements of the source model that are mapped to their corresponding elements of the target model, but the rules do not provide information about how to carry out the mapping. Reengineering patterns are not integrated within a method to carry out the reengineering process. Moreover, it is not proposed a template to describe reengineering patterns.

Some techniques which are part of the methods presented in section 6.1.1 are more related to the ontology learning process than to the reengineering process, as it is the case of the ones mentioned in D2.2.1 [98] to transform the WordNet lexicon into an ontology like learning association links and learning conceptual relations.

6.2.3 Tools

In this section we present tools which support the transformation process to create ontologies from non ontological resources. This survey is not complete, but we present the most representative tools for each group. After analyzing the state of the art, we realized that some of these tools depend on the type of non ontological resource, and others depend on the resource implementation (databases, spreadsheets, XML files). None of these tools help software developers and ontology practitioners to find the best non ontological resources for the transformation.

❑ *Tools for transforming classification schemes into ontologies.*

SKOS2GenTax¹⁹ is an online tool that converts hierarchical classifications available in the W3C SKOS²⁰ format into RDF(S) or OWL ontologies. SKOS2GenTax uses the GenTax algorithm described in [75]; this tool follows the approach based on transforming resource content into an ontology schema.

❑ *Tools for transforming XML files into ontologies.*

XSD2OWL and XML2RDF are tools which support the XML semantic reuse method [56], which was described in section 6.2.1. These tools support the approach for transforming resource

¹⁸ <http://www.w3.org/2004/02/skos/>

¹⁹ <http://www.heppnetz.de/projects/skos2gentax/>

²⁰ <http://www.w3.org/2004/02/skos/>

schema into an ontology schema and the resource content is transformed into instances of the ontology.

□ *Tools for transforming spreadsheet files into ontologies.*

RDF123 [72] is a tool for transforming spreadsheet files into RDF. It allows end users to develop mappings between spreadsheet data and RDF. In this work it is stated that existing spreadsheets to RDF tools typically map a spreadsheet row to an instance, with each column representing a property. RDF123 allows users to map each row in the spreadsheet to a different RDF schema. This tool is intended to create instances of existing ontologies, and hence, we consider it more as a population tool than as a reengineering tool.

□ *Tools for transforming databases into ontologies.*

KAON REVERSE is a tool which supports the reverse engineering approach for transforming databases into ontologies [109] presented in section 6.2.1. This tool helps in the process of transforming the database schema into the ontology schema and also on data migration. Therefore, we can say that it supports the approach for transforming resource schema into an ontology schema and resource content into instances of the ontology.

The mapping language R₂O and its processor ODEMapster [17] also introduced in section 6.2.1 constitute a tool for transforming database content into ontology instances. This tool is intended to create instances of an existing ontology, on demand or in batch processing, and for this reason, we consider it more as a population tool than as a reengineering tool.

6.2.4 Conclusion

After having analyzed the state of the art on non ontological resources reuse and reengineering, we can conclude that there are no detailed guidelines on how to find the most suitable non ontological resources for the development of ontologies. Most of the presented research assumes that we already have suitable resources for conversion, and only focuses on the reengineering process. In conclusion, we can state that there is a clear need for some sort of methods, techniques and tools for non ontological resource reuse.

Regarding reengineering of non ontological resources to build ontologies, we conclude that research efforts have been mainly divided into the data source and the type of non ontological resource. It has also been analyzed how to map non ontological resources content and schema into ontology instances and schema, but none of the research works have taken advantage from the data model which underlies the non ontological resource to guide the reengineering process. Finally, it is left to say that none of the analyzed reengineering approaches propose a set of reengineering patterns to guide the reengineering process and that there is also a lack of reengineering methods to support some of reengineering process activities by using reengineering patterns.

6.3. Type of Non Ontological Resources

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

There is a big amount of non ontological resources that embody knowledge about some particular domains, and which represent some grade of consensus for a user community. These resources present the form of free texts, textual corpora, web pages, standards, catalogues, web directories, classifications [7], thesauri [115], lexicons [11] and folksonomies [116], among others. Non ontological resources have related semantics which allow interpreting the knowledge they hold. Some times this semantic is explicitly specified on documents in natural language; in other cases, however, the semantic is not explicitly available, but it can be extracted from additional sources of information as the user community that use the resource. Regardless of whether the semantic is

explicit or not, the main problem is that the semantic of non ontological resources is not always formalized, and this lack of formalization avoids the use of them as ontologies.

The analysis of the literature has revealed that there are different ways of categorizing non ontological resources [80, 98, 53, 76]. Maedche et al. [80] and Sabou et al. [98] classify non ontological resources into unstructured (e.g. free text), semi-structured (e.g. folksonomies) and structured (e.g. databases) resources. Gangemi et al. [53] distinguish catalogues of normalized terms, glossed catalogues, and taxonomies. Hodge [76] proposes characteristics such as structure, complexity, relationships among terms, and historical functions for classifying them. Currently, an accepted and consensuated typology of non ontological resources does not exist.

In this deliverable we propose a new categorization of non ontological resources according to three different features.

- The first one refers to the *type of non ontological resource*. It refers to the type of knowledge encoded by the resource.
- The second one refers to the designed *data model*, that is, the designed data model used to represent the knowledge encoded by the resource.
- The third feature refers to the resource *implementation*.

This classification is an ongoing work; we will extend it in the deliverable *D2.2.2b. Methods and tools for reengineering* due to M30. Below we explain in more detail the proposed classification.

1 According to the **type of non ontological resource** we classify them into:

- *Glossaries*: A glossary is a terminological dictionary that contains designations and definitions from one or more specific subject fields. The vocabulary may be monolingual, bilingual or multilingual [8]. As an example we mention the FAO Fisheries Glossary²¹.
- *Dictionaries*: A dictionary is a structured collection of lexical units with linguistic information about each of them [9]. As an example we mention the data dictionary of the EDI standard²² used within the NeOn Pharmaceutical case studies.
- *Lexicons*: A lexicon is the vocabulary of an individual person, an occupational group or a professional field [11]. As an example we mention the Specialist Lexicon²³.
- *Classification schemes*. A classification scheme is the descriptive information for an arrangement or division of objects into groups based on characteristics the objects have in common [7]. For example, the Fishery International Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP)²⁴ used within the NeOn FAO case studies.
- *Thesauri*: A thesaurus is a controlled vocabulary arranged in a known order whose purpose is to facilitate retrieval of resources and to achieve consistency in indexing [10]. There are standards for the development of monolingual thesauri (NISO 1998; ISO 1986²⁵) and multilingual thesauri (ISO 1985²⁶). As an example we mention the AGROVOC²⁷ thesaurus used within the NeOn FAO case studies.

²¹ <http://www.fao.org/fi/glossary/default.asp>

²² <http://www.edigateway.com/glossary.html>

²³ <http://lexsrv3.nlm.nih.gov/SPECIALIST/index.html>

²⁴ <http://www.fao.org/figis/servlet/RefServlet>

²⁵ http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=27641

²⁶ http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=12159

²⁷ <http://www.fao.org/agrovoc/>

- *Folksonomies*: A folksonomy is the result of personal free tagging of information and objects (anything with an URI) for one’s own retrieval [116].
- 2 There are different ways of representing the knowledge encoded by the resource. In the following we present several **data model** for classification schemes, that are shown in Figure 18.

Path Enumeration	Category Name	Category Description
1	Category1	Category1Desc
11	Category11	Category11Desc
111	Category111	Category111Desc
12	Category12	Category12Desc
121	Category121	Category121Desc
2	Category2	Category2Desc
...

Category Code	Category Name	Parent Category Code
1	Category1	Null
2	Category2	Null
3	Category3	1
4	Category4	1
5	Category6	3
6	Category7	4
...

a) Path Enumeration

b) Adjacency List

First level categories entity		
Category Code	Category Name	Category Description
1	Category1Level1	Category1Level1Desc
2	Category2Level1	Category2Level1Desc
..

Second level categories entity			
Category Code	First Level Category	Category Name	Category Description
1	1	Category1Level2	Category1Level2Desc
2	1	Category2Level2	Category2Level2Desc
..

Third level categories entity			
Category Code	Second Level Category	Category Name	Category Description
1	1	Category1Level3	Category1Level3Desc
2	2	Category2Level3	Category2Level3Desc
..

c) Snowflake

Flattened entity						
First level		Second level		Third level		...
Category Code	Category Name	Category Code	Category Name	Category Code	Category Name	...
1	Category1Level1	1	Category1Level2	1	Category1Level3	...
1	Category1Level1	2	Category2Level2	2	Category2Level3	...
2	Category2Level1
..

d) Flattened

Figure 18. Classification Schemes Data Models

- *Path Enumeration* [23] is a recursive structure for hierarchy representations defined as a model which stores for each node the path (as a string) from the root to the node. This string is the concatenation of the nodes code in the path from the root to the node. Path

enumeration is used to publish some European classifications by Eurostat²⁸. Figure 18-a) shows this model.

- *Adjacency List* [23] is a recursive structure for hierarchy representations comprising a list of nodes with a linking column to their parent nodes. Figure 18-b) shows this model.
- *Snowflake* [81] is a normalized structure for hierarchy representations. For each hierarchy level a table is created. In this model each hierarchy node has a linked column to its parent node. This representation is similar to the Adjacency List because of the linking column to the parent nodes. However, the difference is that in snowflake models each hierarchy level is stored on a different table, and therefore hierarchy levels must be known in advance. Figure 18-c) shows this model.
- *Flattened* [81] is a denormalized structure for hierarchy representations. The hierarchy is represented using one table where each hierarchy level is stored on a different column. This model is similar to path enumeration because each row has the complete path from the root to the node, but in the path enumeration this information is in one column while in the flattened is stored on several columns, one for each hierarchy level. In the same way as the snowflake model, in this case the hierarchy levels must be known in advance to create the respective columns. Figure 18-d) shows this model.

3. According to the **implementation** we classify non ontological resources into:

- *Databases* [6]: A collection of logically related data stored together in one or more files
- *XML*²⁹: eXtensible Markup Language is a simple, open, and flexible format used to exchange a wide variety of data on and off the Web. XML is a tree structure of nodes and nested nodes of information, in which the user defines the names of the nodes.
- *Flat files*: A flat file is a file that is usually read or written sequentially and does not have indexes that can be individuated from the individual records. In general, a flat file is a file containing records that have no structured inter-relationships.
- *Spreadsheets*: An electronic spreadsheet consists of an array of cells into which a user can enter formulas and values.

Figure 19 shows how a classification scheme can be modeled following one of the four data models presented in the middle layer. The classification scheme is modeled following a path enumeration model, which could be implemented using one of the four implementations depicted in the bottom layer. In this example we observe the same resource implemented in a database and in an XML file.

²⁸ <http://ec.europa.eu/eurostat/ramon>

²⁹ Extensible Markup Language (XML). <http://www.w3.org/XML/>

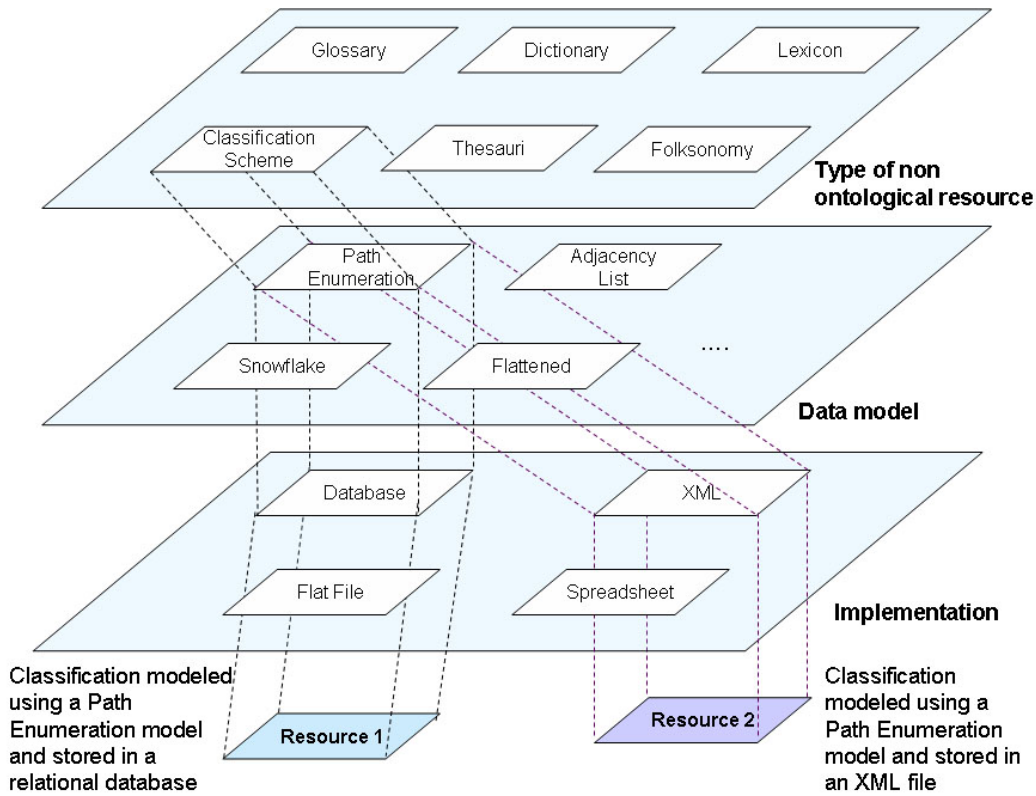


Figure 19. Non Ontological Resources Categorization

To exemplify the non ontological resource categorization presented with a real life classification resource, we use the FAO classification of water areas³⁰ published in the fisheries global information system web page. An excerpt of the water area classification is presented in Figure 20.

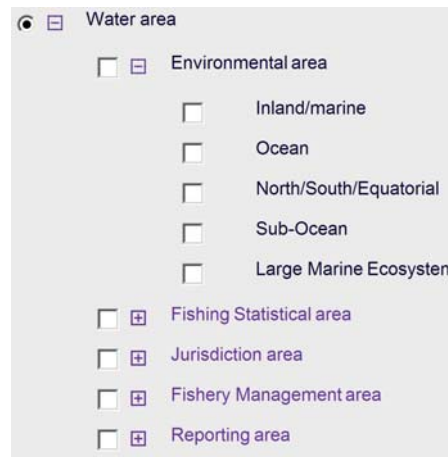


Figure 20. Water Area Classification

This classification resource could be modeled following one of the data models we have presented previously. The water area classification is modeled following an adjacency list model as is

³⁰ <http://www.fao.org/figis/servlet/RefServlet>

depicted in Figure 21-a). An alternative way of modeling the water area classification using a path enumeration model is shown in Figure 21-b).

Id	Category Name	Parent
20000	Water area	1
21000	Environmental area	20000
22000	Fishing Statistical area	20000
24020	Jurisdiction area	20000
21001	Inland/marine	21000
21002	Ocean	21000
21003	North/South/Equatorial	21000
21004	Sub Ocean	21000
21005	Large Marine ecosystem	21000

a) Adjacency List

Id	Category Name
20000	Water area
20000.21000	Environmental area
20000.22000	Fishing Statistical area
20000.24020	Jurisdiction area
21000.21001	Inland/marine
21000.21002	Ocean
21000.21003	North/South/Equatorial
21000.21004	Sub Ocean
21000.21005	Large Marine ecosystem

b) Path Enumeration

Figure 21. Water Area Classification Data Models

Finally these data models can be implemented in any data source format. A directly implementation would be as tables in a relational database or in a spreadsheet. Figure 22 presents an XML implementation of the adjacency list model of the water area classification, Figure 23 presents a spreadsheet implementation of the adjacency list model, and Figure 24 presents an XML implementation of the path enumeration model of the same classification scheme.

```

<Classification>
  <Category>
    <NodeId>20000</NodeId>
    <WaterCategory>Water Area</WaterCategory>
    <parentNodeId>1</parentNodeId>
  </Category>
  <Category>
    <NodeId>21000</NodeId>
    <WaterCategory>Environmental area</WaterCategory>
    <parentNodeId>20000</parentNodeId>
  </Category>
  <Category>
    <NodeId>22000</NodeId>
    <WaterCategory>Fishing statistical area</WaterCategory>
    <parentNodeId>20000</parentNodeId>
  </Category>
  <Category>
    <NodeId>24020</NodeId>
    <WaterCategory>Jurisdiction area</WaterCategory>
    <parentNodeId>20000</parentNodeId>
  </Category>
  <Category>
    <NodeId>21001</NodeId>
    <WaterCategory>inland/marine</WaterCategory>
    <parentNodeId>21000</parentNodeId>
  </Category>
  ...
</Classification>

```

Figure 22. Water Area Classification XML Implementation for the Adjacency List Model

	A	B	C
1	Id	Category Name	Parent
2	20000	Water area	1
3	21000	Environmental area	20000
4	22000	Fishing Statistical area	20000
5	24020	Jurisdiction area	20000
6	21001	lland/marine	21000
7	21002	Ocean	21000
8	21003	North/South/Equatorial	21000
9	21004	Sub Ocean	21000
10	21005	Large Marine ecosystem	21000

Figure 23. Water Area Classification Spreadsheet Implementation for the Adjacency List Model

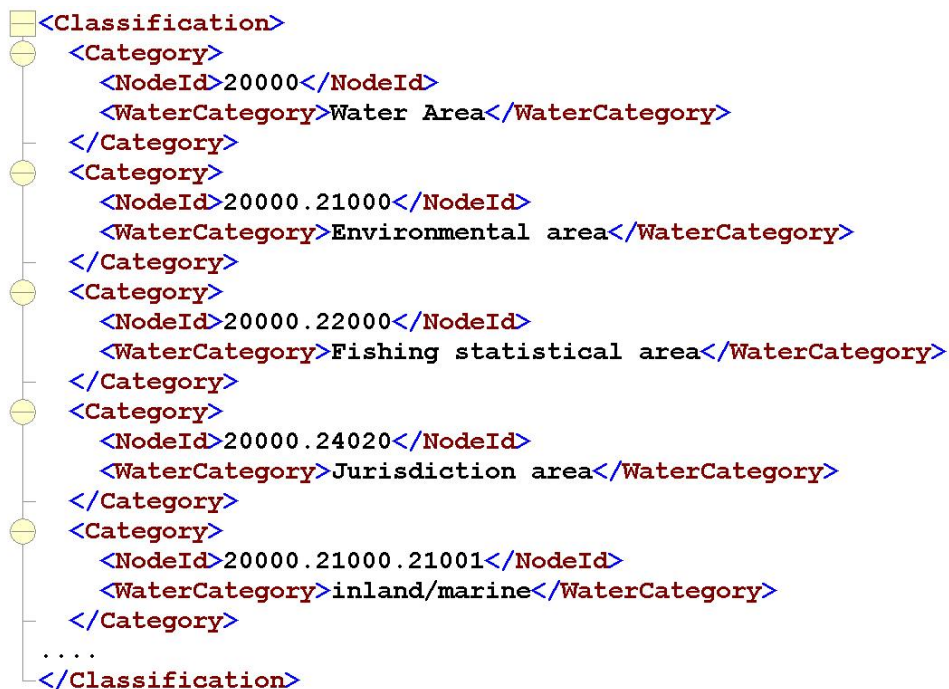


Figure 24. Water Area Classification XML Implementation for the Path Enumeration Model

6.4. The NeOn Approach for Non Ontological Resource Reuse and Reengineering

Non ontological resource reuse and reengineering processes belong to the development scenario called *Building Ontology Networks by Reusing and Reengineering Non Ontological Resources*, in which it is supposed that software developers and ontology practitioners want to develop the ontology network by means of reusing existing non ontological resources.

The NeOn approach to carry out non ontological resource reuse and reengineering processes is depicted in Figure 25. Software developers and ontology practitioners should accomplish first the *non ontological resource reuse process* with the goal of analysing whether existing non ontological resources can be reused to build the ontology network. If they decide that one or more resources are useful for the development, then the *non ontological resource reengineering process* should be carried out to transform the selected non ontological resources into ontologies.

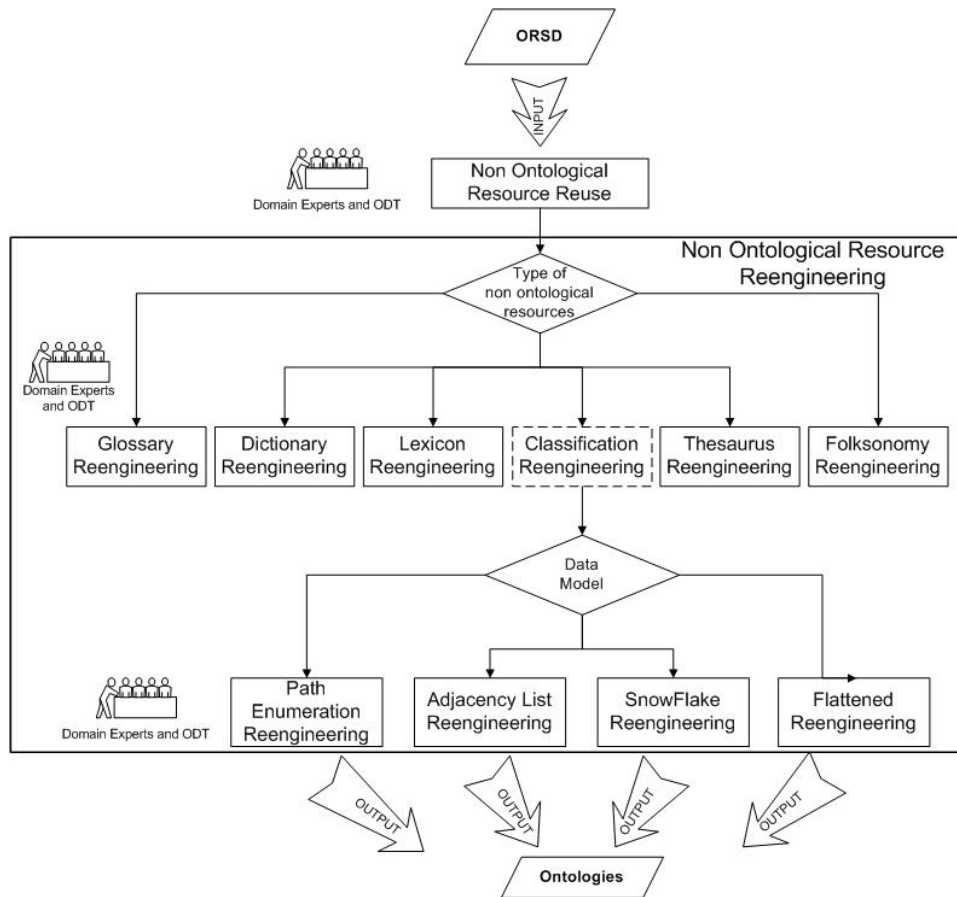


Figure 25. Non Ontological Reuse and Reengineering Approach

The proposed guidelines for non ontological reuse are explained in section 6.5, and the proposed approach for non ontological resource reengineering in section 6.6. It is worth to mention that the non ontological resource reengineering process proposed in this deliverable is inspired in how reengineering is performed in software engineering. In such field, software reengineering [29] is defined as the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. A general software reengineering model presented in [27] is depicted in Figure 26. The software reengineering model uses four different abstraction levels to define each activity: 1) the conceptual level which describes in general terms the functional characteristics of the system; 2) the requirements level which is the specification of the problem being solved; 3) the design level which is the specification of the solution; and 4) the implementation level which refers to the coding, testing and delivery of the operational system.

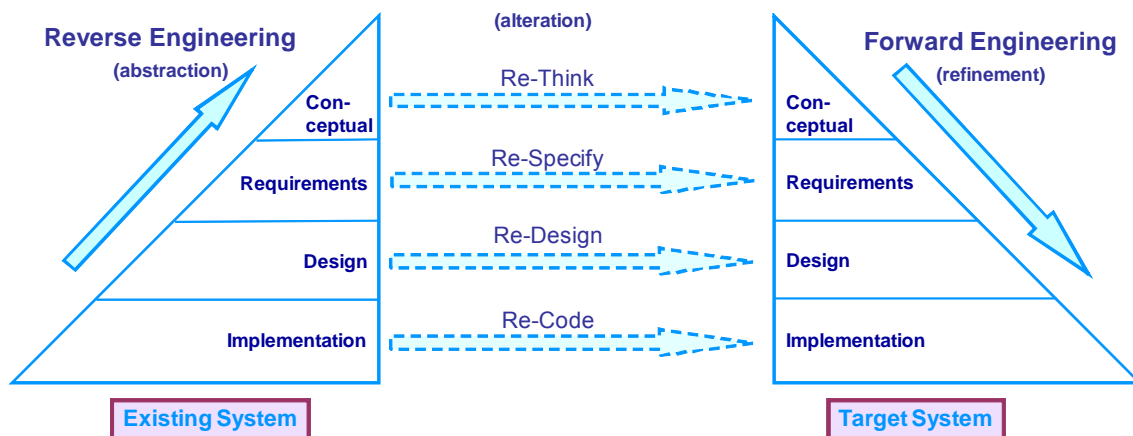


Figure 26. General Model for Software Reengineering [27]

The model presented in Figure 26 depicts the software reengineering main activities that are the explained below:

- ❑ Reverse engineering [29] is the process of analyzing a subject system to identify the system components and their interrelationships, and create representations of the system in another form or at a higher level of abstraction. Two well known activities of reverse engineering are redocumentation and design recovery. Redocumentation aims at recovering documentation about the subject system that existed or should have existed. Design recovery aims at reproducing all the information required for a person to fully understand what a program does and how it does.
- ❑ Alteration, also called restructuring [29], is the transformation from one representation form to another at the same relative abstraction level, while preserving the subject system's external behaviour. Possible transformations identified in [27] are: changes to implementation characteristics (re-code), changes to design characteristics (re-design), changes to requirements characteristics (re-specify) and changes to conceptual characteristics (re-think).
- ❑ Forward engineering [29] is the traditional process of moving from high level abstractions and logical, implementation-independent designs to the physical implementation of a system. Forward engineering and software development are synonymous.

Furthermore, the non ontological reengineering process within NeOn will depend on the type of non ontological resource to be reengineered. The NeOn methodology will define different reengineering processes for glossaries, dictionaries, lexicons, classification schemes, thesauri, and folksonomies. The non ontological resource reengineering process is also strongly influenced by the non ontological resource data model, since it defines how non ontological resource components are modelled.

6.5. NeOn Proposed Guidelines for Non Ontological Resource Reuse

The goal of the Non Ontological Resource Reuse process is to choose the most suitable non ontological resource to be used for building ontologies. Domain experts, software developers and ontology practitioners carry out this process taking as input the ontology requirements specification document (ORSD) 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. Table 11 shows the non ontological resource process filling card, which includes the definition, goal, input, output, who carries out the process and when the process should be carried out.

Non Ontological Resource Reuse	
<i>Definition</i>	
<div style="border: 1px solid black; padding: 5px;"> <p><i>Non Ontological Resource Reuse</i> refers to the process of choosing the most suitable non ontological resources for the development of ontologies³¹.</p> </div>	
<i>Goal</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>To choose the most suitable non ontological resources for building ontologies.</p> </div>	
<i>Input</i>	<i>Output</i>
<div style="border: 1px solid black; padding: 5px;"> <p>The ontology requirements specification document (ORSD).</p> </div>	<div style="border: 1px solid black; padding: 5px;"> <p>A set of non ontological resources that to some extent covers the expected domain.</p> </div>
<i>Who</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>Domain experts, software developers and ontology practitioners.</p> </div>	
<i>When</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>After the ontology specification activity and before the non ontological resource reengineering process.</p> </div>	

Table 11. Non Ontological Resource Reuse Filling Card

This process includes the activities and tasks presented in Figure 27 and explained in the following.

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 ontology requirements specification document (ORSD). They use those terms that have a highest frequency in the ORSD to search for candidate non ontological resources that cover the desired terminology. The activity output is a set of candidate non ontological resources that might present any of the identified typologies described in section 6.3.

³¹ In this document we slightly modify the definition of *Non Ontological Resource Reuse* from the NeOn Glossary of Activities [111].

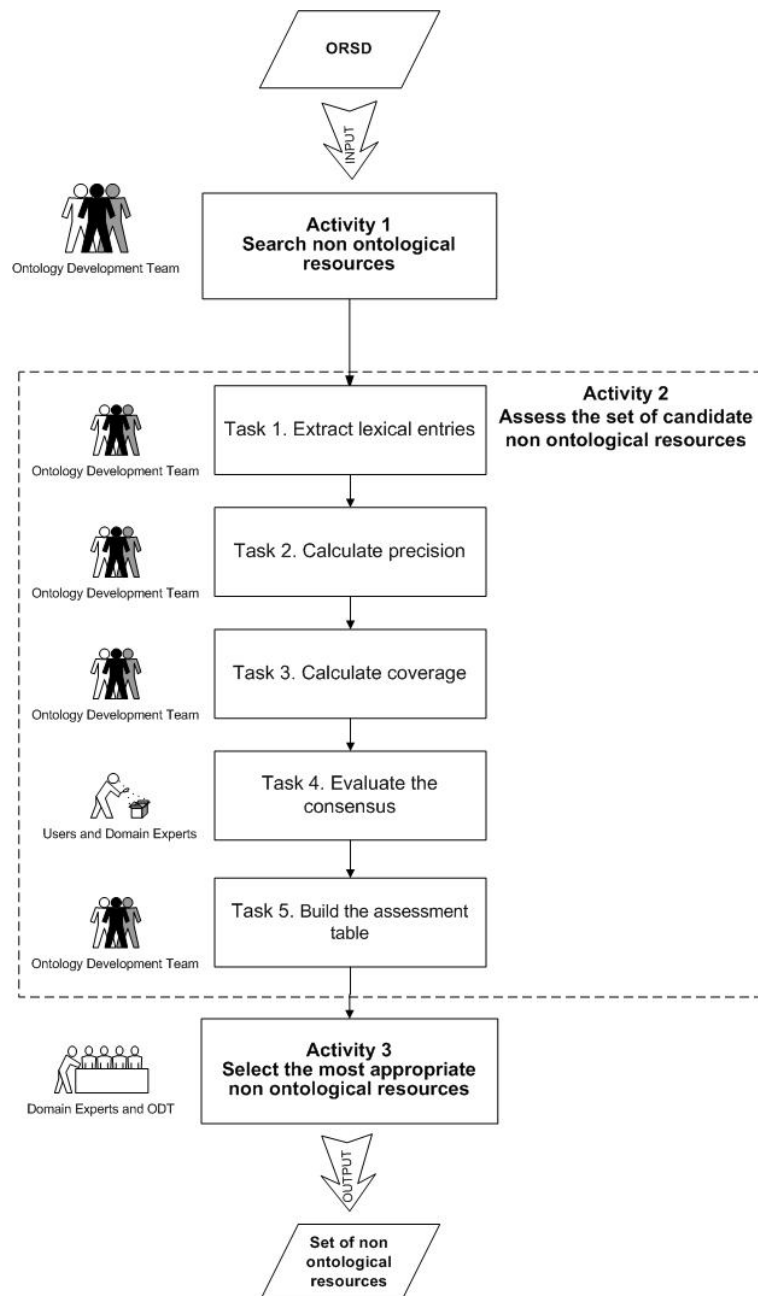


Figure 27. Proposed Activities in NeOn for the Non Ontological Resource Reuse Process

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 take into account the following criteria: *coverage* and *precision*, which are measurable criteria; and *consensus*, which is a subjective criterion. The activity output is an assessment table that shows the evaluation criteria for every non ontological resource.

Since there are no methodological guidelines for assessing non ontological resources, we take as starting point some of the criteria proposed in [54] for evaluating the suitability of an ontology for a particular purpose; the criteria are: *coverage*, *precision*, and *agreement*. Here we adapted such criteria with the goal of assessing the non ontological resources found.

Task 1. Extract lexical entries.

The goal of this task is to extract the lexical entries of the non ontological resources. Software developers and ontology practitioners carry out this task taking as input the non ontological resources for extracting their lexical entries using terminology extraction tools.

Task 2. Calculate precision.

The goal of this task is to calculate the precision of the non ontological resources. Software developers and ontology practitioners carry out this task, taking as input the lexical entries extracted for the non ontological resources and the Ontology Requirements Specification Document (ORSDD) for compute the precision of the non ontological resources. Precision is a measure widely used in information retrieval [16]. It is defined as the proportion of retrieved material that is actually relevant. To adapt this measure into our context we need to define:

$\{NORLexicalEntries\}$ is the set of lexical entries extracted from the non ontological resource.

$\{ORSDDTerminology\}$ is the set of identified terms included in the Ontology Requirements Specification Document (ORSDD).

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:

$$Precision = \frac{\{NORLexicalEntries\} \cap \{ORSDDTerminology\}}{\{NORLexicalEntries\}}$$

Task 3. Calculate coverage.

The goal of this task is to calculate the coverage of the non ontological resources. Software developers and ontology practitioners carry out this task, taking as input the lexical entries extracted from the non ontological resources and the Ontology Requirements Specification Document (ORSDD) for computing the coverage of the non ontological resources. Coverage is based on *recall* measure used information retrieval [16]. Recall is defined as the proportion of relevant material actually retrieved in answer to a search request. To adapt this measure into our context, we use the aforementioned definition of $\{NORLexicalEntries\}$ and $\{ORSDDTerminology\}$. 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:

$$Coverage = \frac{\{NORLexicalEntries\} \cap \{ORSDDTerminology\}}{\{ORSDDTerminology\}}$$

Task 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. Domain experts carry out this task taking as input the non ontological resources for stating whether the non ontological resources contain already consensuated terminology by the community or not.

Task 5. 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 and consensus criteria, for the construction of the assessment table. This table is shown in Table 12. In the first column we include the non ontological resources found. In the precision column we include the calculated precision value for each non ontological resource. In the coverage

column we include the calculated coverage value for each non ontological resource. Finally, in the consensus column we include the domain experts' judgment whether the non ontological resource has consensus by the community or not (Yes/No).

	Precision	Coverage	Consensus
NOR1	NOR1 Precision Value	NOR1 Coverage Value	(Yes/No)
NOR2	NOR2 Precision Value	NOR2 Coverage Value	(Yes/No)
NOR3	NOR3 Precision Value	NOR3 Coverage Value	(Yes/No)

Table 12. Assessment Table

Activity 3. Select the most appropriate non ontological resources.

The goal of this activity is to select the most appropriate non ontological resources. Domain experts, software developers and ontology practitioners carry out this activity taking as input the non ontological resource assessment table. The selection is performed manually and looking for resources with:

- Consensus. This criterion is taken into account in the first place, since if the resource to be reused contains terminology already consensuated by the community, the effort and time spent in finding out the right labels for the ontology terms will decrease considerably.
- High value of coverage. This criterion is taken into account in the second place, because our second concern is to consider most of the identified terms of the ORSD.
- High value of precision. This criterion is taken into account in the third place, because our third 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.

6.5.4. Example

To describe the proposed guidelines in a more practical way, we present an example from the SEEMP Project³². As we mentioned in section 5.4.1, the EU SEEMP project aims at improving workers mobility in Europe. In this example we just show the process we followed for selecting the non ontological resources to be reused in the occupation domain. The description of this example is not intended to be exhaustive. It just describes the most important points. A detailed and complete description is presented in [13].

Activity 1. Search non ontological resources.

Following domain expert suggestions, we have searched existing occupation classification at the Ramon Eurostat Portal³³. In this high reliable Web site we found the following classifications in the occupation domain:

- Standard Occupational Classification System (SOC)
- International Standard Classification of Occupations (ISCO-88)
- International Standard Classification of Occupations, for European Union purposes, ISCO-88 (COM)

³² <http://www.seemp.org>

³³ <http://ec.europa.eu/eurostat/ramon/>

Activity 2. Assess the set of candidate non ontological resources.

Since the NeOn proposed guidelines for non ontological resource reuse was not ready, we just took into account the consensus criterion for assessing the non ontological resources. We also analyzed subjectively in an informal way and without using the precision and coverage criteria, whether the resources provided a rich terminology or not.

It was important for the project that resources focus on the current European reality, because the user partners involved in SEEMP are European, and the outcoming prototype will be validated in European scenarios. Thus, domain experts stated whether the resource was built with consensus by the European community or not.

For the construction of the assessment table we collected all the information of each non ontological resource in a table. This is shown in Table 13.

	Consensus
SOC	No
ISCO-88	No
ISCO-88(COM)	Yes

Table 13. Assessment Table for SEEMP Occupation Standards

Activity 3. Select the most appropriate non ontological resources.

We selected the following non ontological resource:

- International Standard Classification of Occupations, for European Union purposes, ISCO-88 (COM)

6.6. NeOn Approach for Non Ontological Resource Reengineering

In this section we present the NeOn approach for non ontological resource reengineering. We describe the preliminary NeOn proposal for carrying out the non ontological reengineering process. Then, we present an example of how to transform a classification scheme into an ontology using a pattern for reengineering non ontological resources.

6.6.1. NeOn General Model for Non Ontological Resource Reengineering

In a nutshell, the NeOn approach for non ontological resource reengineering considers as input a pool of non ontological resources and patterns for reengineering non ontological resources. Non ontological resources were selected by the non ontological resource reuse process; they include lexica, classification schemes, thesauri, etc. Regarding patterns for reengineering non ontological resources, they provide solutions to the problem of transforming non ontological resources into ontologies; they are included in the NeOn patterns repository.

Based on the software reengineering model presented in section 6.4 we propose the NeOn reengineering model for non ontological resource reengineering in Figure 28. The non ontological resource reengineering process consists of the following activities:

1. **Non Ontological Resource Reverse Engineering**, whose goal is to analyze a non ontological resource to identify its underlying components and create a representation of the resource at higher levels of abstraction. Since non ontological resources (as we already mentioned in section 6.3) can be implemented as XML files, databases or spreadsheet among others, we can consider them as software resources, and therefore, we use the software abstraction levels shown in Figure 26 to depict this activity.

2. **Non Ontological Resource Transformation**, whose goal is to generate a conceptual model from the non ontological resource. We propose the use of Patterns for Reengineering Non Ontological Resources (PR-NOR) to guide the transformation process according to the type of non ontological resource. We are currently working on the definition of these patterns, and the final result of this work will be included in D2.2.2b (Methods and tools for reengineering). In this deliverable we present the template used to describe the patterns for reengineering and one example of a pattern to transform a classification scheme with an adjacency list data model into an ontology.
3. **Ontology Forward Engineering**³⁴, whose goal is to output a new implementation of the ontology on the basis of the new conceptual model. We use the ontology levels of abstraction to depict this activity because they are directly related to the ontology development process.

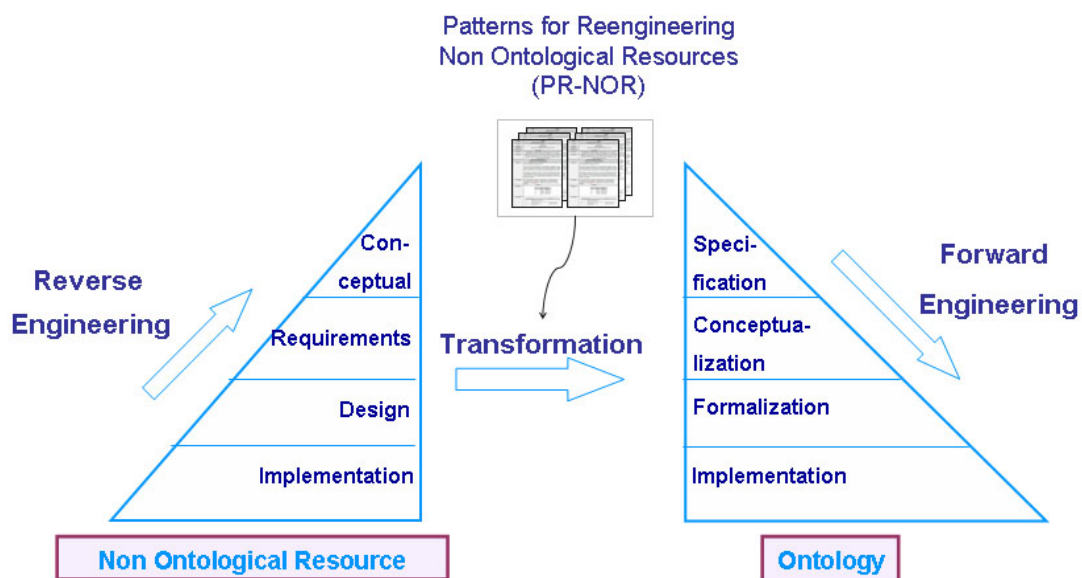


Figure 28. Reengineering Model for Non Ontological Resources

Every non ontological resource selected by the non ontological resource reuse process (explained in section 6.5) has to be transformed into an ontology by the non ontological resource reengineering process. Activities and tasks involved in this process are independent of the type of the non ontological resource, and are presented in section 6.6.2; however the techniques needed to carry out each activity or task could be highly dependent on the type of non ontological resource. This is an ongoing work whose preliminary results are presented in this deliverable. More detailed information on the non ontological reengineering process will appear on a second version of this deliverable and more information about the techniques to carry out the activities and tasks according to the different types of non ontological resources will be presented in D2.2.2b (Methods and tools for reengineering).

6.6.2. NeOn Activities and Tasks for Non Ontological Resource Reengineering

The goal of the Non Ontological Resource Reengineering process is to transform a non ontological resource into an ontology. The output of the process is an ontology. Table 14 shows the non ontological resource reengineering process filling card, which includes the definition, goal, input, output, who carries out the process and when the process should be carried out.

³⁴ Ontology Forward Engineering is defined in the NeOn Glossary of Activities [111].

Non Ontological Resource Reengineering	
<i>Definition</i>	
<div style="border: 1px solid black; padding: 5px;"> <p><i>Non Ontological Resource Reengineering</i> refers to the process of taking an existing non ontological resource and transforms it into an ontology³⁵.</p> </div>	
<i>Goal</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>Create an ontology from a non ontological resource.</p> </div>	
<i>Input</i>	<i>Output</i>
<div style="border: 1px solid black; padding: 5px;"> <p>One or more non ontological resources selected by the reuse process.</p> </div>	<div style="border: 1px solid black; padding: 5px;"> <p>An ontology.</p> </div>
<i>Who</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>Domain experts, software developers and ontology practitioners.</p> </div>	
<i>When</i>	
<div style="border: 1px solid black; padding: 5px;"> <p>After the non ontological resource reuse process and before the conceptualization activity.</p> </div>	

Table 14. Non Ontological Resource Reengineering Filling Card

The proposed tasks for the non ontological resource reengineering activities presented in section 6.6.1 are depicted in Figure 29.

³⁵ In this document we slightly modify the definition of Non Ontological Resource Reengineering from the NeOn Glossary of Activities [111].

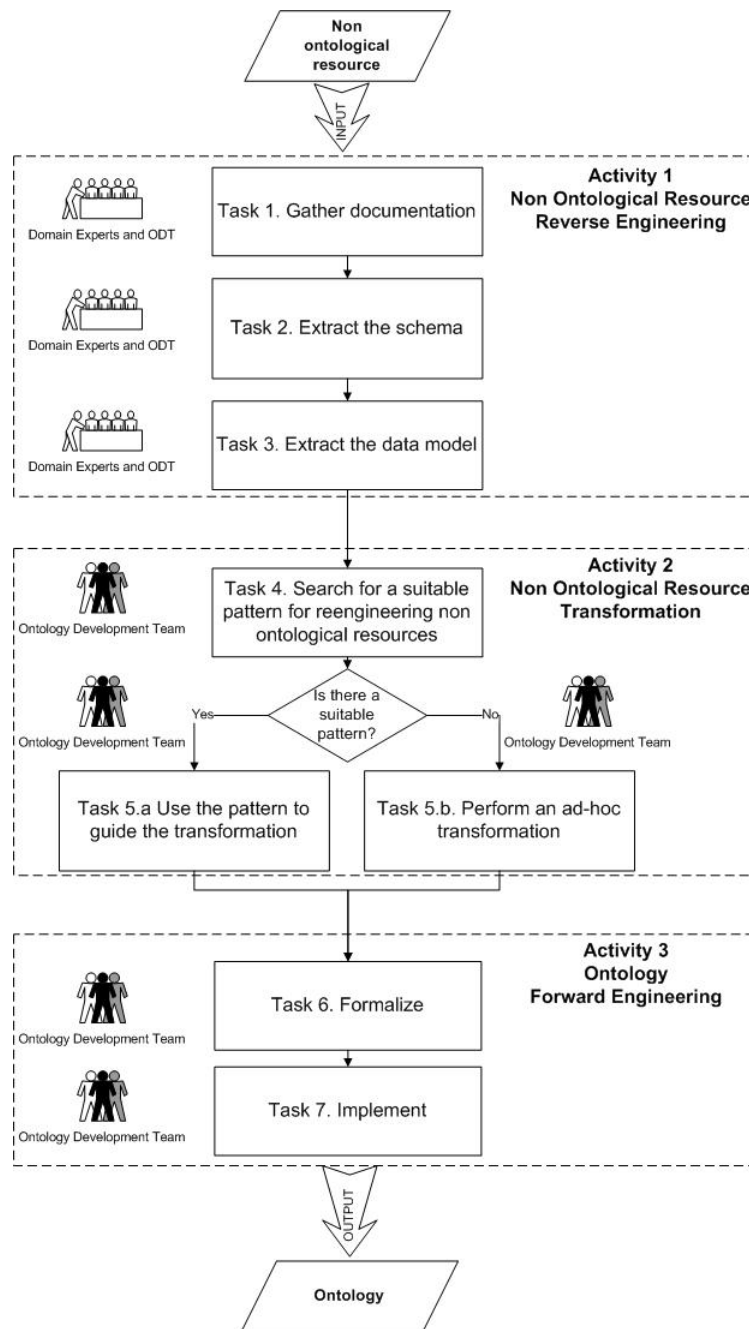


Figure 29. Proposed Activities in NeOn for the Non Ontological Resource Reengineering Process

In the following, we outline the tasks for carrying out the three activities involved in the non ontological resource reengineering process. As we already mentioned, more detailed information on this process guidelines will be provided in the second version of this deliverable.

Activity 1. Non ontological resource reverse engineering.

Task 1. Gather documentation.

The goal of this task is to search and compile all the available documentation about the non ontological resource including purpose, components; data model and implementation details. Domain experts and the ontology development team carry out this activity, taking as input the non ontological resource, searching in the non ontological resource web site and

in general purpose search engines, or requesting the documentation directly to the non ontological resource author.

Task 2. Extract the non ontological resource schema.

The goal of this task is to identify the non ontological resource schema including the conceptual components and their relationships. Domain experts and the ontology development team carry out this activity taking as input the non ontological resource and the documentation obtained in task 1. If the non ontological resource schema is not available in the documentation, the schema should be reconstructed manually or using a data modeling tool.

Task 3. Extract the data model.

The goal of this task is to find out how the non ontological resource schema and its content are represented in the data model. Domain experts and the ontology development team carry out this activity taking as input the non ontological resource, the documentation and schema. If the non ontological resource data model is not available in the documentation, the data model should be reconstructed manually or using a data modeling tool.

Activity 2. Non ontological resource transformation.

Task 4. Search for a suitable pattern for reengineering non ontological resources.

The goal of this task is to find out if there is any applicable pattern for reengineering non ontological resources useful to transform the non ontological resource into a conceptual model. The ontology development team carries out this activity taking as input the non ontological resource, the schema and data model. The search for a suitable pattern for reengineering non ontological resource should be done into the NeOn repository of patterns³⁶, according to the type of non ontological resource, the data model, and the transformation approach.

The transformation approach, explained in section 6.2, refers to: 1) transforming the non ontological resource schema into an ontology schema and the non ontological resource data into instances of the ontology; and 2) transforming the non ontological resource data into an ontology schema.

Task 5.a. Use patterns for reengineering to guide the transformation.

The goal of this task is to apply the reengineering pattern obtained in task 4 to transform the non ontological resource into a conceptual model. If a suitable pattern for reengineering non ontological resource is found then the ontology development team creates the conceptual model from the non ontological resource following the procedure established in the pattern for reengineering.

Task 5.b. Perform and ad-hoc transformation.

The goal of this task is to establish an *ad-hoc* procedure to map the non ontological resource into a conceptual model, just in case a suitable pattern for reengineering was not found. The ontology development team carries out this task choosing a transformation approach and then building a procedure to carry out the chosen transformation. This *ad-hoc* procedure may be generalized to create a new pattern for reengineering non ontological resource.

³⁶ In the NeOn Project a repository of patterns, including reengineering ones, is being developed.

Activity 3. Ontology forward engineering.

Task 6. Formalize.

The goal of this task is to transform the conceptual model obtained in task 5.a or 5.b into a formalized model. The ontology development team carries out this task formalizing the conceptual model according to a knowledge representation paradigm as description logics, first order logic, etc.

Task 7. Implement.

The goal of this task is the ontology implementation in an ontology language. The ontology development team carries out this task implementing the formalized model obtained in task 6 in an ontology language.

6.6.3. Patterns for Reengineering Non Ontological Resources

Reengineering Ontology Design Patterns (RePs) were defined in D2.5.1 [94] as transformation rules applied in order to create a new ontology (target model) starting from elements of an ontological resource that can be either an ontology, or a non ontological resource. As any other pattern, RePs provide solutions to recurrent situations regarding the transformation process. In this section, we present the template used to describe the patterns for reengineering non ontological resources (PR-NOR), and then an example of a reengineering pattern identified in our ongoing research work on transforming classification schemes into ontologies.

To present the reengineering patterns we adapted the tabular template used in NeOn D5.1.1 [110]. The adapted template is shown in Table 15.

Slot	Value
General Information	
<i>Name</i>	Name of the component
<i>Identifier</i>	An acronym composed of: component type + component + number
<i>Type of Component</i>	Pattern for Reengineering Non Ontological Resource (PR-NOR)
Use Case	
<i>General</i>	Description in natural language of the reengineering problem addressed by the reengineering pattern.
<i>Examples</i>	Description in natural language of an example of the reengineering problem.
Pattern for Reengineering Non Ontological Resource	
Resource to be Reengineered	
<i>General</i>	Description in natural language of the non ontological resource.
<i>Example</i>	Description in natural language of an example of the non ontological resource.
Graphical Representation	
<i>General</i>	Graphical representation of the non ontological resource.
<i>Example</i>	Graphical representation of the example of non ontological resource.
Designed Ontology	
<i>General</i>	Description in natural language of the ontology created after applying the pattern for reengineering the non ontological resource.

<i>Graphical Representation</i>	
<i>(UML) General Solution Ontology</i>	Graphical representation of the ontology created for the non ontological resource being reengineered.
<i>(UML) Example Solution Ontology</i>	Example showing a graphical representation of the ontology created for the non ontological resource being used.
<i>How to Reengineer</i>	
<i>General</i>	Description in natural language of the general reengineering process, using a sequence of activities.
<i>Example</i>	Description in natural language of the reengineering process applied to the non ontological resource example, using the above sequence of activities.
<i>Relationships</i>	
<i>Relations to other modeling components</i>	Description of any relation to other PR-NOR patterns or other design patterns.
<i>Design Variants</i>	
<i>General</i>	Description in natural language of possible modifications to the reengineering process.

Table 15. Pattern for Reengineering Non Ontological Resource Template

The following pattern for reengineering non ontological resource suggests a guide to transform a classification scheme into an ontology. The classification scheme is modelled with an adjacency list data model. The goal of this pattern is to create a taxonomy³⁷ from the classification scheme. The reengineering pattern shown in Table 16 uses the water area classification scheme presented in section 6.3 in order to exemplify the reengineering process.

Slot	Value
<i>General Information</i>	
<i>Name</i>	Classification to Taxonomy (adjacency list model)
<i>Identifier</i>	PR-NOR-CLTX-01
<i>Type of Component</i>	Pattern for Reengineering Non Ontological Resource (PR-NOR)
<i>Use Case</i>	
<i>General</i>	Reengineering a classification scheme built following the adjacency list model to design a taxonomy
<i>Examples</i>	Suppose that someone wants to build an ontology based on the water areas classification published by FAO. This classification scheme is delivered in a table, described in Figure 23, with an adjacency list data model.
<i>Pattern for Reengineering Non Ontological Resource</i>	
<i>Resource to be Reengineered</i>	
<i>General</i>	A non ontological resource holds a classification scheme built following

³⁷ According to D5.1.1 [110] a taxonomy is the way of organizing an ontology as a hierarchical structure of classes only related by subsumption relations.

	<p>the adjacency list model.</p> <p>A classification scheme is a rooted tree of concepts, in which each concept groups entities by some particular degree of similarity. The semantic of the hierarchical relation between parents and children concepts may vary depending of the context</p> <p>The adjacency list model [23] for hierarchical classifications proposes to create an entity which holds a list of items with a linking column associated to their parent items.</p>																														
<p><i>Example</i></p>	<p>The FAO classification for water areas groups them according to some different criteria as environment, statistics, and jurisdiction, among others. This classification scheme is available at: http://www.fao.org/figis/servlet/RefServlet</p>																														
<p>Graphical Representation</p>																															
<p><i>General</i></p>	<table border="1" data-bbox="778 719 1150 931"> <thead> <tr> <th>Category Code</th> <th>Category Name</th> <th>Parent Category Code</th> </tr> </thead> <tbody> <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> </tbody> </table>	Category Code	Category Name	Parent Category Code	1	Category1	Null	2	Category2	Null	3	Category3	1	4	Category4	1	5	Category6	3	6	Category7	4						
Category Code	Category Name	Parent Category Code																													
1	Category1	Null																													
2	Category2	Null																													
3	Category3	1																													
4	Category4	1																													
5	Category6	3																													
6	Category7	4																													
...																													
<p><i>Example</i></p>	<table border="1" data-bbox="719 1032 1209 1375"> <thead> <tr> <th>Id</th> <th>Category Name</th> <th>Parent</th> </tr> </thead> <tbody> <tr> <td>20000</td> <td>Water area</td> <td>1</td> </tr> <tr> <td>21000</td> <td>Environmental area</td> <td>20000</td> </tr> <tr> <td>22000</td> <td>Fishing Statistical area</td> <td>20000</td> </tr> <tr> <td>24020</td> <td>Jurisdiction 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>21004</td> <td>Sub Ocean</td> <td>21000</td> </tr> <tr> <td>21005</td> <td>Large Marine ecosystem</td> <td>21000</td> </tr> </tbody> </table>	Id	Category Name	Parent	20000	Water area	1	21000	Environmental area	20000	22000	Fishing Statistical area	20000	24020	Jurisdiction area	20000	21001	Inland/marine	21000	21002	Ocean	21000	21003	North/South/Equatorial	21000	21004	Sub Ocean	21000	21005	Large Marine ecosystem	21000
Id	Category Name	Parent																													
20000	Water area	1																													
21000	Environmental area	20000																													
22000	Fishing Statistical area	20000																													
24020	Jurisdiction area	20000																													
21001	Inland/marine	21000																													
21002	Ocean	21000																													
21003	North/South/Equatorial	21000																													
21004	Sub Ocean	21000																													
21005	Large Marine ecosystem	21000																													
<p>Designed Ontology</p>																															
<p><i>General</i></p>	<p>The generated ontology will be based on the taxonomy architectural pattern (AP-TX-01) [110].</p> <p>Each category in the classification scheme is mapped to a class, and the semantic of the relationship between children and parent categories is mapped to <i>subClassOf</i> relations.</p>																														
<p>Graphical Representation</p>																															
<p><i>(UML) General Solution Ontology</i></p>																															

	<pre> classDiagram Class0 < -- Class1 Class0 < -- Class2 Class1 < -- Class3 Class1 < -- Class4 Class3 < -- Class5 Class4 < -- Class6 </pre>
<p>(UML) Example Solution Ontology</p>	
<p>How to Reengineering</p>	
<p>General</p>	<ol style="list-style-type: none"> 1. Select all categories which do not have a parent category in the adjacency list. 2. If there is just one category X without parent then: <ol style="list-style-type: none"> 2.1. Create a class for the category X (<i>rootClass</i>). 2.2. For each category Y which has as parent X (parent Id in the adjacency list) <ol style="list-style-type: none"> 2.2.1. Create a class CY and set the <i>subClassOf</i> relation between CY and <i>rootClass</i>. 2.2.2. For each category Z which has as parent the category Y. <ul style="list-style-type: none"> • Create a class CZ and set the <i>subClassOf</i> relation between this CZ and CY. • Follow the same approach with the subsequent child categories, until you have iterated through all the categories of the classification scheme. 3. If there is more than one category without parent then: <ol style="list-style-type: none"> 3.1. Create an <i>ad-hoc</i> class (<i>rootClass</i>) 3.2. For each category X which does not have a parent category. <ol style="list-style-type: none"> 3.2.1. Create a class CX and set the <i>subClassOf</i> relation between this CX and <i>rootClass</i>. 3.2.2. For each category Y which has as parent the category X <ul style="list-style-type: none"> • Create a class CY and set the <i>subClassOf</i> relation between CY and CX. • Follow the same approach with the subsequent child categories, until you have iterated through all the categories of the classification scheme.

	<pre> graph TD Start(()) --> Select([Select Categories without parent]) Select --> Decision{ } Decision -- "[Categories without parent = 1]" --> CreateRoot([Create a class (rootClass) for the category without parent]) CreateRoot --> IdentifyX([Identify the categories X which have as parent the category without parent]) IdentifyX --> ForEachX([For each category X create a class CX and assert that CX is subClassOf rootClass]) ForEachX --> IdentifyY1([Identify the categories Y which have as parent the category X]) IdentifyY1 --> ForEachY1([For each category Y create a class CY and assert that CY is subClassOf CX]) ForEachY1 -.-> Ellipsis1[...] Decision -- "[Categories without parent > 1]" --> CreateAdHoc([Create an ad hoc class (rootClass)]) CreateAdHoc --> ForEachX2([For each category X without parent create a class CX and assert that CX is subClassOf rootClass]) ForEachX2 --> IdentifyY2([Identify the categories Y which have as parent the category X]) IdentifyY2 --> ForEachY2([For each category Y create a class CY and assert that CY is subClassOf CX]) ForEachY2 -.-> Ellipsis2[...] Note[Follow the same approach with the subsequent child categories, until you have iterated through all the categories of the classification scheme] </pre>
<p><i>Example</i></p>	<ol style="list-style-type: none"> 1. Create a Water area class as the root of the ontology. 2. For each category X which has as parent id the water area id (20000). <ol style="list-style-type: none"> 2.1 Create a class CX, and assert that CX is <i>subClassOf</i> the Water area class. 2.2 For each category Y which has as parent the category X. <ol style="list-style-type: none"> 2.2.1 Create a class CY and set the <i>subClassOf</i> relation between CY and the CX. 2.2.2 Follow the same approach with the subsequent child categories, until you have iterated through all the categories of the water area classification.
Relationships	
<p><i>Relations to other modeling components</i></p>	<p>Use the AP: TX-AP-01 [110].</p>

Table 16. Example of Pattern for Reengineering Non Ontological Resource

6.7 Conclusions and Future Work

In this chapter we have presented a three level categorization of non ontological resources according to three different features: type of non ontological, resource, designed data model and implementation. Moreover, we presented the NeOn proposal of guidelines for reusing non ontological resources with an example of human resources management classifications in the

SEEMP Project. Further work needs to be done to consider more criteria for the assessment of the non ontological resources, having in mind that the final goal is the development of ontologies.

Regarding reengineering non ontological resources, we have presented the NeOn proposal of a reengineering model for non ontological resources and also preliminary guidelines to carry out the proposed activities in the non ontological resource reengineering model. Additionally, we take advantage of the non ontological resource data model to define patterns for reengineering non ontological resources. We presented a template for describing those patterns and a pattern for reengineering a classification scheme into an ontology. Further work needs to be done to consider data models of the other non ontological resources such as thesauri and lexica. If we can identify data models as we did for classifications schemes we will be able to create more patterns to guide the reengineering process.

7. Ontological Resource Reuse

In this chapter we present how to build ontology networks by reusing ontological resources. For this purpose, we provide methodological guidelines to help software developers and ontology practitioners to reuse ontologies and ontology statements.

7.1. Introduction

As we already mentioned in chapter 4, the NeOn Methodology presents 9 different scenarios for building networks of ontologies. One of these scenarios is *Building Ontology Networks by Reusing Ontological Resources*. In this scenario, software developers and ontology practitioners should analyse whether existing ontological resources can be reused to build an ontology network or not. The underlying principle is that reusing existing ontological resources time and costs associated to the ontology development.

The reuse of ontological resources is encouraged by a recent increase in the number of online available ontologies, ontology libraries and repositories³⁸. The NeOn Glossary of Activities [111] defines *ontology reuse* as the activity of using an ontology or an ontology module in the solution of different problems.

The ontology reuse process is often influenced by the type of ontology to be reused as presented in Figure 30. Based on the terms extracted from the ontology requirements specification document (ORSD), we can consider the reuse of general or common ontologies and/or the reuse of domain ontologies.

On the one hand, general or common ontologies provide conceptualization of generic topics such as time and space. Given the generality of the described topic, it is common to have several ontologies on the same topic, each of them taking a different standpoint in the conceptual specification of the topic. When reusing one of these ontologies, the ontology engineer needs to be aware of the different views and assumptions the ontologies rely on. Further, because these ontologies are often well-formed and self-contained theories, it is common practice to reuse them as a whole. Guidelines for reusing general or common ontologies are provided in section 7.3.

On the other hand, domain ontologies provide knowledge of a concrete domain such as medicine, pharmacy, fisheries, etc. Such ontologies can be helpful in cases when a domain ontology in the same domain is being built. Unlike in the case of general or common ontologies, which are reused as a whole, we distinguish different levels of granularity in the reuse of domain ontologies, as it is shown in Figure 30.

- Ontologies can be reused as a *whole* if they closely meet the expectations and the needs of the ontology engineer. Guidelines for reusing domain ontologies as a whole are provided in section 7.4.
- In certain cases, only one part or *module* of a domain ontology is relevant for reuse. We consider a module [33] as a part of the domain ontology that defines the relevant set of terms. For example, when building an ontology about lung cancer one does not need to reuse an entire ontology about the human body, it suffices to reuse a module describing concepts related to lung. Guidelines for reusing ontology modules will be included in the next version of this deliverable.

³⁸ See for example a list of novel ontology search engines described at: <http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/SemanticWebSearchEngines>.

- Novel developments in the area of semantic search engines facilitate the reuse of ontological knowledge at the *statement* level, allowing the ontology engineer a maximal control of the material that is being reused. Guidelines for reusing ontology statements are provided in section 7.5, considering that an ontology statement (or triple)³⁹ contains the following three components: *subject*, *predicate*, and *object*.



Figure 30. Different Types of Ontological Resource Reuse

It is very important for ontology practitioners and software developers to take an intermediate approach, i.e., a solution between not reusing any knowledge resource at all and reusing too many knowledge resources. The first approach can lead to a creative solution if successful, but if not, it can waste a lot of time; and the second approach, if successful, it can save resources and lead to a good result, but on the contrary, it can result in knowledge resources being reused inappropriately and in wasting time.

Based on the above situations in this deliverable, we modify and further specify the NeOn definition of ontology reuse, and propose the following new ones:

³⁹ <http://www.w3.org/TR/rdf-concepts/>

- *Ontological Resource Reuse* is defined as the process of using available ontological resources (ontologies, modules, statements, or ontology design patterns) in the solution of different problems (e.g., the development of different ontology-based applications, the activity of ontology aligning (as background knowledge), etc.). We distinguish between: ontology reuse, ontology module reuse, ontology statement reuse, and ontology design pattern reuse, as Figure 31 shows.

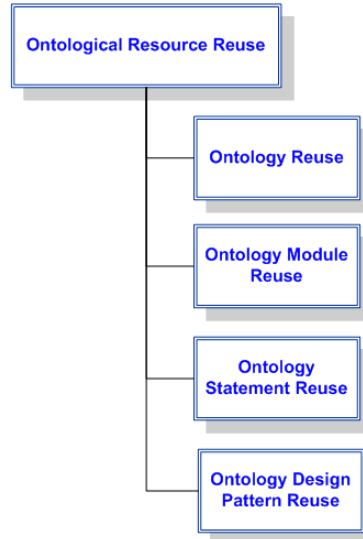


Figure 31. Ontological Resource Reuse Definitions

- *Ontology Reuse* is redefined as the process of using ontologies in the solution of different problems.
- *Ontology Module Reuse* is defined as the process of using ontology modules in the solution of different problems.
- *Ontology Statement Reuse* is defined as the process of using ontology statements in the solution of different problems.
- *Ontology Design Pattern Reuse* is defined as the activity of using ontology design patterns in the solution of different modelling problems during the development of new ontologies.

In this chapter, we present how to build ontology networks by reusing existing ontological resources (ontologies, ontology modules and ontology statements). This is related with scenarios 2 and 3. As we already mentioned in section 4.1, the reuse of ontology design patterns is treated in another scenario (scenario 7), and thus described in chapter 8.

In this chapter, we provide a set of general criteria to be taken into account during the ontological resource reuse process in section 7.2. The rest of the chapter includes methodological guidelines for reusing ontological resources based on the type of ontological resource and based on different levels of granularity of the ontological resource.

In this chapter we provide guidelines for reusing two different types of ontologies: general or common ontologies in section 7.3 and domain ontologies as a whole in section 7.4. We also provide guidelines for reusing ontology statements in section 7.5.

Unlike other chapters in this deliverable, this chapter does not include a section about the state of the art on ontological resource reuse, because in NeOn deliverable D2.2.1 [98] a variety of ontology evaluation and selection tools, methods and techniques that support the process of ontology reuse has been overviewed. To our knowledge, however, there are no methodologies yet that focus on the activity of reusing ontological resources.

7.2. General Criteria for Ontological Resource Reuse

This section includes general criteria for being used during the reuse of any type of ontological resources to be reused in the development of ontology networks. Such criteria have been obtained from different experiences in the reuse of ontological resources through the construction of ontologies in different projects.

These general criteria will be used together with other specific criteria for each type of ontological resource reuse, and further explanations will appear in the different activities proposed in the guidelines of sections 7.3, 7.4 and 7.5.

- **Ontological Resource Domain Coverage.** It means analysing whether the ontological resource covers (totally or partially) the set of competency questions identified in the ORSD.

This criteria can be partially analyzed by means of checking if the terms X, Y, etc., which are essential for the new development, appear in the ontological resource to be reused [60].

- **Ontological Resource Reuse Effort.** It refers to the estimation of the effort needed for accessing and using the ontological resource. In this case, we can mention the following subcriteria:
 - *Economic cost:* if the ontological resource has any type of license, then the cost of acquisition and/or exploitation should be taken into account [60].
 - *Time required for accessing the ontological resource:* if the ontological resource is accessible in slow servers or servers with bad connectivity, the time used for accessing should be taken into account.
- **Ontological Resource Understandability Effort.** It refers to the estimation of the effort needed for understanding the ontological resource. In this case, subcriteria to be taken into account are:
 - If the ontological resource is well documented.
 - If the ontological resource have references to documentation sources and domain experts easily available.
 - If the ontological code is clear enough.
- **Ontological Resource Modularization Effort.** It refers to the estimation of the effort needed for extracting a part of the ontology useful for the requirements of the ontology to be developed, if the ontology is not going to be reused as a whole. In this case, we can mention the following subcriteria:
 - If the ontology is already modularized.
 - If there are tools supporting the identification and extraction of ontology modules from an ontology.
- **Ontological Resource Integration Effort.** It refers to the estimation of the effort needed for integrating the ontological resource in the ontology being developed. In this case, we can consider the following subcriteria:
 - Similarity between the ontological resource naming conventions and the naming conventions used in the ontology being developed.
 - Similarity between the ontological resource implementation language and the implementation language to be used in the ontology being developed.
 - Contradictory bits of knowledge between the ontological resource to be reused and the ontology being developed.
 - Adaptation of definitions and axioms to satisfy the existing restrictions of the reasoner.

- Creation of new axioms and/or relations needed to integrate the ontological resource to be reused in the ontology being developed.
- **Ontological Resource Reliability.** It means analysing whether we can trust in the ontological resource. In this case, subcriteria to be taken into account are:
 - Check if there are tests available for the ontological resource.
 - Check if the ontological resource has been properly evaluated.
 - Check if the ontological resource is supported by a contrasted theory, in the case of common or general ontologies.
 - Check if the development team of the ontological resource is reliable.
 - Check if the ontological resource is reliable. If the ontological resource has been developed as a simple academic example, such ontological resource is less reliable than other resources developed to be used in real projects.
 - Check if there are well known projects or ontologies that are reusing the ontological resource [79].

7.3. Proposed Guidelines for Reusing General or Common Ontologies

A general [117] or common ontology [84] represents knowledge reusable in different domains. They are usually based on well studied theories: mereology, which formalizes parthood relation; topology, which formalizes connection relation; time theories, which formalize terms like *time interval*, *time point*, etc. The goal of reusing general or common ontologies⁴⁰ is to find and select one or several general or common ontologies to be used in the ontology network being developed. Its output is a set of general or common ontologies.

Table 17 shows the filling card for the general or common ontology reuse process, including the definition, goal, inputs and outputs, who carries out the process and when the process should be carried out.

The reuse of common ontologies consists basically in finding and selecting existing ontologies to be reused in the new ontology development. In this activity, two different situations can be considered:

- *Situation 1*, in which there is a previous comparative study on the theory that supports the type of common ontology to be reused (e.g. a study on time modeling). A theory is considered here as a system of definitions, axioms and theorems that can be formal, semi-formal or informally represented.
- *Situation 2*, in which there is not such a previous comparative study on the theory.

In this section we include the NeOn methodological detailed guidelines for each aforementioned situation, and an example for situation 2, which in fact includes situation 1.

For the sake of clarity, we explain the different activities for carrying out the whole process considering the reuse of one ontology. To reuse more than one, the described process should be iteratively performed.

⁴⁰ Henceforth, we refer to general or common ontologies, just as common ontologies.

General or Common Ontology Reuse	
<p><i>Definition</i></p> <p>General or Common Ontology Reuse refers to the process of using general or common ontologies in the solution of different problems.</p>	
<p><i>Goal</i></p> <p>The goal of this process is to find and select general or common ontologies to be integrated in the ontology network being developed.</p>	
<p><i>Input</i></p> <p>Competency questions (CQs) included in the ORSD of the ontology network to be developed, and the implementation language of such ontology.</p> <p>Optionally, there may be a set of tables comparing across the same criteria the candidate ontologies to be reused.</p>	<p><i>Output</i></p> <p>A general or common ontology integrated in the ontology network being developed.</p>
<p><i>Who</i></p> <p>Software developers and ontology practitioners involved in the ontology development. The help of an ontology practitioner familiarized in formal ontologies/theories may be required.</p>	
<p><i>When</i></p> <p>The general or common ontology reuse process should be carried out after the ontology specification activity.</p>	

Table 17. Common Ontology Reuse Filling Card

7.3.1. Detailed Guidelines for Situation 1: the comparative study exists

Figure 32 shows the activities for carrying out the common ontology reuse process in the case of a comparative study between the features of the theory and the existing common ontologies supporting such theory is already available.

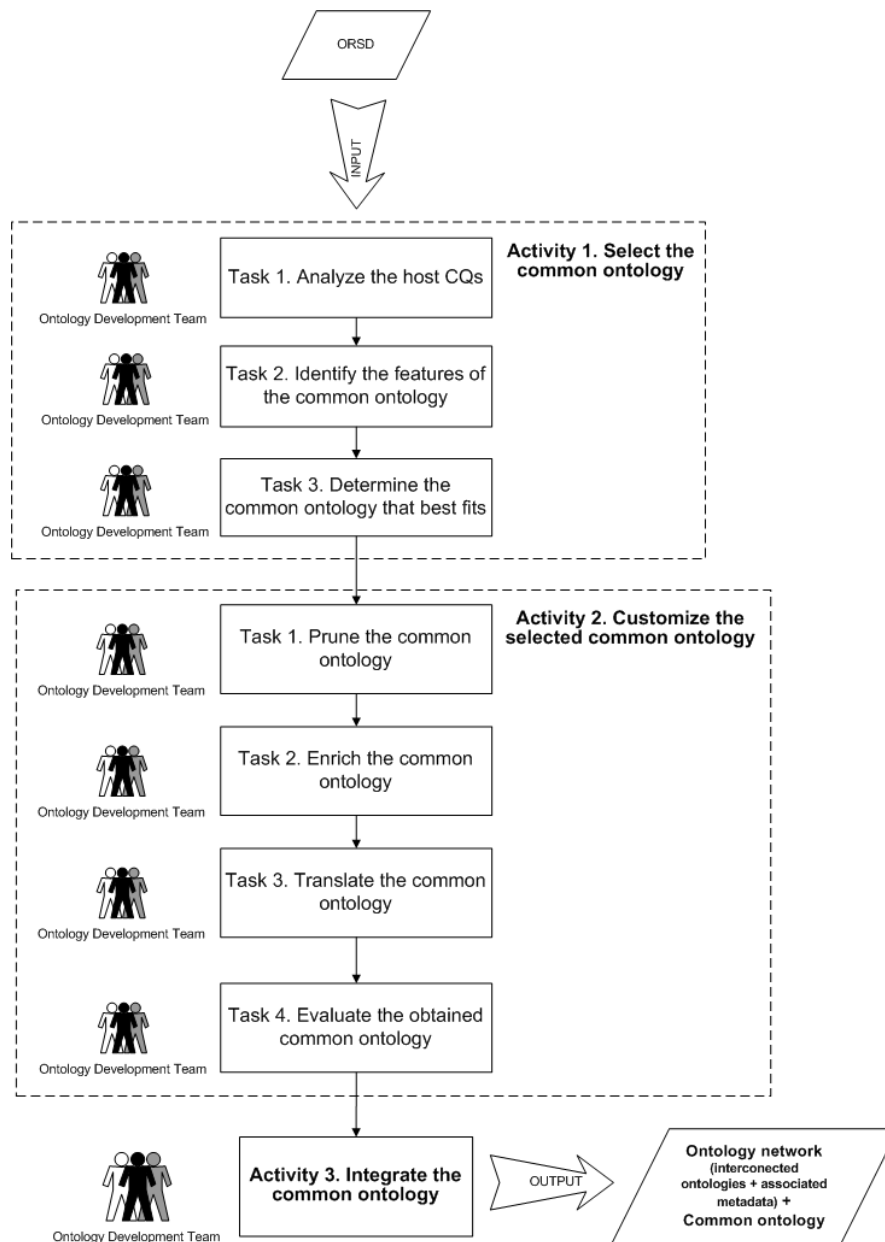


Figure 32. Activities for Reusing Common Ontologies in Situation 1

In this situation, a comparative study (e.g., in the form of a table), which provides the features of the most known ontologies supported by the theory, is available. Thus, in this case it is not necessary to search for someone familiarized with formal ontologies to carry out the study of the support theory.

For example, a table that describes a set of available time ontologies is shown in [46]. Each row of the table represents a set of definitions (or axioms) considered in the support theory, and each column represents an ontology. A cell (i, j) has the value yes if and only if the ontology O_j provides a formalization for the set of definitions (or axioms) D_i .

The activities shown in Figure 32 are explained in detail in the following:

Activity 1. Select the common ontology that best fits the features required by the host⁴¹ ontology, out of those ontologies represented in the table.

This activity is divided in the following tasks:

Task 1. Analysis of the host CQs.

The CQs of the host ontology are reformulated using the characteristic vocabulary of the support theory. For example, if a mereology ontology is going to be reused in a Pharmaceutical Product ontology (PPO), then each PPO CQ should be formulated using mereology vocabulary when possible. For instance, the CQ *which is the drug composition?* Could be reformulated in this way: *which are the parts of the drug?*

Task 2. Identification of the features of the common ontology to be reused.

It is set up what typical definitions, axioms and principles of the support theory are needed in the host application (or ontology). For example, it is possible that *part of* transitivity is needed in PPO, but not the binary sum.

Task 3. Determining the common ontology that best fits the features.

There are several criteria to take into account:

- a) The simplest way of establishing which common ontology achieves the best fit is counting how many features are implemented by each common ontology, and selecting the one that obtains the greater quantity.
- b) A more elaborated criterion would take into account, for example, the complexity of the definitions (and axioms) implemented by each ontology.
- c) Clarity.
- d) Minimum loss of knowledge in the translation to the language of the host ontology.
- e) Less modifications in the common ontology, etc.

The decision about the priority of the criteria depends on the host ontology intended uses.

Activity 2. Customize the selected common ontology according to the needs of the host ontology.

This activity is divided in the following tasks:

Task 1. Prune the common ontology according to the features that are really needed.

For example, if the feature “binary sum” is not needed in the mereology ontology, its definition and those depending on it should be removed.

Task 2. Enrich the common ontology.

For instance, if the common ontology does not include the reflexivity axiom of *part of*, and it is required, it should be added.

Task 3. Translate the common ontology into the implementation language of the host ontology.

The ontology can be automatically translated. If there are important loss of knowledge, it should be re-translated by hand.

Task 4. Evaluate the obtained common ontology.

The ontology resulting from task 3 should be evaluated mainly to ensure that tasks 1, 2 and 3 have been correctly performed.

⁴¹ We mean with “host ontology” the ontology (network) that is being developed

Activity 3. Integrate the common ontology in the host ontology.

The customized common ontology is included in the destination ontology, and the global result is evaluated again.

7.3.2. Detailed Guidelines for Situation 2: the comparative study does not exist

Figure 33 shows the activities for carrying out the general ontology reuse process in the case of a comparative study between common ontologies does not exist.

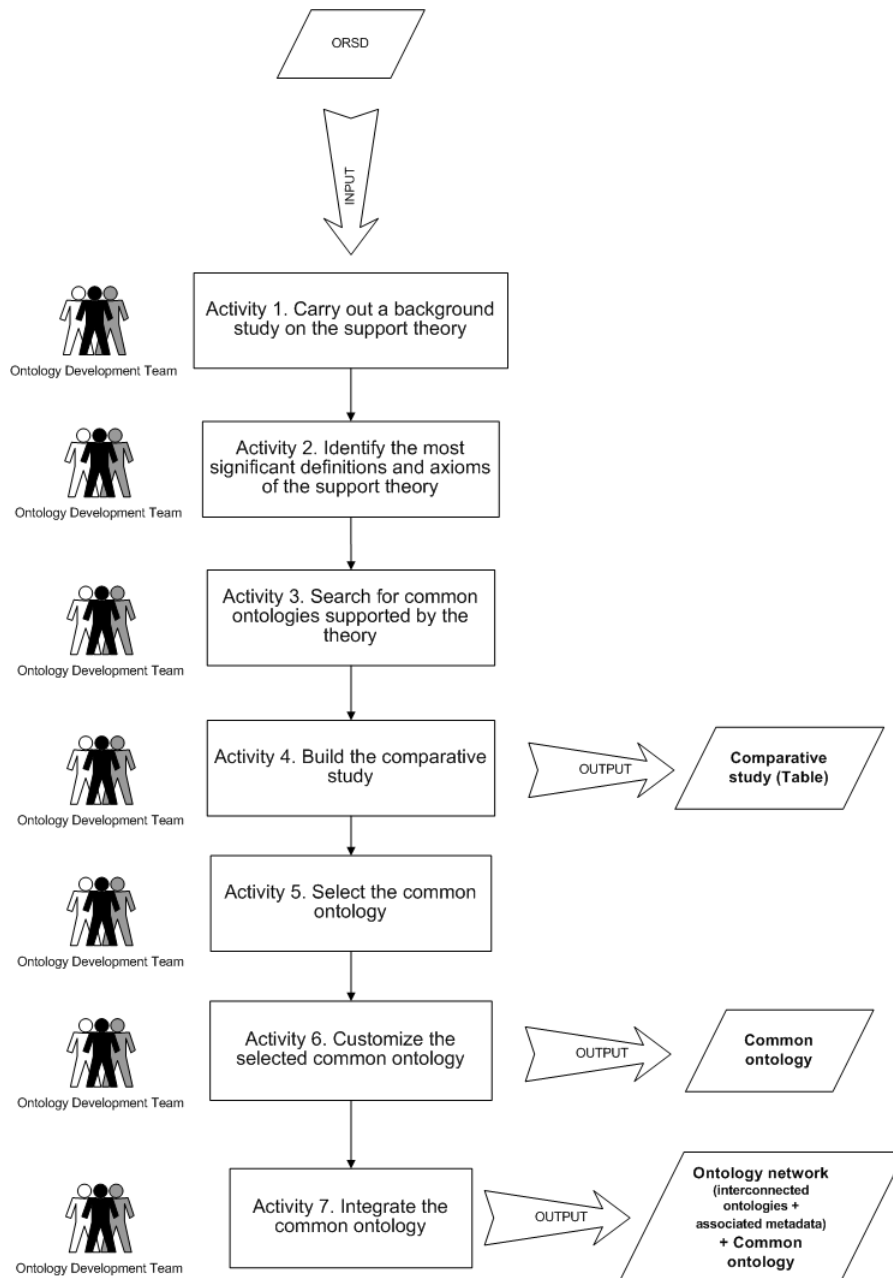


Figure 33. Activities for Reusing Common Ontologies in Situation 2

In this case, a comparative study (e.g., a table) comparing the features of the support theory with the existing common ontologies describing such a theory is not available. Thus, our proposal is to carry out such a comparative study in the form of a table in order to use it for choosing or selecting the common ontology that best fits with the requirements of the ontology to be developed.

The activities shown in Figure 33 are explained in detail in the following:

Activity 1. Carry out a background study on the support theory.

The ontology development team should study the theory (and its variants) that supports the type of common ontology to be reused. Thus, for example, if a mereology ontology is going to be reused, then literature on different mereologies should be analyzed. This activity may require the help of someone familiarized with formal ontologies.

It is important to mention that this activity can be time and resources consuming, but the result can be exploited in further projects. So, for example, if the parthood relation is going to be formalized in different projects, the study on mereology can be profited in all of them.

Activity 2. Identify the most significant definitions and axioms that characterize the support theory.

The most important definitions and axioms (e.g. transitivity of *part of* in mereology), and those that distinguish the different variants of the support theory should be identified. It is distinguished, for example, between atomistic mereologies and not atomistic mereologies. The first ones are characterized by assuming that every entity has atoms. An atom is an entity that does not have other parts than itself.

Activity 3. Search for common ontologies supported (partially or completely) by the theory.

The ontology development team should search for ontologies implemented in a computable language, for example, KIF [57], OWL [88], RDF(S) [24], CommonKADS Modeling Language (CML) [101], etc.

Activity 4. Build a comparative study of features versus ontologies in the form of a table.

It is assumed that each row represents the set of definitions (or axioms) identified in activity 2, and each column, the ontologies found in activity 3. The ontology development team, in collaboration with someone familiarized with formal ontologies, studies what features each ontology has. Following the examples shown in activity 2, the table should show for each mereology ontology if it formalizes the transitivity of *part of*, whether it is atomistic or not, etc.

Activities 5, 6 and 7 correspond with activities 1, 2 and 3 of situation 1, respectively, which are described in section 7.3.1.

7.3.3. Example of Situation 2

In this section an example of reusing a mereology ontology in a Pharmaceutical Product ontology (PPO) is presented. PPO will be part of the support collaboration in pharmaceutical industry, which is concerned with an infrastructure and its APIs to bridge the currently used proprietary systems for managing financial and product knowledge interoperability in the networks/clusters of pharmaceutical labs, companies and distributors in Spain [65]. The composition of drugs, the interaction between them, etc. requires the formalization of the *part of* relation. Consequently, it seems reasonable to consider the reuse of a mereology ontology.

Recently, researchers like Gangemi and colleagues [55] and Massolo and colleagues [83] have proposed the use of mereology in ontology construction. This idea is embodied in Borst work [22], who built and used a mereology ontology in an engineering application for modeling, simulating and designing physical systems. The main difference between the work shown in this section and Borst work is that we analyze, reuse and customize a mereology ontology already built, and do not develop a mereology ontology from scratch.

Next, we show the activities that we have carried out to reuse a mereology ontology in the PPO.

Activities 1 and 2. Carry out a background study on the support theory and identify the most significant definitions and axioms that characterize the support theory.

A *mereology* is a formal theory of parts and associated concepts [22, 100]. We have said ‘a mereology’ instead of ‘the mereology’ because different assumptions can be taken into account in the formalization of parthood. Therefore, different mereologies can be proposed. In the following paragraphs we will show the set of mereologies presented by Varzi [118].

Theory *M*

Most of the authors agrees on the following core of axioms (named with A) and definitions (named with D) [118]:

- *A.1) Reflexivity.* Every object of the universe of discourse is a part of itself. For instance, the European Union (EU) is part of the EU.
- *A.2) Antisymmetry.* If an object x is a part of y , and y is a part of x , then x and y are the same object. For instance, if the territory T_1 is part of the territory T_2 , then the only way so that T_2 is part of T_1 is being T_1 and T_2 the same territory.
- *A.3) Transitivity.* If x is a part of y , and y is a part of z , then x is a part of z . For instance, Madrid is part of a Spain, and Spain is part of EU, therefore, Madrid is a part of EU.

A number of additional mereological predicates can be then introduced by definition. For example:

- *D.1) Proper part.* A proper part is a part that is other than the individual itself. For example, Spain is proper part of EU, since Spain is part of EU and they are different entities.
- *D.2) Direct part.* X is direct part of y if and only if x is proper part of y and there is no part between x and y ⁴². For example, Italy is direct part of EU, but Madrid is not, since Spain is a part between Madrid and EU.
- *D.3) Overlap.* The relation *overlaps* is defined as a sharing part. That is, x and y overlap if and only if there is a z such that z is part of x and part of y . For instance, Spain and Africa overlap, since Spain has territories in Africa (Canaries, Ceuta, Melilla, etc.).
- *D.4) Underlap.* The relation *underlaps* is defined as a sharing whole. That is, x and y underlap if and only if there is a z such that x and y are parts of z . For example, Portugal, Spain, France and Italy underlap because they share a common whole: EU.
- *D.5) Disjoint.* The *disjoint* relation is the logical negation of *overlaps*. For example, EU and USA are disjoint territories.

Theory *M* may be viewed as embodying the common core of any mereological theory. A.1-A.3 should be extended to build a mereology.

Minimal Mereology (*MM*)

A way to extend *M* is assuming the following decomposition principle [118]:

- *A.4) Weak supplementation principle.* Every object x with a proper part y has another part z that is disjoint from y . The domain of territories, for example, fulfills this principle. For example, given that Spain is proper part of the European Union (EU), then EU has other parts that are disjoint from Spain: Portugal, France, Italy, etc.

Most of the authors strengthen that P.4 should be incorporated to *M* as a further fundamental principle on the meaning of *part-of*. Other authors provide scenarios that could be counterexamples of this principle. However, it is far from being demonstrated that such supposed counterexamples have implications in computer applications.

⁴² <http://hcs.science.uva.nl/projects/NewKACTUS/library/lib/mereology.html>

Extensional Mereology (*EM*)

There is another stronger way to express decomposition:

- *A.5) Strong supplementation.* If y is not part of x , then there is a part of y that does not overlap with x . For example, given that Spain is not part of Africa, there is a part of Spain (e.g. Madrid) that is not part of Africa. A.5 implies A.4.

This theory is called 'extensional' because a theorem that can be demonstrated is:

- *T.1)* For all x and y such that x has proper parts or y has proper parts, x and y are identical if and only if x and y have the same proper parts, that is, for all z , z is proper part of x if and only if it is part of y . For example, the territory of Community of Madrid is the same as the one of Province of Madrid because both of them are composed by the same proper parts, that is, by the same municipalities.

Closure Mereology (*CM*)

Another way of extending M is by composition [119]:

- *A.6) Sum principle.* If x and y underlap, then there is a z such that, for all w , w overlaps z if and only if w overlaps x or w overlaps y . That is, if two objects underlap, then it may be assumed that there is a smallest object of which they are part (an object that exactly and completely exhausts both). For instance, Madrid and Barcelona underlap, since they are both parts of Spain. According to (A.6), there is an object made up exactly of Madrid and Barcelona.
- *A.7) Product principle.* If x overlaps y , then there is a z such that for all w , w is part of z if and only if w is part of x and w is part of y . That is, if two objects overlap, then it may be assumed that there is a largest object that is part of both (the common part at their junction). For example, Spain and Africa overlap, and it may be assumed that there is a largest object that is overlapped by both: Canaries, Ceuta, Melilla, etc.

Assumption of A.6 and A.7 is controversial. In fact, it is not obvious that the overlap of Spain and Africa makes an entity.

Closure Extensional Mereology (*CEM*)

The result of adding these axioms to MM or EM instead yields corresponding Minimal or Extensional Closure Mereologies, that is, CMM and CEM , respectively. In the presence of (A.4), (A.7) implies (A.5). Consequently, CMM and CEM are the same theory [119].

The entities whose conditional existence is asserted by (A.6) and (A.7) must be unique in the presence of extensionality. Thus, CEM supports the following definitions:

- *D.6) Binary sum.* $X + y$ is the z that fulfils that for all w , w overlaps z if and only if w overlaps x or w overlaps y . That is, $x + y$ is the smallest object of which x and y are part.
- *D.7) Binary product.* $X \cdot y$ is the z that fulfils that for all w , w is part of z if and only if w is part of x and w is part of y . That is, $X \cdot y$ is the largest object that is part of x and y .

General (classical) Mereology (*GM*)

Another way of extending M is through the following axiom schema:

- *A.8) Unrestricted fusion principle.* For every satisfied property or condition ϕ there is a z such that for all y , y overlaps z if and only if there is an x such that x satisfies ϕ and overlaps y . That is, there is an entity consisting of all those things that satisfy ϕ . For every satisfied property or condition ϕ there is an entity consisting of all those things that satisfy ϕ . For example, let's suppose that ϕ means: "country with more than 10 millions of inhabitants", then there is an object that consists of all the countries with more than 10 millions of inhabitants.

If (A.5) is satisfied, then at most one entity can satisfy the consequent of (A.8). Therefore, the operation of general sum (σ) can be defined:

- *D.8) General sum.* The general sum all x s satisfying ϕ is that z such that for all y , y overlaps z if and only if there is an x such that x satisfies ϕ and overlaps y . That is, the sum of ϕ s is the entity that consists of all entities that satisfy ϕ .

General Extensional Mereology (*GEM*)

The extensions of *MM* and *EM*, which yield the same extensional strengthening of *GM* [119], is the theory of General Extensional Mereology, or *GEM*, since (A.8) implies (A.7) and (A.7)+(A.4) imply (A.5) [103]. It is also clear that *GM* is extension of *CM* and *GEM* is extension of *CEM*, since (A.6) also follows from (A.8).

Atomistic Mereology

In an atomistic mereological theory, every element is made up of elements that are building blocks or atoms. To describe such a theory, the following definition can be provided:

- *D.9) Atom.* It is an element that does not have proper parts.

The atomistic axiom can be formulated in this way:

- *A.9) Atomicity.* Every object has at least a part that is an atom. For example, the administrative division of territories follows this axiom, since there are simple divisions that are not in divided its turn.

Figure 34 shows a diagram with all the theories presented in this section until now.

A mereology X (e.g. *GEM*) extended with the atomicity axiom is known as **AX** (e.g. *AGEM*).

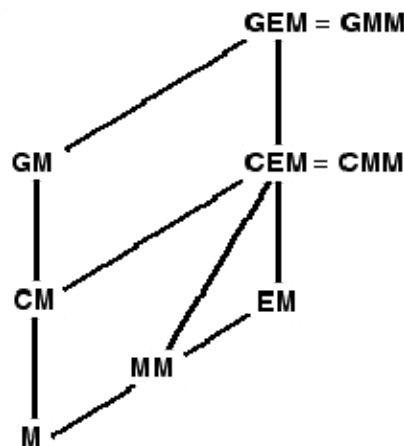


Figure 34. Hasse Diagram of Mereological Theories (from weaker to stronger, going uphill) [119]

Activity 3. Search for common ontologies supported (partially or completely) by the theory.

We have found the following ontologies that implements a mereology or contain mereology definitions:

- *KACTUS* [12] ontology library, implemented in CML [101], is maintained by the University of Amsterdam. Such a library contains Mereological Ontology (MO), which is an adapted version of Borst's proposals [22].
- *DOLCE* is one of the ontologies developed inside the WonderWeb European project⁴³ [83].

⁴³ <http://wonderweb.semanticweb.org/>

- The *Standard Upper Ontology*²³ (SUO) is the result of a joint effort to create a large, general-purpose, formal ontology [89]. It is promoted by the IEEE Standard Upper Ontology working group, and its development began in May 2000. The participants were representatives of government, academia, and industry from several countries. This ontology is implemented in KIF and Protégé format. SUO formally describes mereology and topology terms. The general predicates in this section of the ontology are adapted from Barry Smith, Borgo and colleagues, and Casati and Varzi mereologies.
- *Barry Smith and other authors* [104, 105, 29, 106] ontology in KIF is referred in the SUO web page. It represents various mereological definitions and axioms concerning boundaries and objects⁴⁴.
- *Borgo and colleagues* mereology is another ontology referred in SUO web page. These authors describe a set of definitions and axioms regarding mereology in [20, 21]. Such an ontology is currently implemented in KIF⁴⁵. The ontology formalizes a CEM mereology (excepting the product principle).
- *Casati and Varzi* [30] mereology can be also found implemented in KIF as a referred ontology in the SUO web page.

Activity 4. Build the table of features versus ontologies.

Table 18 represents each definition and axiom identified in activity 1, and the ontologies identified in activity 3. For the case of Borgo and colleagues ontology, the weak supplementation principle (A.4) is not directly represented, but can be inferred from the strong supplementation (A.5) one. As a consequence, if A.4 must be assumed in the host ontology, but not A.5, then A.4 should be completely implemented.

²³ <http://suo.ieee.org/>

⁴⁴ <http://suo.ieee.org/SUO/ontologies/Smith.txt>

⁴⁵ <http://suo.ieee.org/SUO/ontologies/Guarino.txt>

Theory	Principles and definitions	KACTUS MO	DOLCE	SUO	Smith et al.	Borgo et al.	Casati and Varzi
M	A.1) Reflexivity	No	No	Yes	Yes	Yes	No
	A.2) Antisymmetry	Yes	No	Yes	Yes	Yes	No
	A.3) Transitivity	Yes	No	Yes	Yes	Yes	No
	D.1) Proper part	Yes	Yes	Yes	No	Yes	Yes
	D.2) Direct part	Yes	No	No	No	No	No
	D.3) Overlap	Yes	Yes	Yes	Yes	Yes	Yes
	D.4) Underlap	No	No	No	No	No	No
	D.5) Disjoint	Yes	No	No	No	No	No
MM = M + (P.4)	A.4) Weak supplementation	Yes	No	No	Yes	Inferred	No
EM = M + (P.5) (Let's note that (P.5) implies (P.4))	A.5) Strong supplementation	No	No	No	Yes	Yes	No
CM = M + (P.6) + (P.7)	A.6) Sum principle	No	No	No	Yes	Yes	No
	A.7) Product principle	No	No	No	Yes	No	No
CEM = CM + (P.5)	D.6) Binary sum	No	Yes	Yes	Yes	Yes	Yes
	D.7) Binary product	No	No	Yes	Yes	Yes	Yes
GM = M + (P.8)	A.8) Unrestricted fusion principle	No	No	No	Yes	No	No
GEM = GM + (P.5)	D.8) General sum	No	Yes	No	Yes	No	No
AX = (P.9) + a mereology X	D.9) Atom	No	Yes	No	No	No	No
	A.9) Atomicity	No	No	No	No	No	No

Table 18. Features of Ontologies that implement Mereotopology Theories

Activity 5. Select the common ontology that best fits the features required by the host ontology, out of those ontologies represented in the table.

During the exposition of this activity, we will just focus in the four (of 61) competency questions (CQs) that allows us explaining our idea in a clearer way (see Table 19) [65].

Competency question identifier	Competency question in natural language like it is expressed in the ontology specification	Competency question using the vocabulary of mereology	Extracted terms
CQ1-R	<i>Which is the drug composition?</i>	Which are the parts of the drug?	- <i>part of</i>
CQ2-R	<i>What is the drug main active ingredient (molecule)?</i>	(It does not directly require mereotopology)	- <i>active ingredient</i> This term requires the definition of: - <i>part of</i>
CQ3-R	<i>What is the main substance of the composition?</i>	(It does not directly require mereotopology)	- <i>main substance</i> This term requires the definition of: - <i>part of</i>
CQ4-R	<i>Does the drug have interaction with another drug?</i>	Are there parts of the drug that interact with parts of another drug?	- <i>part of</i>

Table 19. Competency Question Analysis for Mereology Ontology Reuse

The tasks that we carried out to identify the mereology terms, axioms and definitions that are needed in the PPO are:

Task 1. Analysis of the host CQs.

Each CQ has been formulated using mereology vocabulary if possible. Table 19 shows how these CQs requires mereology terms in PPO.

Task 2. Identification of the features of the common ontology to be reused.

The inclusion of some properties (e.g. transitivity) has not been obvious. This indicates that the meaning of the CQs was not completely clear. That is, the study of the principles shown in Table 18 has helped us to identify ambiguities and, as we will see in the next paragraphs, it has helped us to precise the meaning of the CQs.

For the PPO case, the following formalization has been necessary:

- ❑ *A.1) Reflexivity.* It is necessary to ensure the right meaning of ontology terms. Thus, for example, if *part of* is not reflexive, then CQ4-R may not be correctly answered when the considered part is the whole drug.
- ❑ *A.2) Antisymmetry.* It will help the user to check constraints.
- ❑ *A.3) Transitivity.* It should be modeled if the different levels of the structure of components need to be provided. For example, Frenadol® is composed of paracetamol, dextrometorphan, and clorfenamine. In its turn, paracetamol is composed of an alcohol, an amino group and a carbonyl group. The alcohol is composed of oxygen and hydrogen, etc. Given that the inclusion of the transitivity axiom is low cost, we have decided to include all the components in the answer of CQs.
- ❑ *D.1) Proper part.* The formalization of this term eases the interpretation of the CQs. Thus, the very substance should not be a result of CQ1-R. However, it should be a result of CQ4-R, since the very substance can interact with a part of another substance.

- *D2) Direct part.* This term allows answering CQ1-R just in a level. Therefore, CQ1-R has been split into two competency questions: (CQ1-R') *which is the drug composition? (considering just a level)* and (CQ1-R'') *idem (considering all components)*.
- *A.4) Weak supplementation principle.* It will help the user to check constraints.
- *D.3) Overlap.* It is needed to formalize (A.4).
- *D.4) Underlap.* It is not necessary for the PPO at the moment.
- *D.5) Disjoint.* It is needed to formalize (A.4).
- *A.5) Strong supplementation principle* is not true if the bounds between atoms are not taken into account. We should remember that (A.5) implies that two entities are identical if and only if they have the same parts. However, isomers are not ruled out in Pharmaceutical Product ontology. An isomer is a chemical compound which has the same number and kind of atoms as another but differs in structural arrangement. If the structure of drugs is required, then a topology ontology is needed.
- *D. 6 and more) Sums and product.* It is not necessary for the PPO at the moment.
- *D.9 and A.9) Atom and atomicity.* It is not necessary for the PPO at the moment.

The aforementioned features are shadowed in Table 20.

Theory	Principles and definitions	<i>KACTUS MO</i>	<i>DOLCE</i>	<i>SUO</i>	<i>Smith et al.</i>	<i>Borgo et al.</i>	<i>Casati and Varzi</i>
M	A.1) Reflexivity	No	No	Yes	Yes	Yes	No
	A.2) Antisymmetry	Yes	No	Yes	Yes	Yes	No
	A.3) Transitivity	Yes	No	Yes	Yes	Yes	No
	D.1) Proper part	Yes	Yes	Yes	No	Yes	Yes
	D.2) Direct part	Yes	No	No	No	No	No
	D.3) Overlap	Yes	Yes	Yes	Yes	Yes	Yes
	D.4) Underlap	No	No	No	No	No	No
	D.5) Disjoint	Yes	No	No	No	No	No
MM = M + (A.4)	A.4) Weak supplementation	Yes	No	No	Yes	Inferred	No
EM = M + (A.5) (Let's note that (A.5) implies (A.4))	A.5) Strong supplementation	No	No	No	Yes	Yes	No
CM = M + (A.6) + (A.7)	A.6) Sum principle	No	No	No	Yes	Yes	No
	A.7) Product principle	No	No	No	Yes	No	No
CEM = CM + (A.5)	D.6) Binary sum	No	Yes	Yes	Yes	Yes	Yes
	D.7) Binary product	No	No	Yes	Yes	Yes	Yes
GM = M + (A.8)	A.8) Unrestricted fusion principle	No	No	No	Yes	No	No
GEM = GM + (A.5)	D.8) General sum	No	Yes	No	Yes	No	No
AX = (A.9) + a mereology X	A.9) Atom	No	Yes	No	No	No	No
	D.9) Atomicity	No	No	No	No	No	No

Table 20. Required Features for the Ontology to be developed (in grey)

Task 3. Determining the common ontology that best fits the features.

KACTUS MO has been selected in base of the criterion of domain coverage and adaptation of definitions and axioms to satisfy the existing restrictions of inference machine where PPO is expected to be reused. According to Table 20, KACTUS MO is the ontology that fulfills most criteria of those derived of the CQs. Besides, this ontology has been built to be easily reused in knowledge based systems, therefore, its definitions are easily adaptable to be used in software applications.

Activity 6. Customize the chosen common ontology according to the needs of the host ontology.

During the customization activity, we have carried out the following tasks: (6.1) prune the reused ontology according to the features that are really necessary; (6.2) enrich the ontology (e.g. with the *part of reflexivity axiom*); (6.3) translate the reused ontology from CML into OWL + SWRL; and (6.4) evaluate the obtaining ontology.

Activity 7. Integrate the common ontology in the host ontology.

The product of the customization has been included in a reduced version of PPO, carried out for this example. The inclusion of the customized ontology in current and complet PPO could be carried out in the near future.

7.4. Proposed Guidelines for Reusing Domain Ontologies as a Whole

The goal of domain ontology reuse is to find and select one or several domain ontologies related with the domain of the ontology being developed in order to be used in such ontology in development. The output is a set of whole domain ontologies.

Table 21 shows the filling card for the domain ontology reuse process, including the definition, goal, inputs and outputs, who carries out the process and when the process should be carried out.

Domain Ontology Reuse	
<i>Definition</i>	
<p><i>Domain Ontology Reuse</i> refers to the process of using domain ontologies in the solution of different problems.</p>	
<i>Goal</i>	
<p>The goal of this process is to find and select one or several domain ontologies related with the domain of the ontology being developed in order to be used in such ontology in development.</p>	
<i>Input</i>	<i>Output</i>
<p>The OSRD.</p>	<p>Ontology network extended with the reused domain ontology.</p>
<i>Who</i>	
<p>Software developers and ontology practitioners.</p>	
<i>When</i>	
<p>The domain ontology reuse process should be carried out after the ontology specification activity.</p>	

Table 21. Domain Ontology Reuse Filling Card

The activities for performing the domain ontology reuse process are explained in detail in the following and can be seen in Figure 35. In this section, we describe the proposed guidelines at activity level, including input and output information. The level of detail provided by the activities is sufficient for explaining the guidelines, and for this reason, it was not deemed necessary to break activities down into smaller unit of works, that is, tasks.

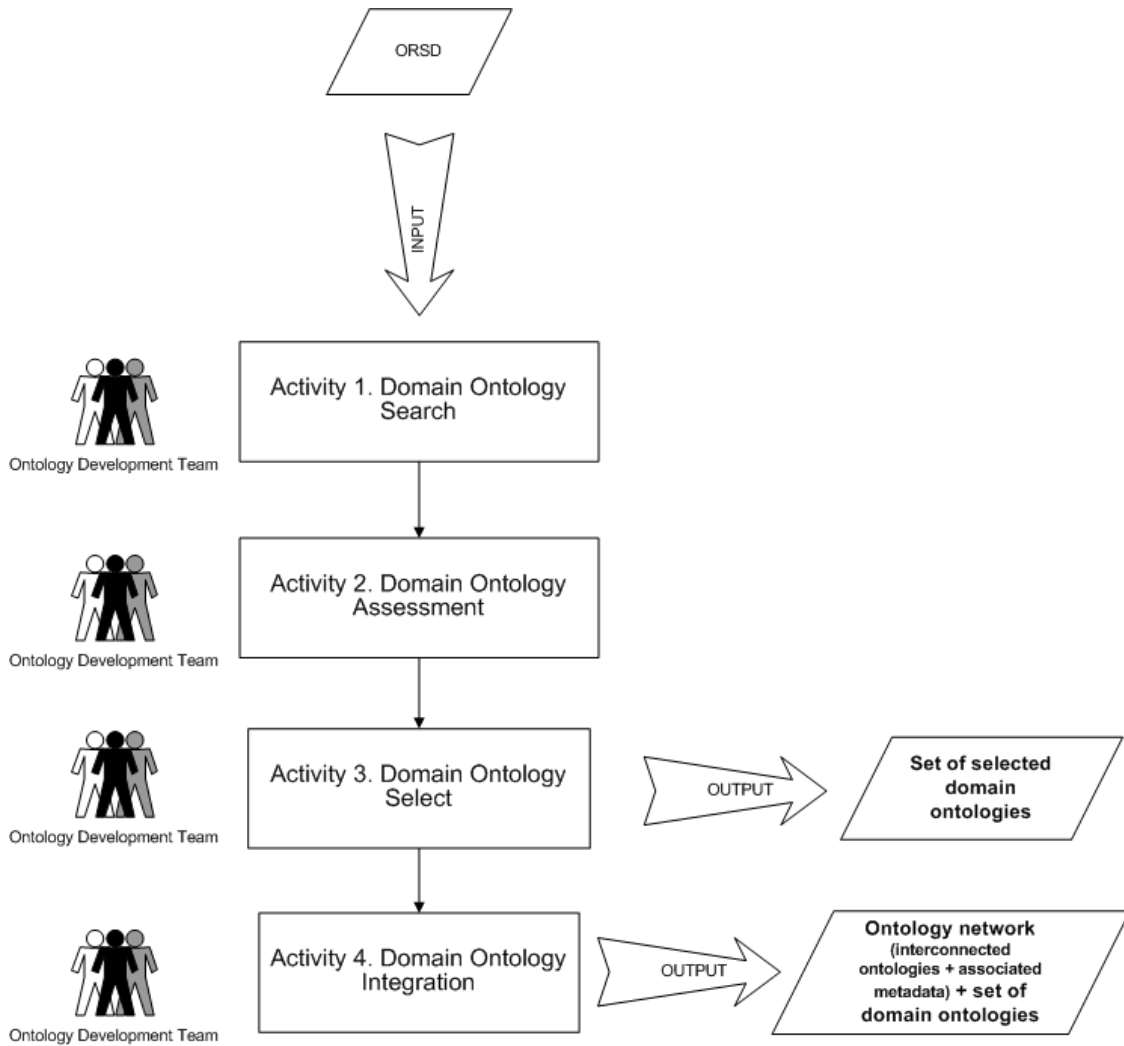


Figure 35. Activities for Reusing Domain Ontologies as a Whole

Activity 1. Domain Ontology Search.

The objective of this activity is to search in libraries, repositories and registries for candidate domain ontologies that could satisfy the needs of the ontology network being developed. The ontology development team carries out this activity taking as input the ORSD, concretely those terms that have a high frequency in the ORSD, using tools as Oyster, Swoogle, etc.

The activity output is a set of candidate domain ontologies that could be implemented in different languages.

Activity 2. Domain Ontology Assessment.

The objective of this activity is to find out if the set of candidate domain ontologies are useful for the development of the ontology network. The ontology development team carries out this activity taking as input the set of domain ontologies obtained in activity 1, using the following criteria for deciding if a particular domain ontology is useful or not.

- ❑ Check if the scope and purpose established in the ORSD are similar to those of the candidate domain ontologies.
- ❑ Check functional ontology requirements established in the ORSD. Examples of requirements can be: the language for implementing the ontology is required to be a particular one (syntactic

level), terms to be used in the ontology must be taken from standards, multilinguality must be represented in the ontology to be developed, etc.

- Checking the CQs included in the ORSD with respect to the candidate domain ontologies, taking into account the following levels:
 - Terminological Level: the ontology development team calculates the precision and recall of the candidate domain ontologies with respect to the terminology included in CQs. Precision and recall (or coverage) have been defined in section 6.5.
 - Semantic Level: the ontology development team checks if the candidate domain ontologies are able to answer the CQs included in the ORSD.

The activity output is an assessment table analysing each candidate domain ontology with respect to the aforementioned criteria. In such an assessment table, useful domain ontologies are shadowed. For deciding that a domain ontology is useful, the set of criteria related with the ontology requirements and CQs has to be satisfied.

Table 22 shows a hypothetical example of an assessment table with three ontologies about publications. The example considers that in the OSRD is established the following: OWL-DL is required as language for implementing the ontology, standard terminology is not needed in the ontology to be developed, and multilinguality is needed in the ontology. Table 22 shows two shadowed columns (*Publication Ontology 1* and *Publication Ontology 3*) for the ontologies considered useful. The other column is not shadowed because *Publication Ontology 2* does not satisfy two criteria related with the ontology requirements (ontology purpose and ontology language).

		Publication Ontology 1	Publication Ontology 2	Publication Ontology 3
<i>Ontology Requirements</i>				
	From the ORSD			
O. Scope	<i>Build an ontology about publications</i>	Build and ontology about publications	Build and ontology about publications	Build and ontology about publications
O. Purpose	<i>Publications related to European Research Projects</i>	Publications related to European Research Projects	Publications related to Education	Publications related to European Research Projects
Syntactic Level: O. Language	<i>OWL-DL</i>	OWL-DL	KIF	OWL-DL
Standards	<i>Not needed</i>	--	--	--
Multilinguality	<i>Yes</i>	Yes	Yes	Yes
<i>Checking CQs</i>				
Terminological Level: Precision		80 %	90 %	80 %
Terminological Level: Recall		100 %	90 %	100 %
Semantic Level: Answering CQs		Completely	Completely	Completely

Table 22. Hypothetical Example of Domain Ontology Assessment Table

Activity 3. Domain Ontology Select.

The objective of this activity is to find out which domain ontologies are the most suitable for the development of the ontology network. The ontology development team carries out this activity taking as input the useful domain ontologies from the assessment table obtained in activity 2, using the following criteria for selecting the most suitable domain ontologies. These criteria are explained in detail in section 7.3.

- ❑ **Ontological Resource Understandability:** check if the domain ontology has good documentation.
- ❑ **Ontological Resource Modularization Effort:** check if the domain ontology is good modularized.
- ❑ **Ontological Resource Integration Effort:** check if the estimation effort for integrating the domain ontology is low and if the domain ontology used naming conventions..

Ontological Resource Reliability: check if the domain ontology is reused by others ontologies or others ontology-based projects and if the ontology have been evaluated. The ontologies that satisfy the larger number of criteria are selected in the selection table by means of shadowed columns. The activity output is a set of domain ontologies selected from the selection table.

Following with the example from Table 22, we can see in Table 23 how the two useful domain ontologies are analysed with respect to the aforementioned criteria, and *Publication Ontology 1* is selected. Guidelines on how to establish the different values for the different criteria will be included in the next version of this deliverable.

	Publication Ontology 1	Publication Ontology 3
<i>Ontological Resource Understandability</i>		
Documentation	Good	Good
<i>Ontological Resource Modularization Effort</i>		
Modularized	Yes	Yes
<i>Ontological Resource Integration Effort</i>		
Estimated Integration Effort	Low	Low
Naming Conventions	Yes	No
<i>Ontological Resource Reliability</i>		
Reused by Others	Yes	Yes
Evaluated	Yes	Yes

Table 23. Hypothetical Example of Domain Ontology Selection Table

Activity 4. Domain Ontology Integration.

The objective of this activity is to integrate the selected domain ontologies in the ontology network being developed. The ontology development team carries out this activity taking as input the set of domain ontologies selected in the selection table obtained in activity 3. For each domain ontology

included in the input set, the ontology development team decides one of the following three modes for integrating:

- The selected domain ontology is reused as they are. The ontology development team integrates the domain ontology in the ontology network being developed.
- The selected domain ontology is reused with significant changes (e.g., use the domain ontology in a different implementation language). In this case, the ontological resource reengineering activity should be carried out with the selected domain ontology. Thus, scenario 4 (Figure 8 from section 4.1) should be followed.
- There are several ontologies in the same domain that are merged to obtain a new domain ontology. In this case, scenario 5 or scenario 6 (Figure 8 from section 4.1) should be followed.

Before reusing the selected domain ontologies by following any reuse mode, it is also convenient to evaluate the domain ontologies through the *ontology evaluation activity* [111].

The activity output is an ontology network including the set of selected domain ontologies.

7.5. Proposed Guidelines for Reusing Ontology Statements

The goal of the ontology statement reuse is to make use of ontology statements in the ontology network being developed. The output of this process is a set of ontology statements to be used in the ontology network being developed.

Table 24 shows the filling card for the ontology statement reuse process, including the definition, goal, inputs and outputs, who carries out the process and when the process should be carried out.

The ontology statement reuse process can be applied in two different situations:

Situation 1. Building ontology networks from scratch. In this scenario, it is much more useful if a preliminary model for the ontology is already established. It is also useful to have a clear idea of what the ontology requirements are, as well as a good understanding of the ontology domain and the ontology purpose, etc.

Situation 2. Extending or improving existing ontology networks.

It is important to mention that ontology statements can serve not only for reuse itself but also for ontology design and for helping in the domain understanding.

Ontology Statements Reuse	
<i>Definition</i>	
<p><i>Ontology Statement Reuse</i> refers to the process of using ontology statements (from domain ontologies) in the solution of different problems.</p>	
<i>Goal</i>	
<p>The goal of this process is to make use of ontology statements from an ontology that was not originally designed for the task at hand.</p>	
<i>Input</i>	<i>Output</i>
<p>The OSRD and available ontology statements (in the same or similar domain that the ontology network being developed).</p>	<p>Ontology network extended with reused ontology statements.</p>
<i>Who</i>	
<p>Software developers and ontology practitioners.</p>	
<i>When</i>	
<p>Ontology statement reuse can be performed in various stages of the ontology life cycle. Most naturally reuse is performed at the stage of building the ontology and it can be helpful in a variety of situations, whether the ontology is built from scratch or extended from an initial ontology. Reuse can also appear at later stages of the life cycle when the ontology is updated and/or extended to cover new knowledge.</p>	

Table 24. Ontology Statement Reuse Filling Card

The activities for carrying out the ontology statement reuse process are explained in detail in the following and they can be seen in Figure 36. In this section, we describe the proposed guidelines at activity level, including input and output information. The level of detail provided by the activities is sufficient for explaining the guidelines, and for this reason, it was not deemed necessary to divide the provided activities into smaller unit of works, that is, tasks.

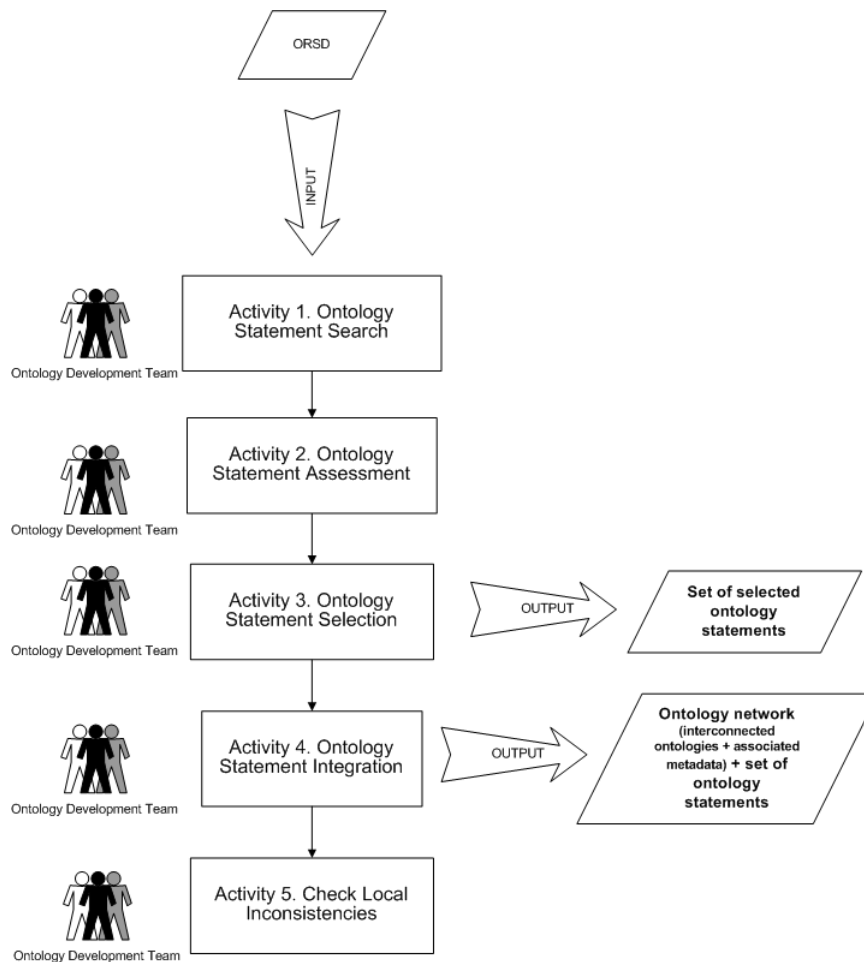


Figure 36. Activities for the Ontology Statement Reuse

Activity 1. Ontology Statement Search.

The goal of this activity is to search the internet for candidate ontology statements that could satisfy the ontological needs in a particular case. The ontology development team carries out this activity taking as input the ORSD, concretely those terms that have a high frequency in the ORSD, and using existing gateways to the Semantic Web, such as WATSON⁴⁶.

The activity output is a set of ontology statements that could be implemented in different languages.

Activity 2. Ontology Statement Assessment.

The goal of this activity is to decide if a concrete ontology statement is useful or not for the ontology network being developed. The ontology development team carries out this activity taking as input the set of ontology statements obtained in activity 1. The ontology development team must inspect the content and granularity of ontology statements to assess whether they satisfy its needs or not.

Given the heterogeneity of online available ontologies from a quality perspective, the assessment activity is not trivial and it can be a major issue for practitioners that are not ontology engineers.

⁴⁶ <http://watson.kmi.open.ac.uk/WatsonWUI/>

Therefore, to support them, we provide a set of criteria for assessing each ontology statement. These criteria have been identified in hands-on experiments during which three ontology engineers used the Watson plugin to build and extend ontologies by reusing statements provided by online ontologies. Such experiments are detailed in section 7.6.1. We regard these identified criteria as initial and wish to refine and extend them in the future based both on further experimentation and on practitioner's feedback.

- ❑ Check if the ontology statement belongs to an ontology with the same or similar scope to the ontology network being developed.

For example, if the ontology statement is contained in an ontology about "Sport Events" and the ontology network to be developed is about "European Project or Research Project", then the ontology statement is not useful for the ontology network's purpose.

- ❑ Check the purpose of the ontology statements found and the purpose of the ontology network being developed to know if they are similar or not.

For example, the statements "Meeting is subclass of SocialInteraction", "Boiling is subclass of StateChange" can be true, but they may not be useful for the purpose of the ontology being developed.

- ❑ Check the clarity of the ontology statement.

For example, ambiguous statements like "Book subclass of '_anon699'", "Pan subclass of T" are not useful by themselves without additional information about the original ontology (e.g., the ontology itself, documentation, etc) which could help to clarify their meaning. Such statements should not be reused.

- ❑ Check the information content of the statement.

In some cases, the retrieved statements provide little additional information, for example by linking a concept to an abstract root concept (e.g., "Publication is subclass of: 'Root', 'Object', 'Thing', 'DEF_ROOT_CONCEPT', 'Resource'"), or by declaring that a concept is equivalent with itself ("Publication is equivalent to: 'Publication'"). Such statements should not be reused either.

- ❑ Assess the correctness of the statement from a (formal) modeling perspective.

- ❑ Check that the naming of concepts in the ontology statement reflect the intended meaning of the statement given its ontological context.

For example, "Publication is subclass of Event" is not correct, because the name 'Publication' does not reflect the intended meaning of the statement, which, given its context in the ontology, was that of 'publication = publishingEvent'. When such statements are reused it is important to rename their concepts in a way that they clearly reflect the meaning of the statement.

More examples of this kind of error are: "Book is subclass of Reference" (in this case, 'book = bookReference'); "Book is subclass of Image" (book = bookImage); "Journal is subclass of Paper" (journal = journalPaper); "Conference is subclass of Pear" (conference = conferencePear).

- ❑ Check if the ontology statement is not invalid from a formal perspective, e.g., by confusing "subclassOf" relations with other relations such as "partOf" or "relatedTo" relations.

For example, “Chapter subclass of Book”: in this case the relation ‘subclassOf’ is incorrectly used to model partonomy. “Article is subclass of Journal” is an example where a relatedness relation (e.g., published in) is modelled through subsumption. These types of modelling errors are fairly common in online ontologies and the person performing the reuse should avoid introducing such errors by reuse. One way to avoid this kind of statements should be to ask (search in internet) if “a chapter is a book.

The activity output is a set of ontology statements useful for the ontology network being developed.

Activity 3. Ontology Statement Selection.

The goal of this activity is to decide between the useful ontology statements which ones are the best or most convenient for the ontology network being developed. The ontology development team carries out this activity taking as input the set of useful ontology statements obtained in activity 2, using the following criterion for comparing ontology statements and selecting the most convenient ones.

- ❑ Minimum effort needed for integrating the ontology statement in the ontology network being developed. For example, if several ontology statements are valid, but they use different naming conventions, then (if possible) reuse the one using the same naming convention as yours to avoid adapting the statement to your ontology network.

The activity output is the set of ontology statements that is the most appropriate for their ontology network requirements.

Activity 4. Ontology Statement Integration.

The goal of this activity is to decide how the selected ontology statements will be integrated in the ontology network being developed. The ontology development team carries out this activity taking as input the selected ontology statements obtained in activity 3, using any of the following three integration modes:

- ❑ The selected ontology statements will be reused as they are.
- ❑ The selected ontology statements will be reengineered.
- ❑ The selected ontology statements will be merged.

Apart from these integration modes, the ontology development team has also to decide between importing the ontology statements, copying the ontology statements or establishing mappings with the ontology statements.

Detailed guidelines for taking both aforementioned decisions will be included in next version of this deliverable.

The activity output is an ontology network including the set of selected ontology statements.

After integrating an ontology statement, the following work will be probably done:

- Changing names (concepts, properties) to adapt them to the naming conventions used in the ontology network being developed.
- Adding range in properties; and changing cardinalities.
- Adding restrictions.

Activity 5. Check Local Inconsistencies.

The goal of this activity is to check for local inconsistencies in the ontology network. Such inconsistencies could be introduced by adding new knowledge to the ontology network.

The ontology development team carries out this activity taking as input the ontology network including the set of selected ontology statements obtained in activity 4.

The activity output is an ontology network, including the set of selected ontology statements, without inconsistencies.

7.6.1. Experiments on Ontology Statement Reuse

In this section we will exemplify the reuse of ontology statements. Guidelines described in section 7.6 have been inspired by our hands-on experiments in using the NeOn Watson plugin in two use-cases:

- 1) to extend two existing ontologies in the European Research Project domain.
- 2) to build an ontology about Paella from scratch.

The description of these proposed use-cases is detailed in Annex A.

The experiment was performed by three ontology engineers, and their experiences were discussed and compared. This is a summary of their final conclusions.

The ontology extension activity was easy to perform because the European Research Project domain is well-covered by online-ontologies. When reusing statements in this case, the following situations were observed:

- Some statements that were reused brought a new insight into the domain that could lead to changes in the structure of the base ontology. For example, one ontology returns Thesis < Publication, however in the base ontology Publication already exists but not as a parent of Thesis.
- There were some cases when different ontologies returned contradictory bits of knowledge: in some ontologies Deliverable < Report, in others Report < Deliverable. In these cases, it was up to the engineer to decide the standpoint (s)he agreed on.
- Some ontologies were incorrect from a formal modelling perspective, e.g., Chapter < Book.
- In some cases, online ontologies provided information about a different sense for a concept: when searching for Article (as a publication), some ontologies returned information about it in the context of language grammar (i.e., as a determiner). In this and other cases, it can be difficult to judge the meaning of certain recommended statements without being able to interpret the broader ontological context. As a result, there were sometimes (possibly) relevant concepts/relations which were not reused simply because the engineers did not understand what they meant.

The second case study focused on building an ontology from scratch. The general experience in this case was that a sound ontology is impossible to be built purely by reusing ontology statements. The experimenters who wished to rely exclusively on reused knowledge experienced a phenomenon of being “drifted away” in their modelling by the knowledge presented by the plugin. The resulting ontology was unfocused and unclear. As a result, in a second round, the experiment was repeated with this in mind: experimenters first built a skeleton for the ontology specifying the main concepts that they wished to describe (e.g., kitchen equipment, food ingredients) and worked within the boundaries set by this frame. This approach was experienced as much more efficient. When reusing statements, engineers experienced situations similar to those described in the case of ontology extension (use-case 1).

8. Ontology Design Patterns Reuse

This chapter describes the reuse of Ontology Design Patterns (ODPs), identified in scenario 7 of this deliverable (Figure 8, in section 4.1). As in previous chapters, the goal is to define guidelines for the performance of the Ontology Design Patterns reuse in the framework of the construction of ontology networks. With this aim, we start by referring to previous deliverables in NeOn that have already dealt with ODPs in section 8.1.

Then, we include a brief state of the art (section 8.2) on Ontology Design Patterns reuse with three subsections about existing methods, techniques and tools. In this section, we relate ODPs reuse to the reuse of design patterns in the neighbouring field of Software Engineering, since the latter has a longer tradition in the patterns reuse and allows us to find interesting parallelisms. This brief analysis shows important limitations in the reuse of design patterns that we will try to alleviate in the ontological field. Therefore, our goal is to propose methods and guidelines for the reuse of ODPs. In the present deliverable, methods and guidelines are intended for a specific kind of users, namely, *naive users*. By *naive users* we understand software developers and ontology practitioners with little expertise in the ontology development and insufficient command of ontology languages (OWL, RDF(S), etc.), ODPs, etc. In next versions of this deliverable we aim at proposing methods and guidelines in the reuse of ODPs for *expert users*, i.e. software developers and ontology practitioners with expertise in the modelling of ontologies and reuse of design patterns. In section 8.3, the main techniques and a novel tool for supporting the method intended for *naive users* are presented. Finally, the last section (section 8.4) is devoted to the proposal of a set of methodological guidelines for the reuse of ODPs in NeOn.

8.1. Introduction

As we already mentioned, our objective is to propose methods and guidelines for the reuse of Ontology Design Patterns (ODPs henceforth) during the ontology development. Two previous deliverables in NeOn have already dealt with ODPs: D5.1.1 [110], and D2.5.1 [95].

The first one deals with the identification of the modelling components to be used for modelling networked ontologies, and with the creation templates for describing them. The first version of this deliverable mainly focuses on those design patterns that are based on OWL. These patterns have been divided into three different types: *logical patterns*, *architectural patterns* and *content patterns*.

D2.5.1, more centred on identifying methods, techniques and tools for assisting ontology practitioners, is completely devoted to Ontology Design Patterns, which are here classified into five main groups: *Structural ODPs*, which can be *Logical* or *Architectural ODPs*, *Correspondence ODPs*, which can be subdivided as well in *Reengineering* or *Mapping ODPs*, *Content ODPs*, *Reasoning ODPs*, and *Presentation ODPs*. Section 2.2 of D2.5.1 contains definitions of each type of pattern, but the rest of the document is dedicated to the development and usage of Content ODPs.

In both deliverables, ODPs have been described following a specific template, designed for this purpose within NeOn, which provides a complete and clear description of the pattern.

The mentioned template contains the following sections:

- ❑ *general information* of the pattern as its name and type
- ❑ a description and an example in natural language of the *modelling problem* addressed by the pattern
- ❑ a description in natural language of the *solution provided by the pattern*

- ❑ a graphical *representation* of the pattern in NeOn UML and a *formalization* in terms of the NeOn OWL Ontology Metamodel
- ❑ relations of the pattern to other modelling components and remarks in natural language clarifying the use of it.

8.2. State of the Art

The term *design pattern* was introduced in the seventies by Christopher Alexander in the architecture domain, as explained in Bushmann *et al.* [25], for designating those modelling solutions that after being recurrently used for solving similar design problems, could be identified as generalized design solutions to be applied whenever a similar problem appeared. In the own words of Alexander⁴⁷, patterns described solutions "in such a way that you can use this solution a million times over, without ever doing it the same way twice". In the mid 1980s, Cunningham and Beck adapted Alexander ideas to software development (as reported in 25), but it was not until the publication of the *Design Patterns - Elements of Reusable Object-Oriented Software* [51] book by the so-called *Gang of Four* (Gamma, Helm, Johnson and Vlissides) that design patterns became broadly used in the object-oriented software design. Since then, design patterns have been applied in a great variety of areas within Computer Science, as for example, Hypermedia and Web applications⁴⁸, E-learning⁴⁹ or User Interface design⁵⁰, among others. However, it is not until the beginning of the 21st century that design patterns are fully introduced in the Ontology Engineering domain by ontology experts as Gangemi and colleagues [52], Rector and Rogers [97], Svatek [113] or the W3C Consortium⁵¹.

Benefits of design patterns in Software Engineering are well known. These can be summarized in three points, as in [92]:

- ❑ design patterns allow less experienced users to produce a better design
- ❑ design patterns "encourage recording and reusing best practices even for experienced designers"
- ❑ design patterns can improve communication by defining a common design terminology

Design pattern reuse in object-oriented design is an extended practice, well supported by design pattern repositories and manuals as the one by Gamma *et al.* [51], or the above mentioned by Buschmann *et al.* [25]. Nowadays, those pattern repositories are integrated in software tools in order to allow a quicker access and integration of patterns. However true that might be, most of the existent manuals or repositories presuppose prior design knowledge and expertise. This fact and other limitations of the reuse of patterns are being recently discussed in public forums by some experts in the Software Engineering domain [43] (see also *The Software Patterns Blog*⁵²). The main limitations are related to the lack of general methodologies or standards for the reuse of the different pattern repositories, since some efforts in that sense are limited to steps or recommendations for local use developed by the authors of the manuals themselves. In the same sense, templates follow different styles depending on the manual, so that some of the steps or approaches given by certain authors cannot be extrapolated or reused in searching other design pattern repositories. Finally, an additional limitation reported by practitioners is related to the efforts

⁴⁷ <http://c2.com/cgi/wiki?CategoryPattern>

⁴⁸ <http://www.designpattern.lu.unisi.ch/index.htm>

⁴⁹ http://www2.tisip.no/E-LEN/patterns_info.php

⁵⁰ <http://www.deyalexander.com/resources/uxd/design-patterns.html>

⁵¹ <http://www.w3.org/2001/sw/BestPractices/>

⁵² <http://www.pattern.ijop.org>

that the search activity requires, which apart from being time consuming, demands a careful analysis of the templates on the part of the user.

Limitations in the reuse of design pattern repositories in Software Engineering can be summarized as follows:

- ❑ Assumption of users prior knowledge about design patterns reuse
- ❑ Inexistence of generalized methodologies for design patterns reuse
- ❑ Inexistence of standardised design pattern templates
- ❑ Inexistence of techniques and tools for supporting the design pattern selection
- ❑ Time consuming task for users without prior knowledge

Regarding the reuse of design patterns in Ontology Engineering, this practice is not so widespread because of two obvious reasons: the incipient stage in the ODPs research, and the almost inexistence of ODPs repositories. Currently, we find some ODPs on-line repositories, as the one focused on the Biological domain⁵³, or the preliminary repository of OWL-based Content OPs⁵⁴ at the Laboratory for Applied Ontology wiki page. The latter repository is being extended and enhanced within the NeOn project and is expected to be available in 2008 at the Ontology Design Patterns wiki page⁵⁵.

As in the case of object-oriented design patterns, there exist no methodologies *per se* for guiding users and facilitating the reuse of ODPs. It is as well assumed that users have some expertise in the reuse of object-oriented design patterns and know that they have to access design pattern repositories and search for the appropriate pattern. There have been, however, some initiatives for helping users in the process of adapting or implementing ODPs by means of wizards, as for example, the ones provided by the CO-ODE project⁵⁶ for the Protégé ontology editor⁵⁷, as reported in [39].

Nevertheless, if we want to bring ontologies closer to the average user that not necessarily has expertise in design pattern reuse in general, some effort has to be put into the following actions:

- ❑ Creation of standardized templates for the description of ODPs understandable to different types of users
- ❑ Creation of generalized methods or guidelines for users *with* and *without* previous experience in the ODPs reuse
- ❑ Creation of techniques and tools for supporting a semi-automatic or automatic pattern selection

In this deliverable we focus on the creation of methods and guidelines for the reuse of ODPs directed to *naive users* or users with little expertise in the development of ontologies. Moreover, we also describe a novel technique for a semi-automatic reuse of ODPs, and the tool that supports it. In the next sections we try to give a brief overview on available methods (8.2.1), techniques (8.2.2) and tools (8.2.3) for the reuse of design patterns. In the last point of this section (8.2.4) we summarize the main conclusion from this state of the art.

⁵³ <http://odps.sourceforge.net/odp/html/index.html>

⁵⁴ <http://wiki.loa-cnr.it/index.php/LoaWiki:CPRRepository>

⁵⁵ <http://www.ontologydesignpatterns.org>

⁵⁶ <http://www.co-ode.org/downloads/wizard/>

⁵⁷ <http://protege.stanford.edu/>

8.2.1. Methods

Methods followed for reusing design patterns in Software Engineering are limited to some advices or recommendations in the use of design pattern repositories. Expert users access repositories that collect patterns, and by means of analogy criteria based on their experience, and relying on the descriptions included in the catalogues, they select the most adequate design pattern for their needs.

In this sense, Gamma *et al.* [51] devoted two sections in the introduction to their book to the *selection* and *use* of design patterns. The first section gives some recommendations to users accessing the book for the first time. The aim is to guide users through the different book chapters, in order for them to take advantage of the contained information and finally make the pattern selection. The second section, the one about *use*, offers users a “step-to-step approach” to applying patterns effectively, that can be summarized in a close analysis to the different sections of the template designed for the pattern description. Buschmann *et al.* [25] also dedicate a chapter to the *pattern selection* based on the templates they propose and by means of some examples. The authors recommend the user to carefully “specify the problem”, so that he exactly knows what he needs, and select the pattern that better satisfies his needs, but they do not propose any guides about how to achieve that.

In Ontology Engineering, the scenario is quite similar. Search, selection and application activities are taken for granted. Regarding the selection activity, in D2.5.1 [95] authors propose to apply the “typical procedures for ontology selection (...)” to the selection of ODPs, but they do not say how the ontology selection method can be employed, or which techniques or tools would be needed.

As far as the adaptation activity of ODPs to real use cases is concerned, D2.5.1 offers some guidelines for matching Content ODPs to real use cases. These matching possibilities are: *precise* matching, *broader* matching, *narrower* matching, *partial* matching and *redundant* matching. In any case, it is assumed that the matching activity is performed manually, which means that no techniques or tools are foreseen for supporting it.

8.2.2. Techniques

At the present stage of this research, the identified techniques in the process of ODPs reuse are aimed at helping users in the process of adapting or implementing ODPs while developing ontologies by means of wizards, as is the case of the CO-ODE project⁵⁸. CO-ODE wizards have been designed for the ontology editor Protégé, as mentioned in section 8.2., and help users to reuse OWL-based patterns.

8.2.3. Tools

Regarding tools for the reuse of ODPs, we can refer to the storage system of catalogues that contain ODPs. The few catalogues containing ODPs are stored in web pages (as in the case of the catalogue being developed in the Gene Ontology Next Generation (GONG) project⁵⁹, or the one from the Laboratory for Applied Ontologies⁶⁰), or integrated in ontology editors (as it is expected to be the case of the NeOn toolkit), in which cases the benefit from this storage system is a faster access to design patterns for being reused.

⁵⁸ <http://www.co-ode.org/downloads/wizard/>

⁵⁹ <http://www.gong.manchester.ac.uk/>

⁶⁰ <http://wiki.loa-cnr.it/index.php/LoaWiki:CPRRepository>

8.2.4. Conclusion

After having analyzed the state of the art on ODPs reuse, we can conclude that there are no methodologies or guidelines *per se* for the reuse of ODPs. Target users are supposed to be aware of the existence of ODPs catalogues, as an analogy from Software Engineering, and to be able to apply them for their needs. Therefore, users are assumed to be experts in the ODPs reuse, either by having collected some expertise in the object-oriented design, or by being experts in ontology development.

However, we consider that the expertise of users cannot always be taken for granted, and that techniques and tools for supporting the reuse of ODPs are necessary for helping users in selecting ODPs. As reported in section 8.2.2, some initiatives in this sense are already taking place, and a library of Content ODPs as well as some recommendations for their manual adaptation, will be available in 2008 in the framework of the NeOn project. Nevertheless, some additional efforts need to be made for improving the reuse of ODPs, and for this reason we have decided to focus on these two research issues:

- Development of methods and guidelines for the reuse of ODPs
- Creation of techniques and tools for supporting those methods with the aim of enabling an easier and practical reuse of ODPs

8.3. NeOn Method for the Reuse of Ontology Design Patterns by Naive Users

One objective of the NeOn project is to expand the use of ontologies among a wider community of users, especially novice users. For this purpose, methods intended for users with little expertise in the development of ontologies, and in its turn, in the reuse of Ontology Design Patterns (ODPs), have to be supported by user-friendly tools. This implies in most cases the inclusion of some Natural Language (NL) components that make the interaction of the non expert user with the machine easier. As far as the ODPs reuse is concerned, the inclusion of NL components serves as interface between naive users and ODPs, and is a crucial issue for bringing ontologies closer to the novel user.

In this sense, we propose a novel method for the reuse of ODPs that has as starting point an expression in NL of the phenomenon or domain parcel the user wants to model, and ends up with the obtainment of the adequate ODP. We assume that the user has a good command of the domain (s)he wants to model, and that the information expressing the modelling aspect in NL is *correct* from the content viewpoint⁶¹. Assumptions about users introducing wrong information from the point of view of the domain content are left out of the scope of this research for the time being. It also is important to observe, that the proposed method allows the user to *freely introduce a sentence in NL*, without any kind of restrictions regarding the use of controlled languages⁶² or vocabularies.

The NeOn method can be divided in three main steps:

1. Formulation in NL of the domain aspect to be modelled
2. Selection of the ODP that better matches the expression in NL
3. Integration of the ODP into the ontology with the information extracted from the NL expression

⁶¹ If the user introduces a sentence as *Animals are divided into vertebrates and omnivores*, the system will take it as right.

⁶² Controlled Natural Languages are subsets of natural languages whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity. Traditionally, controlled languages fall into two major categories: those that improve readability for human readers, particularly non-native speakers, and those that improve computational processing of the text. (Available at: <http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages/> [31/03/08])

In order to support this method, we proposed the development of a tool for enabling an automatic or semi-automatic selection of ODPs, mainly intended for users with little expertise, but also recommendable for expert users, which relies in the application of NL techniques for performing a semi-automatic selection. Current research has been centred on Logic ODPs ODPs [see 110 and 95], as explained in section 8.3.1.

In the next sections, we include a description of the techniques and tools that support this method.

8.3.1. Enrichment of NeOn Ontology Design Patterns with Lexico-Syntactic Patterns

With the aim of supporting the proposed method with NL techniques and tools for the reuse of ODPs, we enriched the ODPs created in NeOn (cf. D5.1.1 [110] and D2.5.1 [95]) with a NL field. The NL field consists of Lexico-Syntactic Patterns⁶³ that exactly match the relation of interest expressed by the ODP, in the sense that a 100% correspondence between the ODP and the natural language expression or linguistic construct is established.

The term Lexico-Syntactic pattern (LSPs from now on) was first introduced by Hearst [73] in Computation in the early 1990s. The goal of her research was the automatic acquisition of lexical syntax and semantics for building lexicons. LSPs identified by Hearst had the following characteristics: (1) all expressed a hyperonymy-hyponymy relation, (2) were directly extracted from texts, and (3) had as main elements prepositional phrases, paralinguistic signs or conjunctions (not verbs). Examples of Hearst patterns are shown in Table 25.

<p><i>NP⁶⁴ such as {NP1, NP2... (and or) NPn</i></p> <p><i>NP {,NP}* {,}</i> or other NP</p> <p><i>NP {,} especially {NP,}* { or and } NP</i></p>
--

Table 25. Examples of Hearst Patterns

Since then, there have been many authors that have applied Hearst LSPs for the automatic discovery of lexical items. In Ontology Engineering, the LSP identification has been aimed at extracting related concepts or instances in an automatic or semi-automatic way for ontology population. For this end, authors as Snow [107], or Cimiano [31] have extended the original set of Hearst patterns with additional patterns expressing the hypernym-hyponym relation, or new ones expressing the relations of meronymy, agency, cause, etc. Some patterns were similar to Hearst ones, that is, *not verb oriented*, others had verbs as main elements. Other research works have used patterns for finding out how concepts are related, as in Haase and Völker [71], Sanchez Ruenes [99], Marshman [82] or Feliu [44], among others. *However, no research has been oriented to obtain Lexico-Syntactic patterns equivalent to the relations expressed in Ontology Design Patterns, which is one of the main objectives of our proposal.*

A remarkable difference between the original Hearst patterns and the LSPs we propose here is that ours are *verb oriented*, i.e., we focus our research on the identification of patterns that express a relation of interest by means of verbs, which are normally the ones that carry the semantic of the relation. The main reason for that is our assumption that for modelling categories of the world in ontologies the usual way for describing them and expressing how they are related is by means of verbs in affirmative or declarative sentences in the simple present tense. For example: *Animals are*

⁶³ Linguistic constructions that express a conceptual relation are known in literature as Lexico-Syntactic Patterns, and they are said to “occur frequently and in many text genres, almost always indicate a relation of interest, and be recognized with little or no pre-encoded knowledge” [73]. We understand LSPs as *formalized linguistic schemas or constructions derived from recurrent expressions in NL that consist of certain linguistic and paralinguistic elements, following a specific syntactic order, and that permit to extract some conclusions about the meaning they express.*

⁶⁴ NP: Noun Phrase

divided into two major categories: *vertebrates and invertebrates*, would be the usual way of describing domain knowledge. Additionally, it must be taken into account that Hearst patterns were directly identified in texts, in which they were embedded, and typical of the written language, while ours, although being also common of written texts, are expected to appear as independent self-contained declarative statements.

Finally, it is left to say that most of the research on LSPs has been done for the English language. In our case, because of the importance conferred to multilinguality in the NeOn project, we aim at retrieve LSPs in various languages: English, Spanish and German, in a first stage.

Then, for the purpose of enriching the NeOn ODPs template with NL information by means of LSPs, we created a new field to be included in the template designed in NeOn for the description of ODPs, explained in section 8.1. This new field consists of two slots:

1. *Formalization*. This first slot includes the various LSPs as formalizations or abstractions of linguistic constructs that a certain language has for expressing the relation contained in the design pattern. This slot is mandatory.
2. *Examples*. The second row shows some examples of sentences in NL that match the LSPs in question. This slot is optional.

In Table 26 we show the new field included in the NeOn ODPs template, , as already described in D2.5.1 [95].

Slot	Value
General Information	
<i>Name</i>	Name of the component
<i>Identifier</i>	An acronym composed of: component type + component + number
<i>Type of Component</i>	Logical Pattern (LP)
Use Case	
<i>General</i>	Description in natural language of the general problem addressed by the modelling component.
<i>Examples</i>	Description in natural language of some examples for the general problem.
Ontology Design Pattern	
<i>Informal</i>	
Lexico-Syntactic Patterns	
<i>Formalization</i>	NL representations of the ODP in the form of formalized Lexico-syntactic patterns
<i>Examples</i>	NL expressions matching the LSP in question
<i>(UML) Diagram for the General Solution</i>	Graphical representation of the general solution provided, taking into account the UML Profile proposed in [9].
<i>(UML) Diagram for Examples</i>	Graphical representation of the solution provided, using examples and taking into account the UML Profile proposed in [9].
Formalization	
<i>General</i>	Formalization of the pattern in terms of the NeOn OWL Ontology Metamodel [31].
<i>Examples</i>	Formalization of the examples (using abstract syntax for OWL code).
Relationships	
<i>Relations to other modelling components</i>	Description of any relation to other modelling components (use, specialize, etc.).
Comments	
<i>Comments</i>	Remarks for clarifying the use of the modelling component.

Table 26. LSPs Field included in the NeOn ODPs Template

Let us take a closer look at how LSPs are derived from NL, and associated to ODPs. For better understanding this process we will explain it in the light of some examples. We will analyze one of

the most representative relations in ontologies, which is the *SubClassOf* relation, identified as LP-SC-01 in D5.1.1 [110]. This relation is present in the three sentences in NL included below, all of them extracted from the fisheries domains.

- 1) *Animals are divided into two major categories: vertebrates and invertebrates*
- 2) *Fish can be classified into three groups which are: jawless fish, sharks and rays, and bony fish.*
- 3) *Amphibians are divided into: frogs and toads, newts and salamanders, and caecilians.*

These three ways of expressing the *SubClassOf* relation can be generalized or abstracted in one LSP, formalized as follows:

LSP 4: NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>)* and NP<subclass>

For the purpose of formalizing LSPs, the BNF⁶⁵ notation has been used with some extensions. Restricted words and symbols appearing in the LSP exemplified above are collected in Table 27.

SYMBOL	Description
NP	Noun Phrase. It is defined as a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers, and that functions as the subject or object of a verb. We have decided to accompany NPs by the semantic role played by the concept it represents in the conceptual relation in question. In this specific example, NP is followed by <i>superclass and subclass</i> , representing the semantic role of each concept in the relation.
CATV	Categorization Verbs. Set of verbs of classification plus the preposition that normally follows them. Some of the most representative verbs in this group are: <i>classify in/into, comprise in, contain in, compose of, group in/into, divide in/into, fall in/into, belong to</i> , etc.
CD	Cardinal Number
CN	Class Name. Generic names for <i>class</i> usually accompanied by preposition, as <i>class, group, type, member, subclass, category, representative</i> , etc.
PARA	Paralinguistic symbols like <i>colon</i>
()	Parenthesis group two or more elements
*	Asterisk indicates repetition
[]	Elements in brackets are meant to be optional, which means that they can be present either at that stage or not, and by default of appearance, the pattern remains unmodified.

Table 27. Restricted Words and Symbols in LSPs

The same process performed above for the three sentences in NL has to be repeated for the many expressions in NL that express the conceptual relation *SubClassOf* (LP-SC-01) in a certain language. Finally, all LSPs associated to each ODP are collected and included in the ODP, as can be seen in Table 28. In the examples provided here we just show LSPs for the English language (en). Therefore, if our aim is to have multilingual LSPs, we will have to include the same two slots for the rest of languages, since LSPs are considered to be language dependant and not interchangeable among languages, despite of some of them overlapping, as research in the field has already proven, see [82]. In fact, within NeOn our objective is to include LSPs for English, Spanish and German in the first stage of this research.

⁶⁵ <http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html>

Lexico-Syntactic Patterns (LP-SC-01) (en)		
<i>Formalization</i>	1	NP<subclass> be NP<superclass>
	2	[(NP<subclass>)* and] NP<subclass> be [CN] NP<superclass>
	3	[(NP<subclass>)* and] NP<subclass> (group in into as) (fall into) (belong to) CN NP<superclass>
	4	NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>)* and NP<subclass>
<i>Examples</i>	<p><i>Birds are vertebrate animals</i></p> <p><i>Vertebrates are members of the subphylum Vertebrata</i></p> <p><i>Birds and mammals belong to the group of vertebrates</i></p> <p><i>Amphibians are divided into three groups: frogs and toads, newts and salamanders, and caecilians.</i></p>	

Table 28. LSPs (en) for the *SubClassOf* ODP (LP-SC-01)

Since this is an ongoing research, we include here by way of example, representative cases of LSPs associated to ODPs, some already included in the first version of D2.5.1 [95], others new here for: (1) *DisjointClasses* (LP-Di-01) in Table 29, and *ExhaustiveClasses* (LP-EC-01) in Table 30. As the rest of enriched ODPs included in D2.5.1, these belong to the so-called Logical ODPs. *SubClassOf*, *DisjointClasses* and *ExhaustiveClasses* are patterns related with taxonomical knowledge, and they the ones that have received most attention in this first stage of our research, as well as the *Part-WholeRelation* ODP (CP-PW-01).

Lexico-Syntactic Patterns (LP-Di-01) (en)		
<i>Formalization</i>	1	(NP<class>)* and NP<class> (have NEG) (do NEG have) elements individuals instances in common
	2	(NP<class>)* and NP<class> do NEG share elements individuals instances [CN]

Table 29. LSPs (en) for the *DisjointClasses* ODP (LP-Di-01)

Lexico-Syntactic Patterns		
<i>Formalization</i>	1	(NP<subclass>)* and NP<subclass> be the only [CN] NP<superclass>
	2	Only just (NP<subclass>)* and NP<subclass> be belong to group in into NP<superclass>
<i>Examples</i>	<i>The hagfish and the lamprey are the only representatives of Agnathans.</i>	

Table 30. LSPs for the *ExhaustiveClasses* ODP (LP-EC-01)

With the aim of making LSPs processables by NL processing tools, the identified LSPs have been implemented with JAPE⁶⁶, a component of the GATE architecture.

JAPE stands for Java Annotations Patterns Engine, and is a grammar that consists of a set of phrases, each of which consists of a set of pattern/action rules. The left-hand-side (LHS) of the rules consists of an annotation pattern, and the right-hand-side (RHS) consists of manipulation statements, which can be made up of any Java code.

By way of example, we include in Table 31 the JAPE rule corresponding to one of the LSPs identified for the SubClassOf ODP (LP-SC-01), *LSP 4*:

LSP 4: NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>)* and NP<subclass>

```

Macro: NOUN
(
  ({Token.category == NN} | {Token.category == NNS}
  | {Token.category == NP} | {Token.category == NPS})
)
Rule: SubClassOf
(
  // NP<superclass>
  (NOUN)
  // CATV e.g. be divided|classify into | include | comprise |...
  (({Token.lemma == "be"}
  ({Token.lemma == "divide"} | {Token.lemma == "classify"})
  ({Token.lemma == "in"} | {Token.lemma == "into"})) | (...))
  // [CD] [PUNCT] e.g. two major categories:
  ({Token.category == CD}
  ({Token.category == JJ})*
  (NOUN))?
  ({Token.kind == punctuation})?
  // (NP<subclass>)*
  (({Token.category == JJ})*(NOUN)
  ({Token.kind == punctuation})?)+
  // and NP <subclass>.
  {Token.category == CC}
  ({Token.category == JJ})*
  (NOUN)
  {Token.category == SENT}
):SubClassOf1 -->
:SubClassOf1.SubClassOf = {kind = "SubClassOf "}

```

Table 31. JAPE Rule for LSP 4 of LP-SC-01

Once ODPs have been enriched with LSPs, and JAPE rules have been created for each LSP, we have to make the repository of ODPs accessible to naive users, in order to enable them an automatic or semi-automatic selection of ODPs. For that, we are developing a new NeOn plug-in

⁶⁶ <http://www.gate.ac.uk/sale/tao/#x1-1500007>

for an automatic and semi-automatic selection of ODPs called SOS, *System for Ontology design patterns Support*, and described in section 8.3.2.

8.3.2. SOS NeOn plug-in, *System for Ontology design patterns Support*

The goal of the SOS NeOn plug-in is to select and retrieve the most appropriate ODP, i.e., the one that meets the modelling needs of the user as expressed in the NL sentence introduced as input. The SOS NeOn plug-in workflow, illustrated by Figure 37, has been divided in 5 main steps and is detailed below.

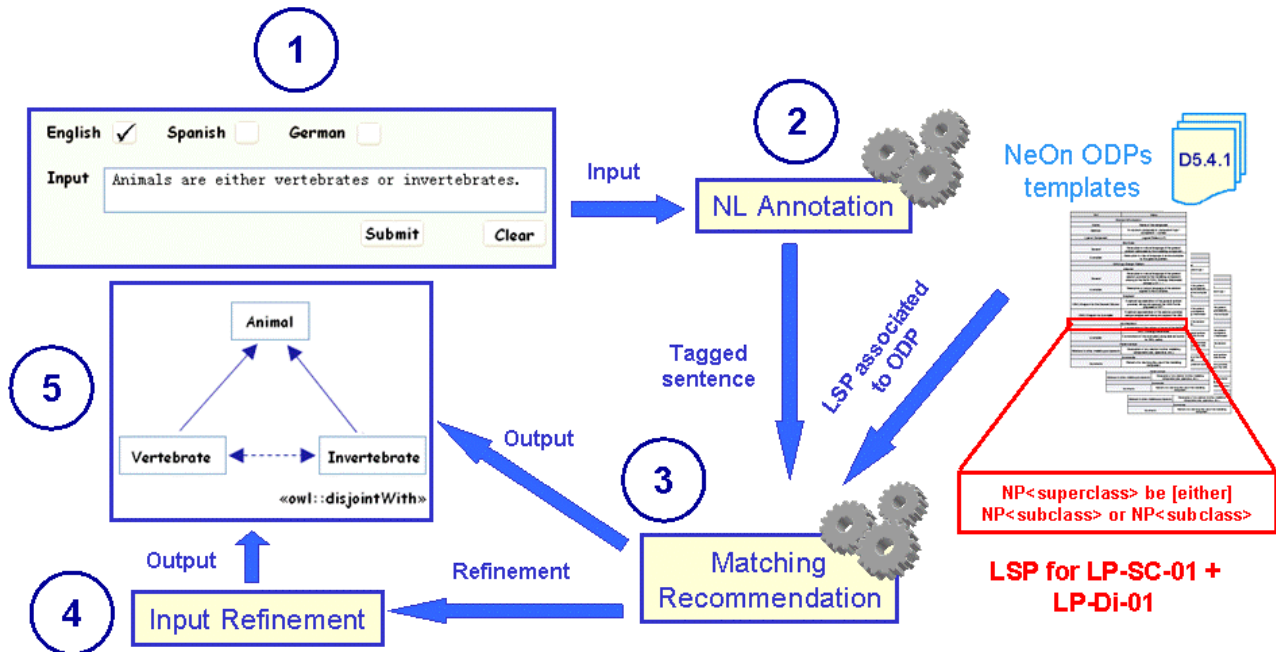


Figure 37. SOS NeOn Plug-in Workflow

Step 1: Input introduction. The user *freely* formulates in NL the phenomenon (s)he wants to model without any constraints from controlled languages or vocabularies, and introduces it in the system. In this sense, our proposal differs from other applications intended to naive users for creating or editing ontologies as CLIE [50] or GINO [18], in which the input has to be compliant with the controlled language the application relies on. For an extended review on such tools see [49].

As already explained at the beginning of this section, we assume the correctness of the statement content.

Step 2: NL tagging. The system analyses the input (a sentence in NL) and annotates it with NL processing tools. For the annotation, some GATE components (General Architecture for Text Engineering⁶⁷) are used, namely, ANNIE information system –which contains a tokeniser, a sentence splitter, a part-of-speech tagger, a gazetteer, and a coreference resolver–, and TreeTagger, an external part-of-speech tagger that supports the annotation for languages as Spanish and German. The result of the annotation is a *tagged sentence*, as illustrated in Table 32, with:

- Information about the syntax by means of the category of words and syntax order
- Lexical information by means of the lemma

⁶⁷ <http://www.gate.ac.uk/>

String	Lemma	Category	Kind
Animals	animal	NNS	word
are	be	VBP	word
divided	divide	VVN	word
into	into	IN	word
two	two	CD	word
major	major	JJ	word
categories	category	NNS	word
:	:	:	punctuation
vertebrates	vertebrate	NNS	word
and	and	CC	word
invertebrates	invertebrate	NNS	word
.	.	SENT	punctuation

Table 32. Example of Annotation Results by ANNIE (GATE)

Although the GATE architecture can provide much more information about texts in NL, preliminary experiments show that the results of the selected annotations are enough to carry out the matching recommendation action between NL sentences and ODPs.

Step 3: Matching of tagged sentence to ODPs (by means of JAPE rules). The output of the 2nd step, namely, the tagged sentence, is analyzed against the JAPE rules created for all LSPs associated to ODPs (see Table 31 in section 8.3.1 for an example of a JAPE rule). As result of this analysis, GATE returns a matching recommendation, which can be of two types:

- *Exact matching* = 1 tagged sentence – 1 ODP
- *Inexact matching* = 1 tagged sentence – N ODPs

If the result of the matching action is *Exact matching*, the SOS NeOn plug-in normally goes to *Step 5*, except for some special situations, in which the knowledge represented by the corresponding ODP can be enriched with additional knowledge, and it then goes to *Step 4*. From the set of ODPs enriched with LSPs so far, this happens when the *Exact matching* is performed with the *SubClassOf* ODP (LP-SC-01).

If, on the contrary, the tagged sentence is matched to more than one ODP (*Inexact matching*), a refinement of the initial input has to be performed in order to obtain an *Exact matching*. In that case, the system goes to *Step 4*. The cause of *Inexact matching* in the analyzed ODPs is *lexical ambiguity*⁶⁸, which means, that some of the verbs in which LSPs are based can have different meanings (in other words, they are considered polysemous verbs⁶⁹). This is the specific case of some verbs as *divide* or *include* that are present in the identified LSPs for *SubClassOf* (LP-SC-01) and *Part-WholeRelation* (CP-PW-01). This will be detailed in section 8.3.3.

Step 4: Input refinement. The previous step ends in two outputs:

a) *Exact matching to 1 ODP advisable to be enriched with additional knowledge.* This case is exemplified by the *SubClassOf* ODP (LP-SC-01) [110]. From the ontological perspective, it is recommendable to further specify this basic taxonomical relation by adding knowledge about disjointness and/or exhaustiveness. The purpose of this is basically to develop a robuster ontology with a richer conceptualization to avoid eventual errors and inconsistencies in future ontology-based applications. Therefore, the SOS NeOn plug-in will offer the user the option of refining the

⁶⁸ Lexical ambiguity is a linguistic phenomenon by which certain lexical entries have more than one meaning (based on [96])

⁶⁹ Polysemy happens when the same lexical entry has different meanings.

input with that kind of knowledge by means of some question-answering techniques that basically launch queries to the user that (s)he has the option of answering (or not) for enabling an *Input refinement*. Details are provided in section 8.3.3.

b) *Inexact matching to N ODPs to be disambiguated*. In this case, a refinement of the input is compulsory, because *Exact matching* must be achieved for the NeOn SOS plug-in to continue the process. The SOS NeOn plug-in will interact with the user until *Exact matching* is obtained. As already mentioned, this case is exemplified by the *SubClassOf* ODP (LP-SC-01) and the *Part-WholeRelation* ODP (CP-PW-01), since some LSPs associated to them overlap. The *Input refinement* process has been explained in more detail in section 8.3.3.

Step 5: Final output. Once the *Exact matching* has been performed, the corresponding ODP is selected and returned to the user in the form of a NeOn UML diagram, which is instantiated with concepts, relations, etc. extracted from the NL sentence. The SOS NeOn plug-in is capable of identifying the elements in the NL sentence and the role they play in the knowledge expressed by the sentence. Therefore, the NeOn UML diagram provided in ODPs is instantiated with the corresponding concepts, relations, etc. This is the output shown to the user, as can be seen in Figure 37.

Once the process has been repeated with several NL sentences introduced by the user, the SOS NeOn plug-in relates the instantiated NeOn UML diagrams among each other, whenever this is possible, and the output is a complete picture of the whole conceptualization.

8.3.3. Input Refinement

In this section, our aim is to illustrate how the SOS NeOn plug-in is expected to tackle the problems we have already identified in Step 3 of the SOS plug-in workflow (section 8.3.2).

a) Exact matching to 1 ODP advisable to be enriched with additional knowledge

This situation results from the suitability of specifying the basic taxonomical relation represented by the *SubClassOf* ODP (LP-SC-01) with knowledge about disjointness and/or exhaustiveness.

Disjointness is generally understood in ontological modelling as the property of two classes of not sharing subclasses or individuals. Exhaustiveness has to do with the property of a set of classes of belonging to a superclass and entirely including all individuals that belong to that superclass, without excluding any of them.

In D5.1.1 [110], the pattern for modelling *DisjointClasses* (LP-Di-01) is meant to express that *an element belonging to a certain group or set, cannot belong to another group of set*. The one for modelling *ExhaustiveClasses* (LP-EC-01) is defined as *the union of set of mutually disjoint subclasses*.

Assuming that the *SubClassOf* ODP (LP-SC-01) has been matched in a particular case, it is advisable to find out if subclasses are mutually disjoint. If this is the case, subclasses are additionally modelled as *disjoint*, and thus apart from the LP-SC-01, the LP-Di-01 is also matched. Further, if it happens that the set of subclasses is also complete or exhaustive and covers the superclass, then this relation has to be additionally modelled as *exhaustive*, and thus the LP-EC-01 is also matched. The possibilities for enriching the subclass-of relation represented by the *SubClassOf* ODP (LP-SC-01) with disjointness and exhaustiveness knowledge are illustrated in Figure 38.

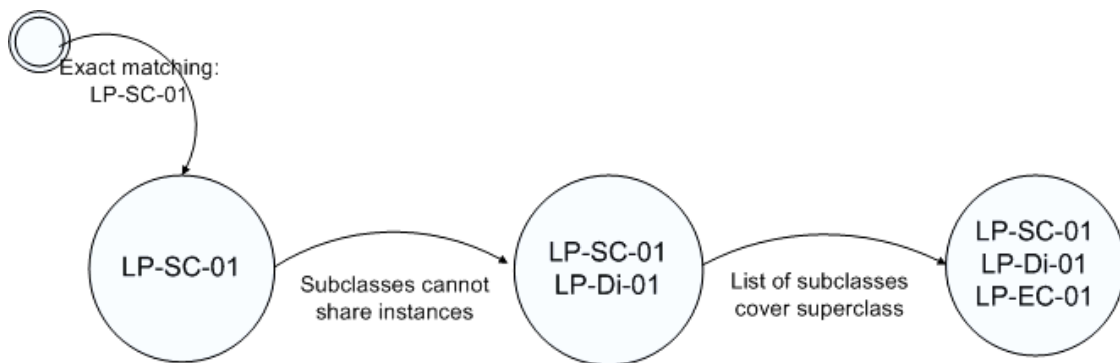


Figure 38. Possibilities for enriching Taxonomies

It is foreseen that the user assists the SOS NeOn plug-in in discovering if the taxonomy or hierarchy (s)he wants to model can be completed with disjoint and/or exhaustive knowledge. For this reason, we have decided to refine the input by asking the user some questions that he will have to answer with *yes* or *no* in the input window of the NeOn editor. Let us illustrate this technique in the light of one of the examples introduced in section 8.3.1:

- 1) *Animals are divided into two major categories: vertebrates and invertebrates*

Some examples of questions are shown below.

- Regarding *disjointness*:

-Can a certain animal belong to the category of vertebrates and to the category of invertebrates at the same time?

If the answer is *no*, the system will further model those subclasses as disjoint classes. If the answer is *yes*, it will remain modelled as *SubClassOf* relation.

- Regarding *exhaustiveness*:

-Are there any other types of animals?

If the answer is *yes*, the system will offer the user the possibility of introducing the missing subclasses in the input window of the NeOn editor. As soon as the user completes the list of subclasses, it appears updated in the input window and the system assumes that the list has been completed with the missing classes. Therefore, the system proceeds to model those classes according to the *ExhaustiveClasses* relation.

Should the answer to this question be *no*, the system would directly assume that the enumeration of subclasses is exhaustive, and will proceed to formalize it in that way.

As already outlined, the *ExhaustiveClasses* relation identified in D5.1.1 implies disjointness. This means that, if the classes expressed by the user are disjoint and exhaustive, this will be represented by the patterns: *SubClassOf* relation and *ExhaustiveClasses* relation. However, there is no pattern in the current repository for expressing that classes are exhaustive but not disjoint. In this case, just the *SubClassOf* relation could be represented, and the information about exhaustiveness would be lost.

b) Inexact matching to N ODPs to be disambiguated

This situation results from the ambiguity present in some of the lexical items included in LSPs. In fact, there are some ODPs that have associated the same LSP, as in the case of some LSPs for the *SubClassOf* ODP (LP-SC-01) and the *Part-WholeRelation* ODP (CP-PW-01). Let us take a look at the sentences below:

- 1) *Drugs are divided into A, B and C.*
- 2) *Each half of the brain is divided into X lobes.*

Either sentences match the same LSP, since the verb *divide into* can introduce *types of or parts of*.

LSP 4: NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>)* and NP<subclass>

This LSP is associated to two ODPs, the one for the *SubClassOf* relation, and the other for the *Part-Whole* relation.

In the following examples, we find a similar example, because the verb *include* is as well ambiguous. It can express that what follows are *types of or parts of*.

3) *Arthropods include insects, crustaceans, spiders, scorpions, and centipedes.*

4) *Reproductive structures in female insects include ovaries, bursa copulatrix and uterus.*

Again, both sentences match the same LSP:

LSP 4: NP<superclass> CATV [CD] [CN] [PARA] (NP<subclass>)* and NP<subclass>

Obviously, the difference is clear for an intelligent being, but not for a tool. Here again we have to resort to the question-answering technique for refining the search. Some examples of the so-called “refining questions” for sentence 1) are:

-Are A, B and C types of drugs?

-Are A, B and C parts or components of drugs?

In this case, the answer to the first question should be *yes*, and to the second one, *no*, because A, B and C are types of drugs, and not parts of it. By interacting with the SOS NeOn plug-in in this way, the user would help it to disambiguate the relation expressed by the NL sentence, and to determine the *Exact matching* to the *SubClassOf* relation (LP-SC-01).

8.4. Proposed Guidelines for Ontology Design Patterns Reuse by Naive Users

In this section we include the preliminar methodological guidelines proposed in NeOn for building ontology networks by reusing Ontology Design Patterns (ODPs). As already mentioned in section 8.1, we make a distinction between two types of users: naive users and expert users, and in this version of the deliverable we propose guidelines for naive users.

We assume the existence of repositories or catalogues of Ontology Design Patterns. These may be of great use to expert users, but opaque to naive user, who may encounter difficulties in understanding ontology languages such as OWL or RDF, UML representation diagrams, etc. In both cases we assume that users have sufficient knowledge of the domain for which they want to develop an ontology.

The goal of the Ontology Design Patterns reuse by naive users is to enable the use Ontology Design Patterns in the development of the ontology for solving modelling difficulties. The output of this activity is an Ontology Design Pattern integrated into the ontology network being developed.

Table 33 shows the filling card for the ODPs reuse activity by naive users, including definition, goal, inputs, outputs, who carries out the task, and when the task should be taken.

Ontology Design Patterns (ODPs) Reuse by Naive Users	
<i>Definition</i>	
<p><i>Ontology Design Patterns (ODPs) Reuse</i> is defined as the activity of using ontology design patterns in the solution of different modelling problems during the development of new ontologies or during the activity of ontology aligning (as background knowledge).</p>	
<i>Goal</i>	
<p>The goal is to allow the reuse of ODPs during the ontology development in order to model those parts of the ontology that present modelling difficulties to the user.</p>	
<i>Input</i>	<i>Output</i>
<p>Modelling problem during the ontology development.</p>	<p>Ontology design pattern integrated into the ontology network being developed.</p>
<i>Who</i>	
<p>Software developers and ontology practitioners that have little expertise in the ontology development task and insufficient command of ontology languages (OWL, RDF(S), etc.), Ontology Design Patterns (ODPs), UML diagrams, etc.</p>	
<i>When</i>	
<p>During the development of the Ontology Conceptualization activity, the Ontology Formalization activity, or the Ontology Implementation activity.</p>	

Table 33. ODPs Reuse by Naive users Filling Card.

In this use case, users access Ontology Design Patterns repositories in an indirect way, by means of the SOS NeOn plug-in, described in 8.3. Tasks involved in this activity are explained in the following and shown in Figure 39.

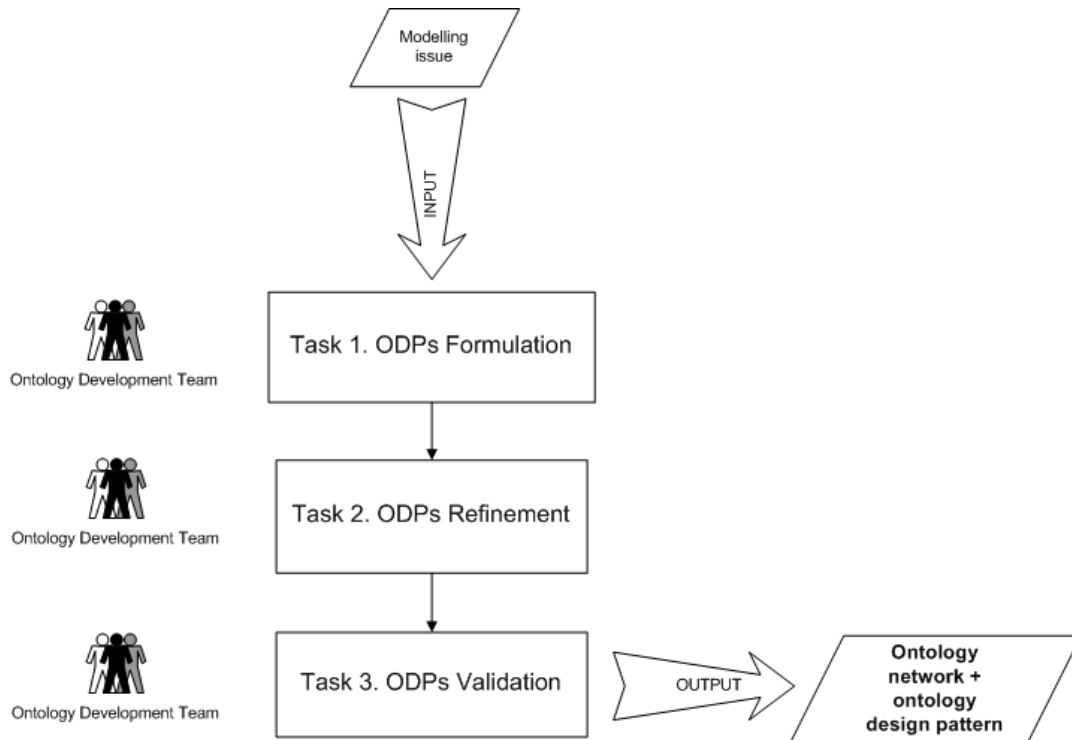


Figure 39. Tasks for ODPs Reuse by Naive Users

Task 1. ODPs Formulation.

The goal of this task is the formulation in NL of the domain aspect to be modeled. Naive software developers and ontology practitioners carry out this task taking as input a modelling problem, and using their own knowledge of the domain. The task output is a sentence in NL expressing the domain aspect to be modelled.

Formulating right expressions of conceptual relations in NL is crucial for the success of the modelling process. The user should check the advices given by the SOS NeOn plug-in before starting the formulation task. For example, some of the advices given by the SOS NeOn plug-in may propose the user to express statements in an assertive way, avoiding the use of unnecessary adverbs or adjectives, and exclusively introducing those words that are required for expressing how concepts are related. We are currently working on the identification of additional advices.

The SOS NeOn plug-in will provide support for three natural languages in a first stage: English, Spanish and German.

Task 2. ODPs Refinement.

The goal of this task is refining the NL sentence resulting from the previous task. Naive software developers and ontology practitioners carry out this task taking as input one or various *refining questions* returned to the user by the SOS NeOn plug-in using NL techniques. The task output is one or a set of answers that the SOS NeOn plug-in will process for refining the original NL sentence. This task will be repeated until the exact matching is obtained.

For example, in NL expressions such as *Drugs are divided into A, B and C* or *Each half of the brain is divided into X lobes*, the system may not be able to distinguish between the *SubClassOf* ODP (LP-SC-01) and the *PartWholeRelation* ODP (CP-PW-01). Regarding especial modelling issues as the one rose by the *SubClassOf* OPD (for example in *Animals are divided into two major categories: vertebrates or invertebrates*), it is recommendable to enrich such relation with knowledge about *disjointness* (LP-Di-01) or *exhaustiveness* (LP-EC-01).

Task 3. ODPs Validation.

The goal of this task is the validation of the NeOn UML diagram returned by the SOS plug-in corresponding to the ODP that matches the modelling issue expressed by the user through the NL sentence. Naive users carry out this task taking as input the NeOn UML diagram (extended or instantiated with information from the NL formulation), displayed in the NeOn editor. The task output is an ODP ready to be integrated in the ontology network being developed.

9. Conclusions and Future Work

After analysing the state of the art on existing methodologies for building ontologies, we can say that METHONTOLOGY and On-To-Knowledge are up to now the most complete methodologies for building ontologies from scratch. They mainly include guidelines for supporting single ontology construction from the ontology specification to the implementation. The three analyzed methodologies (METHONTOLOGY, On-To-Knowledge, and DILIGENT) do not treat the critical dimensions identified within the NeOn project, which are collaboration, context and dynamics. Furthermore, the degree of coverage given by those methodologies to the same processes and activities included in this deliverable is very low. And finally, the analyzed methodologies are not targeted to software developers and ontology practitioners in general, but towards ontology researchers.

As already mentioned, our aim within the NeOn project is to create the *NeOn methodology for building ontology networks* covering the drawbacks presented in the three analyzed methodologies, and benefiting from the advantages included in such methodologies, with respect to the aforementioned characteristics.

Concretely, regarding NeOn dimensions, the first version of the NeOn methodology for building ontology networks benefits from the collaboration aspects included in DILIGENT. Furthermore, we have taken into account the proposal given by METHONTOLOGY and On-To-Knowledge about the use of competency questions for the ontology specification activity to create the methodological guidelines for this activity presented in this deliverable. With respect to the reuse of ontologies, using the list of activities proposed by METHONTOLOGY, we have improved and extended them to propose the corresponding methodological guidelines in the first version of the NeOn methodology.

Therefore, this deliverable has presented the first version of the NeOn methodology for building ontology networks taking into account the aforementioned characteristics. In this sense, this deliverable presents an step forward by means of the following contributions:

- ❑ Analysis of how argumentation and collaboration dimensions are related to the different nine identified scenarios for collaboratively building network of ontologies. This analysis is presented in section 4.2.
- ❑ Preliminar guidelines for deciding if it is better to develop a single ontology or an ontology network are presented in section 4.3.
- ❑ Prescriptive methodological guidelines for carrying out the ontology specification activity, including three examples on how to apply the proposed methodological guidelines. Such guidelines are provided in chapter 5.
- ❑ Methodological guidelines for reusing and reengineering non ontological resources are presented in chapter 6. In this case, a typology of non ontological resources is also provided.
- ❑ Prescriptive methodological guidelines for reusing ontological resources, focused on general or common ontologies, domain ontologies as a whole, and ontology statements are provided in chapter 7.
- ❑ Methodological guidelines for reusing ontology design patterns by naive users are presented in chapter 8.

To conclude, we present the comparison between the three analyzed methodologies (METHONTOLOGY, On-To-Knowledge, and DILIGENT) and the first version of the NeOn methodology for building ontology networks, with respect to the following characteristics: (1) Neon dimensions that are: collaboration, context and dynamics; (2) degree of coverage of the process or

activities included in this deliverable by means of providing detailed guidelines; and (3) and methodology audience. Such a comparison is shown in Table 34

	METHONTOLOGY	On-To-Knowledge	DILIGENT	NeOn Methodology (Version1)
NeOn Dimensions				
Collaboration	Not mentioned	Not mentioned	Treated	<i>Mentioned, but not treated in detail</i>
Context	Not mentioned	Not mentioned	Not mentioned	<i>Not mentioned</i>
Dynamic	Mentioned, but not treated	Mentioned, but not treated	Mentioned, but not treated	<i>Not mentioned</i>
Detailed Guidelines for Processes and Activities				
Ontology Specification	Not provided Only Competency Questions are proposed	Not provided Only Competency Questions are proposed	Not provided In fact, this activity is not proposed by the methodology	<i>Provided</i>
Reusing Non Ontological Resources	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	<i>Provided</i>
Reengineering Non Ontological Resources	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	<i>Provided in a preliminar manner</i>
Reusing Ontologies	Not provided Only a list of activities to be carried out is proposed	Not provided Only recommendation of identifying ontologies to be reused is given	Not provided, neither explicitly mentioned	<i>Provided</i>
Reusing Ontology Design Patterns	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	Not provided, neither explicitly mentioned	<i>Provided in a preliminar manner</i>
Audience				
Targeted to Software Developers and Ontology Practitioners	Targeted to ontolgy engineers and researchers	Not targeted to ontolgy engineers and researchers	Intended to domain experts and users	<i>Targeted to ontolgy engineers and researchers</i>

Table 34. Comparative Analysis of Three Analyzed Methodologies and the NeOn Methodology Version 1

Furthermore, future methodological work (methods, techniques and tools) for continuing the presented step forward will be included in D5.3.2 and D5.4.2 at M36.

The second version of this deliverable, that is D5.4.2, will be focused on:

- ❑ improving and extending the methodological guidelines proposed here;
- ❑ selecting, comparing and combining non ontological resources, ontological resources, and ODPs for building ontology networks;
- ❑ evaluating ontology networks;
- ❑ modularizing ontology networks; and
- ❑ evolving ontology networks.

Currently, we are also analysing which is the coverage of the existing and planned NeOn plug-ins with respect to the activities included in the NeOn Glossary. The results of such an analysis will be included in D5.3.2, which will be an improved version of D5.3.1. D5.3.2 will include updated and more detailed guidelines on how to establish an ontology network life cycle, as part of the scheduling activity. Additionally, the NeOn plug-in “gOntf”, which supports the scheduling activity, will be also explained.

References

1. *IEEE Standard for Information Technology. Software Life Cycle Processes. Reuse Processes.* IEEE Std 1517-1999 (R2004).
2. *IEEE Standard for Developing Software Life Cycle Processes.* IEEE Std 1074-1997 (Revision of IEEE Std 1074-1995; Replaces IEEE Std 1074.1-1995).
3. *IEEE Guide for Developing Software Life Cycle Processes.* IEEE Std 1074.1-1995.
4. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE St 830-1993.
5. *IEEE Standard Glossary of Software Engineering Terminology.* IEEE Std 610.12-1990 (Revision and redesignation of IEEE Std 792-1983).
6. *IEEE Standard Glossary of Data Management Terminology.* IEEE std 610.5-1990.
7. *ISO/IEC International Standard 11179, Part 1, Framework for the specification and standardization of data elements,* 1999.
8. ISO 1087-1:2000. *Terminology work. Vocabulary. Part 1: Theory and application.*
9. ISO 1087:1990. *Terminology work -- Vocabulary -- Part 1: Theory and application.*
10. *ANSI/NISO Z39.19-2005 Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies.* 2005.
11. *Glossary of Terms for the Standardization of Geographical Names,* United Nations Group of Experts on Geographic Names, United Nations, New York, 2002.
12. KACTUS (1996) *The KACTUS Booklet version 1.0.* Esprit Project 8145 KACTUS. <http://www.swi.psy.uva.nl/projects/NewKACTUS/Reports.html>.
13. SEEMP Consortium. *SEEMP D31a. Supporting the State of the Art.* July 2006. <http://www.seemp.org/>.
14. SEEMP Consortium. *SEEMP D11. User Requirement Definition.* May 2006. <http://www.seemp.org/>.
15. J.C. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez. *WebODE in a nutshell.* AI Magazine. 2003.
16. R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley. 1999. ISBN 0-201-39829.
17. J. Barrasa, O. Corcho, A. Gómez-Pérez. *R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language.* Second Workshop on Semantic Web and Databases (SWDB2004), 2004.
18. A. Bernstein, E. Kaufmann. *GINO-a guided input natural language ontology editor.* In Proc. 4th. ISWC'06, 2006.

19. M. Blázquez, M. Fernández-López, J.M. García-Pinar, A. Gómez-Pérez. *Building Ontologies at the Knowledge Level using the Ontology Design Environment*. 1998. In: Gaines BR, Musen MA (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98). Banff, Canada, SHARE4:1–15.
20. S. Borgo, N. Guarino, C. Masolo. *An Ontological Theory of Physical Objects*. In: Ironi I (ed) 11th International Workshop on Qualitative Reasoning (QR 1997). Cortona, Italy, pp 223–231.
21. S. Borgo, N. Guarino, C. Masolo. *A Pointless Theory of Space Based on Strong Connection and Congruence*. In: Carlucci-Aiello L, Doyle J (eds) 5th International Conference on Principles of Knowledge Representation and Reasoning (KR 1996). Morgan Kaufmann Publishers, San Francisco, California, pp 220–229.
22. W.N. Borst. *Construction of Engineering Ontologies*. Centre for Telematica and Information Technology, University of Twente. Enschede, The Netherlands. 1997.
23. D Brandon. *Recursive Database Structures*. ACM Digital Library, Journal of Computing Sciences in Colleges, Volume 21. 2005.
24. D. Brickley, R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Working Draft. <http://www.w3.org/TR/PR-rdf-schema>. 2004.
25. F. Bushmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-oriented software architecture. A system of patterns*. John Wiley & Sons, Chichester. 1996.
26. T. Buzan. *Use your head*. BBC Books. 1974.
27. E.J. Byrne. *A conceptual foundation for software re-engineering*. In Proceedings of the International Conference on Software Maintenance and Reengineering, pages 226–235. IEEE Computer Society, 1992.
28. C. Caracciolo, A. Gangemi. *NeOn Deliverable D7.2.2. Revised and Enhanced Fisheries Ontologies*. August 2007. Available at: <http://www.neon-project.org/>
29. R. Casati, B. Smith, A. Varzi. *Ontological Tools for Geographic Representation*. In N. Guarino (ed) Formal Ontology in Information Systems, Trento. Italy IOS Press, Amsterdam, pp 77–85. 1998.
30. R. Casati, A. Varzi. *Holes and other superficialities*. MIT Press, Cambridge, Massachusetts. 1995.
31. P. Cimiano. *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*. Springer. ISBN 0-387-30632-3. 2006.
32. R.A. Coté, D.J. Rothwell, L. Brochu, eds. *SNOMED International* (3rd ed.), Northfield, Ill, College of American Pathologists, 1994.
33. M. d'Aquin, A. Schlicht, H. Stuckenschmidt, M. Sabou. *Ontology Modularization for Knowledge Selection: Experiments and Evaluations*. 18th International Conference on Database and Expert Systems Applications. DEXA 2007, Regensburg, Germany.
34. M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, E. Motta. *Characterizing Knowledge on the Semantic Web with Watson*. Workshop on Evaluation of Ontologies and Ontology-based tools, 5th International EON Workshop, collocated with the International Semantic Web Conference (ISWC'07), Busan, Korea.

35. M. d'Aquin, C. Baldassarre, L. Gridinoc, M. Sabou, S. Angeletou, E. Motta. *Watson: Supporting Next Generation Semantic Web Applications*. WWW Internet conference 2007, Spain.
36. A. Davis. *Software Requirements: Objects, Functions and States*, Upper Saddle River, New Jersey: Prentice Hall, 1993.
37. J. de Bruijn, H. Lausen, A. Polleres, D. Fensel. *The web service modeling language: An overview*. In: Proceedings of the 3rd European Semantic Web Conference (ESWC2006), Budva, Montenegro, Springer-Verlag (2006).
38. K. Dellschaft, H. Engelbrecht, J. Monte Barreto, S. Rutenbeck, S. Staab. *Cicero: Tracking Design Rationale in Collaborative Ontology Engineering*. Proceedings of the ESWC 2008 Demo Session. Available at: <http://www.uni-koblenz.de/~klaasd/Downloads/papers/Dellschaft2008CTD.pdf>.
39. M. Egaña Aranguren, R. Stevens, E. Antezana. *Ontology Design Patterns for bio-ontologies*. In Bio-Ontologies SIG Workshop 2007 at ISMB/ECCB, 2007.
40. M. Engler, D. Vrandečić, Y. Sure. *A Tool for DILIGENT Argumentation: Experiences, Requirements and Design*. In Robert Tolksdorf and Elena Paslaru Bontas and Klaus Schild, 1st International Workshop on Semantic Technologies in Collaborative Applications STICA 06. IEEE, IEEE, Manchester, UK, June 2006.
41. J. Euzenat. *Corporate memory through cooperative creation of knowledge bases and hyperdocuments*. In: Gaines BR, Musen MA (eds) 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96). Banff, Canada. 1996. <http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html>.
42. J. Euzenat. *Building Consensual Knowledge Bases: Context and Architecture*. In: Mars N (ed) Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases (KBKS '95). University of Twente, Enschede, The Netherlands. IOS Press, Amsterdam, The Netherlands, pp 143–155. 1995.
43. M. E. Fayad, G. K. Srikanth. *Choosing the Right Pattern- Real Challenges*. [Available at The Software Patterns Blog at pattern.ijop.org, <http://pattern.ijop.org/?p=20> (31-01-08)].
44. J. Feliu Cortès. *Relacions conceptuals i terminologia: anàlisi i proposta de detecció semiautomàtica*. PhD Thesis. Institut Universitari de Lingüística Aplicada Universitat Pompeu Fabra. 2004.
45. D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, J. Domingue. *Enabling Semantic Web Services - The Web Service Modeling Ontology*. 2007. Springer Verlag. ISBN 3-540-34519-1 & 978-3-540-34519-0.
46. M. Fernández-López, A. Gómez-Pérez. *Searching for a time ontology for Semantic Web applications*. In Vari A, Vieu L (eds) 3th Formal Ontology in Information Systems, Turin, Italy, pp 331-441. 2004.
47. M. Fernández-López, A. Gómez-Pérez, M. D. Rojas-Amaya. *Ontologies' crossed life cycles*. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW 2000). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp 65–79.
48. M. Fernández-López, A. Gómez-Pérez, N. Juristo. *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. 1997. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33–40.

49. A. Funk, B. Davis, V. Tablan, K. Bontcheva, H. Cunningham. *SEKT Deliverable D2.2.2 Report: Controlled Language IE Components version 2*. SEKT Project, <http://www.sekt-project.com>. January, 2007.
50. A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, S. Handschuh. *CLOnE: Controlled Language for Ontology Editing*. In Proceedings of the ISWC'07, 2007.
51. E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley: New York. 1995.
52. A. Gangemi, C. Catenacci, M. Battaglia. *Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations*. In D. M. Pisanelli (ed.), *Ontologies in Medicine*. IOS Press, Amsterdam. 2004.
53. A. Gangemi, D. Pisanelli, G. Steve. *Ontology Integration: Experiences with Medical Terminologies* Formal Ontology in Information Systems IOS Press, pp. 163—178. 1998.
54. A. Gangemi, C. Catenacci, M. Ciaramita, J. Lehmann. *Modelling ontology evaluation and validation*. Proceedings of the Third European Semantic Web Conference, ESWC06, volume 4011 of LNCS. Springer, pp. 140-154. 2006.
55. A. Gangemi, N. Guarino, C. Massolo, A. Oltramari. *Understanding top-level ontological distinctions*. In: Gómez-Pérez A, Grüninger M, Stuckenschmidt H, Uschold M (eds) IJCAI 2001 Workshop on Ontologies and Information Sharing. Seattle, Washington, pp 26-33. CEUR Workshop Proceedings 47:26-33. Amsterdam, The Netherlands (<http://ceur-ws.org/Vol-47/>).
56. R. García, O. Celma. *Semantic Integration and Retrieval of Multimedia Metadata*. Proceedings of the ISWC 2005 Workshop on Knowledge Markup and Semantic Annotation (Semannot'2005)". 2005.
57. M.R. Genesereth, R.E. Fikes. *Knowledge Interchange Format. Version 3.0. Reference Manual*. Technical Report Logic-92-1. Computer Science Department. Stanford University, California. <http://meta2.stanford.edu/kif/Hypertext/kif-manual.html>. 1992.
58. F. Giunchiglia, M. Marchese, I. Zaihrayeu. *Encoding Classifications into Lightweight Ontologies*. The Semantic Web: Research and Applications. Springer Berlin/ Heidelberg. 2006.
59. A. Gómez-Pérez, M. Fernández-López, O. Corcho. *Ontological Engineering*. November 2003. Springer Verlag. Advanced Information and Knowledge Processing series. ISBN 1-85233-551-3.
60. A. Gómez-Pérez, A. Lozano-Tello. *Applying ONTOMETRIC Method to Measure the Suitability of Ontologies*. Business Systems Analysis with Ontologies. Eds: Green, P.; Rosemann, M. Idea Group Publishing. 2005. PP: 249-269. ISBN: 1-59140-339-1.
61. A. Gómez-Pérez, M. D. Rojas-Amaya. *Ontological Reengineering for Reuse*. Knowledge Acquisition, Modeling and Management: 11th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW 1999. Dagstuhl Castle, Germany, May 1999. LNCS Volume 1621/1999. Springer-Verlag, Berlin, Germany, pp 139–156.
62. A. Gómez-Pérez, N. Juristo, C. Montes, J. Pazos. *Ingeniería del Conocimiento*. Editorial Centro de Estudios Ramón Areces, S.A. 1997. ISBN: 84-8004-269-9.
63. J.M Gómez-Pérez, T. Pariente, C. Daviaud, G. Herrero. *NeOn Deliverable D8.1.1. Analysis of the pharma domain and requirements*. 2006.

64. J.M. Gómez-Pérez, T. Pariente, C. Buil-Aranda, G. Herrero. *NeOn Deliverable D8.2.1. Software architecture for the NeOn pharmaceutical case studies*. 2007.
65. J.M. Gómez-Pérez, T. Pariente, C. Buil-Aranda, G. Herrero, A. Baena. *NeOn Deliverable D8.3.1. Ontologies for pharmaceutical case studies*. 2007.
66. E. Greenwood. *Metodología de la investigación social*. Paidós, Buenos Aires, Argentina. 1973.
67. M. Gruninger, M. S. Fox. *The role of competency questions in enterprise engineering*. In Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, Trondheim, Norway, 1994.
68. M. Gruninger, M.S. Fox. *Methodology for the design and evaluation of ontologies*. In Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6.1–6.10. (1995).
69. P. Haase, S. Brockmans, R. Palma, J. Euzenat, M. d'Aquin. *NeOn Deliverable D1.1.2. Updated Version of the Networked Ontology Model*. August 2007. Available at: <http://www.neon-project.org/>.
70. P. Haase, S. Rudolph, Y. Wang, S. Brockmans, R. Palma, J. Euzenat, M. d'Aquin. *NeOn Deliverable D1.1.1. Networked Ontology Model*. November 2006. Available at: <http://www.neon-project.org/>.
71. P. Hasse, J. Völker. *Ontology Learning and Reasoning- Dealing with Uncertainty and Inconsistency*. ISWC-URSW 2005: 45-55. 2005.
72. L. Han, T. Finin, C. Parr, J. Sachs, A. Joshi. *RDF123: a mechanism to transform spreadsheets to RDF*. Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06). 2006.
73. M. A. Hearst. *Automatic Acquisition of Hyponyms from Large Text Corpora*. In Proceedings of 14th International Conference on Computational Linguistics, pp. 539-545. 1992.
74. M. Hepp. *A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards*. Int'l Journal on Semantic Web & Information Systems (IJSWIS), Vol. 2, pp. 72-99. 2006.
75. M. Hepp, J. de Bruijn. *GenTax: A generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies*. Proceedings of the 4th European Semantic Web Conference (ESWC2007). pp. 129-144. Springer-Verlag, 2007.
76. G. Hodge. *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*. Council on Library and Information Resources. 2000. <http://www.clir.org/pubs/reports/pub91/contents.html>.
77. R. de Hoog. *Methodologies for Building Knowledge Based Systems: Achievements and Prospects*. In: Liebowitz J (ed) Handbook of Expert Systems. CRC Press Chapter 1, Boca Raton, Florida. 1998.
78. M. Hristozova, L. Sterling. *Experiences with ontology development for value-added publishing*. In S. Cranefield, T. Finin, V. Tamma, and S. Willmott, editors, Proc. of the OAS'03 Workshop, 2003.

79. A. Lozano-Tello. PhD Thesis: *Métrica de idoneidad de ontologías*. Spain. Universidad de Extremadura, 2002. ISBN: 84-7723-537-6.
80. A. Maedche, S. Staab. *Ontology Learning for the Semantic Web*. IEEE Intelligent Systems. 2001.
81. E. Malinowski, E. Zimányi. *Hierarchies in a multidimensional model: From conceptual modeling to logical representation*. Data & Knowledge Engineering. 2006.
82. E. Marshman, T. Morgan, I. Meyer. *French patterns for expressing concept relations*. In Terminology, 8:1, 1-29. 2002.
83. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, L. Schneider. *The WonderWeb Library of Foundational Ontologies*, Preliminary Report, WonderWeb deliverable D1.7 (<http://www.loa-cnr.it/Papers/DOLCE2.1-FOL.pdf>). 2003.
84. R. Mizoguchi, J. Vanwelkenhuysen, M. Ikeda. *Task Ontology for reuse of problem solving knowledge*. In: Mars N (ed) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS 1995). University of Twente, Enschede, The Netherlands. IOS Press, Amsterdam, The Netherlands, pp 46–57.
85. M. Mochol, E. Paslaru. *Practical Guidelines for Building Semantic eRecruitment Applications*, International Conference on Knowledge Management (iKnow'06), Special Track: Advanced Semantic Technologies. 2006.
86. N. F. Noy, D. L. McGuinness. *Ontology development 101: A guide to creating your first ontology*. Tech. rep., KSL-01-05, Stanford Knowledge Systems Laboratory., 2001.
87. L. Paradela. PhD Thesis: *Una Metodología para la Gestión del Conocimiento*. Spain. Universidad Politécnica de Madrid, 2001.
88. P.F. Patel-Schneider, P. Hayes, I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax*. W3C Recommendation. <http://www.w3.org/TR/owl-semantics/>. 2004.
89. R.A. Pease, I. Niles. *IEEE Standard Upper Ontology: A Progress Report*. The Knowledge Engineering Review 17(1):65–70. 2002.
90. H. S. Pinto, C. Tempich, S. Staab. *DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolVnG Engineering of oNTologies*. In Ramón López de Mantaras and Lorenza Saitta, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), August 22nd - 27th, pp. 393--397. IOS Press, Valencia, Spain, August 2004. ISBN: 1-58603-452-9. ISSN: 0922-6389.
91. H. S. Pinto, S. Staab, Y. Sure, C. Tempich. *OntoEdit empowering SWAP: A case study in supporting Distributed, Loosely-controlled and evolVnG Engineering of oNTologies (DILIGENT)*. Proceedings of the 1st European Semantic Web Symposium. Crete, May 10-12, 2004.
92. L. Prechelt. *An experiment on the usefulness of design patterns: Detailed description and evaluation*. Technical Report 9/1997. University of Karlsruhe. [Available at <http://citeseer.ist.psu.edu/121551.html> (21-02-08)].
93. R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. New York, NY: McGraw-Hill, 2001.

94. V. Presutti, A. Gangemi, S. David, G. Aguado de Cea, M. C. Suárez-Figueroa, E. Montiel-Ponsoda, M. Poveda. *NeOn Deliverable D2.5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies*. NeOn Project. <http://www.neon-project.org>. February 2008.
95. V. Presutti, A. Gangemi, S. David, G. Aguado de Cea, M.C. Suárez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. *NeOn Deliverable D2.5.1 A Library of Ontology Design Patterns: reusable solutions for collaboratively design of networked ontologies*. NeOn Project, <http://www.neon-project.org>. February, 2008.
96. J. Pustejovsky. *The Generative Lexicon*. MIT Press.1995.
97. A. Rector, J. Rogers. *Patterns, properties and minimizing commitment: Reconstruction of the Galen Upper Ontology in OWL*. In A. Gangemi and S. Borgo (eds.), Proceedings of the EKAW04 Workshop on Core Ontologies in Ontology Engineering. CEUR. 2004.
98. M. Sabou, S. Angeletou, M. d'Aquin, J. Barrasa, K. Dellschaft, A. Gangemi, J. Lehman, H. Lewen, D. Maynard, D. Mladenic, M. Nissim, W. Peters, V. Presutti, B. Villazón. *NeOn Deliverable D2.2.1. Selection and integration of reusable components from formal or informal specifications*. NeOn Project. <http://www.neon-project.org>. May 2007.
99. D. Sánchez Ruenes. *Domain Ontology Learning from the Web*. PhD Thesis, Departament de Llenguatges i Sistemes Informàtics. Universidad Politècnica de Catalunya. 2007.
100. L. Schneider. *How to Build a Foundational Ontology. The Object-Centered High-level Reference Ontology OCHRE*. Saarland University forthcoming publication. (<http://www.ifomis.uni-saarland.de/Research/Publications/forthcoming/ki2003epaper.pdf>). 2004.
101. G. Schreiber, B. Wielinga, H. Akkermans, W. van de Velde, A. Anjewierden. *CML: The CommonKADS conceptual modelling language*. In Steels L, Schreiber ATH, Van de Velde W (eds) *A Future for Knowledge Acquisition*. Proceedings of the 8th European Knowledge Acquisition Workshop EKAW 1994. (Lecture Notes in Artificial Intelligence LNAI 867), Springer-Verlag, Berlin/Heidelberg, pp 1-25.
102. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. van de Velde, B. Wielinga. *Knowledge engineering and management. The CommonKADS Methodology*. MIT press, Cambridge, Massachusetts. 1999.
103. P. Simons. *Parts: A Study in Ontology*. Clarendon Press, Oxford, United Kingdom. 1987.
104. B. Smith. *Boundaries: An Essay in Mereotopology*. In Hahn L (ed) *The Philosophy of Roderick Chisholm* (Library of Living Philosophers), LaSalle: Open Court, pp 534-561 (<http://ontology.buffalo.edu/smith/articles/chisholm/chisholm.html>). 1997.
105. B. Smith. *Mereotopology: A Theory of Parts and Boundaries*. *Data and Knowledge Engineering*, 20, 287-303. 1996.
106. B. Smith , A. Varzi. *Fiat and Bona Fide Boundaries*. *Philosophy and Phenomenological Research* 60: 401–420. 2000.
107. R. Snow, D. Jurafsky, A. Y. Ng. *Learning syntactic patterns for automatic hypernym discovery*. In *Advances in Neural Information Processing Systems* 17. 2004.
108. S. Staab, H.P. Schnurr, R. Studer, Y. Sure. *Knowledge Processes and Ontologies*. *IEEE Intelligent Systems* 16(1):26–34. (2001).

109. L. Stojanovic, N. Stojanovic, R. Volz. *A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web*. In Intelligent Information Processing, 2002.
110. M. C. Suarez-Figueroa, S. Brockmans, A. Gangemi, A. Gomez-Perez, J. Lehmann, H. Lewen, V. Presutti, M. Sabou. *NeOn Deliverable D5.1.1 Neon modelling components*. NeOn Project, <http://www.neon-project.org>. April, 2007.
111. M. C. Suárez-Figueroa, G. Aguado de Cea, C. Buil, C. Caracciolo, M. Dzbor, A. Gómez-Pérez, G. Herrero, H. Lewen, E. Montiel-Ponsoda, V. Presutti. *NeOn Deliverable D5.3.1. NeOn Development Process and Ontology Life Cycle*. NeOn Project. <http://www.neon-project.org>. August 2007.
112. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke. *OntoEdit: Collaborative Ontology Engineering for the Semantic Web*. In: Horrocks I, Hendler JA (eds) First International Semantic Web Conference (ISWC 2002). Sardinia, Italy. (Lecture Notes in Computer Science LNCS 2342) Springer-Verlag, Berlin, Germany, pp 221–235.
113. V. Svatek. *Design patterns for semantic web ontologies: Motivation and discussion*. In Proceedings of the 7th Conference on Business Information Systems, Poznan. 2004.
114. M. Uschold. *Building Ontologies: Towards A Unified Methodology*. In: Watson I (ed) 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems. Cambridge, United Kingdom. (1996). <http://citeseer.nj.nec.com/uschold96building.html>.
115. M. van Assem, M. Menken, G. Schreiber, J. Wielemaker. *A method for converting thesauri to RDF/OWL*. Proceedings of the Third International Semantic Web Conference (ISWC). pp. 17-31. Springer, 2004.
116. T. Vander Wal. *Folksonomy coinage and definition*. 2007. <http://www.vanderwal.net/folksonomy.html>.
117. G. van Heijst, A.T. Schreiber, B.J. Wielinga. *Using explicit ontologies in KBS development*. International Journal of Human-Computer Studies 45:183–292. 1997.
118. A. Varzi. *Spatial Reasoning and Ontology: Parts, Wholes, and Locations*. In Aiello M, Pratt-Hartmann I, van Benthem J (eds) Springer-Verlag, pp 945-1038. 2007.
119. A. Varzi. *Mereology*. In Zalta (ed) Stanford Encyclopedia of Philosophy, Stanford: CSLI (on line publication) (<http://plato.stanford.edu/entries/mereology/> . Last access: 6th February 2008). 2003.
120. D. A. Waterman. *A Guide to Expert Systems*. Addison-Wesley, Boston, Massachusetts. 1986.
121. B.J. Wielinga, A.T. Schreiber, J.A.C. Sandberg. *From Thesaurus to Ontology*. First International Conference of Knowledge Capture KCAP01 Victoria, British Columbia, Canada, ACM Press 2001.
122. B.J. Wielinga, J. Wielemaker, G. Schreiber, M. van Assem. *Methods for Porting Resources to the Semantic Web*. Proceedings of the First European Semantic Web Symposium (ESWS04) Heraklion, Greece. 2004.

Annex A. Hands-on Experiments in using the NeOn Watson Plug-in

Use Case 1: Ontology Enrichment

Given two ontologies (Documentation and Event Ontologies from the Knowledge Web project), this experiment consists in enriching both ontologies with new knowledge available in the web. Concretely:

- Enrich the Documentation Ontology with new types of documents related to European research projects.
- Enrich the Documentation Ontology with new properties involving the following concepts: Thesis, Article, Deliverable, and Template.
- Enrich the Event Ontology with new types of Events related to European research projects.
- Enrich the Event Ontology with new properties involving the following concepts: International Conference and Review.

Use Case 2: Ontology Development

Develop an ontology for representing the ingredients needed in the Paella recipe, reusing statements available in the web. The ontology should answer, at least, the following Competency Questions:

- What is a Paella?
- Which kind of pot and pan would you need for cooking Paella?
- Which kind of ingredients would you use for cooking Paella?

You can use the following information for developing the ontology:

Paella is a typical Spanish dish and is traditionally cooked in a "paellera" - a round flat pan with two handles - which is then put on the table. It is normally made using shellfish but can also be made with fish, chicken or rabbit. In many Spanish villages, especially in coastal areas, they use a giant paellera to cook a paella on festival days which is big enough to feed everybody.

INGREDIENTS:

Small onion, finely chopped	Small clams
Green pepper, finely chopped	Squid
Red pepper, boiled until soft and then cut into long thin strips	Mussels
Medium-sized tomatoes, skinned and finely chopped	Rice
Carrots, finely chopped	Garlic, coarsely chopped
Peas, cooked	Saffron
Prawns	Parsley, finely chopped
	Olive oil
	Water