



UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

DOCTORAL THESIS

**NeOn Methodology for Building Ontology Networks:
Specification, Scheduling and Reuse**

Author: María del Carmen Suárez de Figueroa Baonza

Advisors: Prof. Dr. Asunción Gómez Pérez

Dr. Mariano Fernández López

June 2010



UNIVERSIDAD POLITÉCNICA DE MADRID

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

Presidente: _____

Vocal: _____

Vocal: _____

Vocal: _____

Secretario: _____

Suplente: _____

Suplente: _____

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

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

Para Iván.

Para mis padres y mi hermana Cris.

Agradecimientos

Aunque esta página aparece como una de las primeras en mi tesis doctoral, en realidad es la última que he escrito. Ha sido la última página de un largo periodo de mi vida. Periodo en el que he aprendido muchas cosas, en el que ha habido momentos de desesperación y también momentos de felicidad. Periodo que ahora acaba, y que dejará paso, sin duda, a otra etapa nueva, quien sabe si mejor o peor, pero sobre todo nueva.

Reflexiono ahora sobre este largo periodo de mi vida, en el que he dedicado mi tiempo y esfuerzo al desarrollo de esta tesis doctoral. Se que todo hubiera sido más difícil sin las personas que han estado a mi lado en algún momento de esta etapa de mi vida.

Quiero agradecer su confianza, apoyo y cercanía a mi directora, Asun. Gracias por todas esas discusiones de investigación que hemos tenido y que han sido claves tanto en el nacimiento como en el desarrollo de esta tesis doctoral.

Quiero dar las gracias a mi director, Mariano, que me ha enseñado muchas cosas, que me ha dado sabios consejos, y que me ha ayudado a mejorar los resultado de la tesis.

Quiero expresar mi agradecimiento a aquellos profesores de la facultad que, de una manera u otra, me han ayudado y servido de apoyo durante este difícil periodo de doctoranda que hoy acaba. Gracias Aurora, Santiago y Ernestina por vuestros buenos consejos. Gracias Ricardo por tu ayuda con los modelos de ciclo de vida. Gracias Arminda por tus consejos y por tus revisiones estadísticas. Gracias Lupe por tus ánimos y tu ayuda terminológica.

Gracias también Charo por tu paciencia durante la revisión de esta tesis y por tus correcciones que han mejorado sustancialmente el inglés del documento.

Quiero dar las gracias a mis compañeros del proyecto NeOn, y especialmente, a Martin Dzbor, cuyas recomendaciones han enriquecido la presente tesis doctoral.

Quiero dar las gracias a mis compañeros del OEG por todos los momentos compartidos. Gracias a los que estaban al principio (Oscar, Miguel, Angel, Raúl G.), a los que han ido llegando (Boris, Carlos B., María, Oscar M.), y a los que se fueron en busca de aventuras (Javi, Alex, Angel B.). Y fundamentalmente, quiero dar las gracias por todo su apoyo y ayuda desinteresada a Elena y a Jose Ángel.

Quiero agradecer la comprensión y el aliento de mis amigos que han estado siempre conmigo, tanto en los momentos buenos como en los no tan buenos. Raúl, Maribel, Antonio, Merce, Carol, Diego, Laura, gracias a todos.

Quiero dar las gracias a toda la gente de mi entorno, mis abuelos, mis tíos, mis primos, los padres y la abuela de Iván. A los que están y a los que ya se fueron, gracias por las muestras de cariño y los ánimos constantes.

Quiero agradecer a mis padres, Marcelo y Juli, todo lo que me han dado y lo que me han enseñado. Papá, mamá, gracias por haberme brindado la oportunidad de elegir mi camino. Gracias por apoyarme en todas mis decisiones y compartir conmigo los buenos y malos momentos de la vida.

Quiero dar las gracias a mi hermana Cris que juega un papel muy importante en mi vida. Gracias Cris por todas las risas y los buenos ratos compartidos, por tus consejos, que me han aportado otra visión de las cosas, y por tus muestras de apoyo y ánimo cuando más lo necesitaba.

Y muy especialmente, quiero dar mil gracias a mi amigo, a mi compañero, a mi amor, a mi vida, a mi todo, a mi recién estrenado marido. Gracias Iván porque sin ti esta tesis doctoral no hubiera sido posible. Gracias por recorrer conmigo este duro camino, por comprenderlo todo y por ayudarme tanto, profesional y personalmente. Gracias por creer en mí y en mis posibilidades, y por conseguir que no desistiera en los momentos de debilidad. Gracias por ser como eres y por estar siempre a mi lado.

Table of Contents

Table of Contents.....	i
List of Tables.....	iv
List of Figures	vi
English Summary.....	ix
Resumen.....	xi
1. Introduction	1
1.1. Thesis Context	1
1.2. Thesis Main Motivations.....	2
1.3. Thesis Main Objective and Tasks	5
1.4. Thesis Main Contributions.....	6
1.5. Thesis Structure	6
2. State of the Art	9
2.1. Introduction	9
2.2. Basic Terminology from Software Engineering	9
2.3. State of the Art in Software Engineering	11
2.3.1. <i>Development Process</i>	11
2.3.2. <i>Life Cycle Models</i>	16
2.4. State of the Art in Knowledge Engineering	26
2.4.1. <i>Development Process</i>	26
2.4.2. <i>Life Cycle Models</i>	27
2.5. State of the Art in Ontology Engineering.....	30
2.5.1. <i>Main Methodologies</i>	30
2.5.2. <i>Ontology Requirements Specification</i>	38
2.5.3. <i>Ontological Resource Reuse</i>	41
2.6. Conclusions.....	43
3. Work Objectives	47
3.1. Terminological Clarifications	47
3.2. Goals and Open Research Problems	48
3.3. Contributions to the State of the Art	49
3.4. Work Assumptions: Premises, Hypotheses and Restrictions	50
4. Research Methodology	53
4.1. General Framework for Describing the NeOn Methodology	53
4.2. Requirements of the NeOn Methodology for Building Ontology Networks	56
4.2.1. <i>Necessary Requirements</i>	56
4.2.2. <i>Sufficient Requirements</i>	57
5. Glossary of Processes and Activities.....	59
5.1. Introduction	59
5.1.1. <i>Ontology Network Development Process</i>	59
5.1.2. <i>The Need for Reaching an Agreement on Processes and Activities</i>	59

5.1.3. Chapter Structure	60
5.2. Consensus Reaching Process	60
5.3. NeOn Glossary of Processes and Activities.....	69
5.3.1. Definitions in the NeOn Glossary	69
5.3.2. Getting Feedback from the Ontology Engineering Community.....	75
5.3.3. Towards Glossary Standardization.....	76
5.4. Required or If Applicable Processes and Activities	76
5.5. Processes and Activities Classification based on IEEE Groups	77
5.6. Relation among Processes and Activities included in the NeOn Glossary and Ontology Networks	79
5.7. Conclusions.....	82
6. Scenarios for Building Ontology Networks.....	83
6.1. Introduction	83
6.2. Nine Scenarios for Building Ontology Networks	83
6.2.1. Scenario 1: From Specification to Implementation.....	85
6.2.2. Scenario 2: Reusing and Reengineering Non-Ontological Resources	87
6.2.3. Scenario 3: Reusing Ontological Resources.....	88
6.2.4. Scenario 4: Building, Reusing and Reengineering Ontological Resources	90
6.2.5. Scenario 5: Reusing and Merging Ontological Resources.....	92
6.2.6. Scenario 6: Reusing, Merging and Reengineering Ontological Resources	93
6.2.7. Scenario 7: Reusing Ontology Design Patterns	94
6.2.8. Scenario 8: Restructuring Ontological Resources.....	95
6.2.9. Scenario 9: Localizing Ontological Resources	96
6.3. Relation between Processes and Activities and Scenarios	97
6.4. Collaboration and Argumentation in NeOn Scenarios	97
6.5. Conclusions.....	99
7. Life Cycle Models for Ontology Networks.....	101
7.1. Introduction	101
7.1.1. Ontology Network Life Cycle Model Definition	101
7.1.2. Preliminary Collection of Ontology Network Life Cycle Models.....	102
7.2. Waterfall Ontology Network Life Cycle Models	103
7.3. Iterative-Incremental Ontology Network Life Cycle Model	108
7.4. Relation between Scenarios and Life Cycle Models	109
7.5. Coverage of the Set of two Life Cycle Models	110
7.6. Conclusions.....	111
8. Ontology Requirements Specification.....	113
8.1. Introduction	113
8.2. Methodological Guidelines for Ontology Requirements Specification.....	114
8.3. Ontology Requirements Specification: Examples	120
8.4. Conclusions.....	121
9. Planning and Scheduling: Obtaining the Ontology Network Life Cycle.....	123
9.1. Introduction	123
9.2. Scheduling Ontology Development Projects	124
9.3. Groundwork for Scheduling Ontology Development Projects	126
9.3.1. Scenarios and Life Cycle Models	126
9.3.2. Scenarios and Processes and Activities.....	128
9.3.3. Processes , Activities, and Life Cycle Models	128

9.3.4. Order of Processes and Activities	128
9.4. gOntt: NeOn Toolkit plug-in for the Scheduling Activity	131
9.5. Guidelines for Scheduling with gOntt	136
9.6. Conclusions.....	138
10. Ontological Resource Reuse	139
10.1. Introduction	139
10.2. Overview of Ontological Resource Reuse	139
10.3. Reusing General or Common Ontologies	141
10.3.1. Time Modelling	143
10.3.2. Mereology Modelling.....	145
10.3.3. Methodological Guidelines for Reusing General Ontologies.....	149
10.4. Methodological Guidelines for Reusing Domain Ontologies as a Whole	173
10.5. Methodological Guidelines for Reusing Ontology Statements	177
10.6. Conclusions.....	183
11. Experimentation	185
11.1. Introduction	185
11.2. Action Research.....	186
11.2.1. Experiments on Scenarios for Building Ontology Networks.....	186
11.2.2. Experiments on Life Cycle Models	188
11.2.3. Experiments on Requirements Specification and Reuse	189
11.3. Controlled Experiments	190
11.3.1. Experiments for the ontology requirements specification activity.....	190
11.3.2. Supporting ontology network life cycle establishment.....	197
11.3.3. Scheduling ontology development with gOntt	204
11.4. Conclusions.....	206
12. Conclusions and Future Work.....	207
References.....	217
ANNEX I. Ontology Requirements Specification: Examples	231
1. SEEMP Reference Ontology Requirements Specification	231
2. Invoice Reference Ontology Requirements Specification	238
3. Semantic Nomenclature Reference Ontology Requirements Specification	241
ANNEX II. General Ontology Reuse: Example	247
ANNEX III. Hands-on Experiments in using the NeOn Watson Plug-in	255
ANNEX IV. Ontology Requirements Specification: User Study 1	257
ANNEX V. Ontology Requirements Specification: User Study 2	261
ANNEX VI. Ontology Network Life Cycle Establishment: User Study 1	263
ANNEX VII. Ontology Network Life Cycle Establishment: User Study 2	265
ANNEX VIII. Scheduling using gOntt: User Study	267

List of Tables

Table 1. Activity groups from IEEE [IEEE, 2006]	12
Table 2. Summary of conclusions	45
Table 3. Template for the process and activity Filling Card	55
Table 4. Template for processes and activities in the NeOn Glossary	63
Table 5. Required-If Applicable processes and activities	78
Table 6. Relation among processes and activities and ontology networks.....	81
Table 7. Correspondence between scenarios and processes and activities	98
Table 8. Relation between scenarios and life cycle models	110
Table 9. Ontology Requirements Specification Filling Card.....	114
Table 10. Template for the OSRD	115
Table 11. Scheduling Filling Card	125
Table 12. Correspondence between processes and activities and ontology network life cycle model phases.....	129
Table 13. Summary of notions in the time modelling	145
Table 14. Summary of the axioms and definitions in the mereology modelling	148
Table 15. General Ontology Reuse Filling Card	149
Table 16. Heuristics to determine what type of general ontologies is needed	151
Table 17. Examples of the heuristics to determine what type of general ontologies is needed....	152
Table 18. Features of the ontologies that implement time theories (based on the work described in [Fernández-López and Gómez-Pérez, 2004])	155
Table 19. Heuristics to transform CQs	156
Table 20. Example of CQs transformations	157
Table 21. Heuristics to determine which time features are useful	158
Table 22. Examples of the heuristics to determine which time features are useful	159
Table 23. Heuristics to determine which mereology features are useful	161
Table 24. Examples of heuristics to determine which mereology features are useful	163
Table 25. Decision criteria to select a general ontology	168
Table 26. Table to select a general ontology	170
Table 27. Domain Ontology Reuse Filling Card.....	173
Table 28. Assessment table.....	176
Table 29. Ontology Statement Reuse Filling Card.....	178
Table 30. Answer to the question selected	205
Table 31. Answers to the questions selected	205
Table 32. Experiments and the related hypothesis.....	206

Table 33. Comparative analysis of the three methodologies analysed and the NeOn Methodology	212
Table 34. Examples of terminology.....	235
Table 35. Excerpt of SEEMP Reference Ontology Requirement Specification Document.....	237
Table 36. Excerpt of Invoice Ontology Requirements Specification Document.....	240
Table 37. Example of terms and frequency	244
Table 38. Examples of objects.....	244
Table 39. Excerpt of Semantic Nomenclature Reference Ontology Requirement Specification Document.....	245
Table 40. Excerpt of the competency questions of PPO	247
Table 41. Features of ontologies that implement mereology theories	249
Table 42. Competency questions reformulation	250
Table 43. Features required for the PPO	251
Table 44. Analysis of the candidate mereology ontologies	253
Table 45. Instantiation 1 of the preliminary methodological guidelines for Ontology Requirements Specification	257
Table 46. Instantiation 2 of the preliminary methodological guidelines for Ontology Requirements Specification	258
Table 47. Questionnaire about the preliminary methodological guidelines for Ontology Requirements Specification	260
Table 48. Questionnaire about the methodological guidelines for Ontology Requirements Specification.....	261
Table 49. Questionnaire about the methodological guidelines for establishing the Ontology Network Life Cycle	263
Table 50. Questionnaire about the methodological guidelines for establishing the Ontology Network Life Cycle	265
Table 51. Questionnaire about the use of the gOntt plug-in	268

List of Figures

Figure 1. Graphical representation of terminological relationships in methodologies [Gómez-Pérez et al., 2003].....	11
Figure 2. Process groups in the software development process [ISO, 2006]	14
Figure 3. Pure waterfall life cycle model	17
Figure 4. Waterfall life cycle model [Sommerville, 2007]	18
Figure 5. 'V' life cycle model	19
Figure 6. Spiral life cycle model [Boehm, 1976].....	22
Figure 7. Extreme programming methodology.....	25
Figure 8. Example of CommonKADS cycle [Schreiber et al., 1994]	28
Figure 9. Conical-Spiral life cycle model [Alonso et al., 1995]	29
Figure 10. METHONTOLOGY ontology development process.....	31
Figure 11. METHONTOLOGY ontology life cycle	33
Figure 12. Ontology reengineering activities [Gómez-Pérez and Rojas-Amaya, 1999]	34
Figure 13. On-To-Knowledge ontology life cycle [Staab et al., 2001]	36
Figure 14. DILIGENT ontology life cycle [Pinto et al., 2004].....	38
Figure 15. Simple graphical examples of single ontologies, of a set of interconnected single ontologies, and of ontology networks.....	48
Figure 16. Inputs taken into account for creating the NeOn Methodology.....	53
Figure 17. Process, activities, and tasks.....	54
Figure 18. Consensus reaching process	62
Figure 19. Approach to creating initially the NeOn Glossary of Processes and Activities	64
Figure 20. Knowledge acquisition activity: table of preferences	65
Figure 21. Processes and activities agreed upon in the first round, on 23 rd January at Bled	66
Figure 22. Activities agreed upon in the second round, on 29 th June at Dubrovnik.....	66
Figure 23. Processes and activities agreed upon in the third round during July 2007	67
Figure 24. History of the consensus reaching process	68
Figure 25. NeOn Glossary of Processes and Activities	74
Figure 26. Processes and activities classification.....	79
Figure 27. Scenarios for building ontologies and ontology networks.....	84
Figure 28. Levels of abstraction for the ontological resource reengineering process	90
Figure 29. Ontological Resource Reengineering Model	91
Figure 30. Schematic vision of an ontology network following (a) an incremental model and (b) an iterative model	102
Figure 31. Pyramid of the versions of the Waterfall Ontology Network Life Cycle Model.....	104
Figure 32. The 4-Phase Waterfall Ontology Network Life Cycle Model	105

Figure 33. The 5-Phase Waterfall Ontology Network Life Cycle Model	106
Figure 34. The 5-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model.....	106
Figure 35. The 6-Phase Waterfall Ontology Network Life Cycle Model.....	107
Figure 36. The 6-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model.....	107
Figure 37. Schematic vision of the Iterative-Incremental Model	109
Figure 38. Tasks for Ontology Requirements Specification.....	116
Figure 39. Decision tree for selecting model version and scenarios.....	127
Figure 40. Example of a scheduling template for the 4-Phase Waterfall Model with Scenarios 1 and 9	130
Figure 41. Example of a scheduling template for the 6-Phase Waterfall Model with Scenarios 2 and 3	130
Figure 42. Example of a scheduling template for the 6-Phase Waterfall Model with Scenarios 3, 4, 7, and 8	131
Figure 43. Example of the Ontology Localization Filling Card in gOntt.....	133
Figure 44. Example of the workflow and of some methodological guidelines for the Ontology Localization activity in gOntt	134
Figure 45. Screenshot of the gOntt plug-in	135
Figure 46. A specific plan generated by gOntt.....	135
Figure 47. Tasks for scheduling ontology development projects with gOntt.....	136
Figure 48. Decision tree for selecting the most appropriate model.....	137
Figure 49. Different types of ontological resource reuse	141
Figure 50. Example of a Java program.....	142
Figure 51. Example of a SPARQL query	143
Figure 52. Relations between intervals.....	144
Figure 53. The two usual representations of infinite temporal intervals (see footnote 77).....	145
Figure 54. Hasse diagram of mereological theories (from weaker to stronger, going uphill) [Varzi, 2003]	148
Figure 55. Activities for Reusing General Ontologies	150
Figure 56. Activities for Reusing Domain Ontologies as a Whole.....	174
Figure 57. Activities for the Ontology Statement Reuse	179
Figure 58. Workflow corresponding to the preliminary methodological guidelines for Ontology Requirements Specification	191
Figure 59. Workflow corresponding to the quasi-final methodological guidelines for Ontology Requirements Specification	192
Figure 60. Steps to be carried out for the ontology network life cycle establishment	198
Figure 61. Tasks for scheduling ontology development projects	203
Figure 62. Overview of the results presented in this thesis	211
Figure 63. Excerpt of the Competency Questions and answers in an Excel file.....	233
Figure 64. Excerpt of the Competency Questions in a Mind Map tool	233

Figure 65. Competency Questions groups	234
Figure 66. Competency Questions groups in detail	234
Figure 67. Excerpt of the Semantic Nomenclature Competency Questions in a Word file	242
Figure 68. Examples of Competency Questions in groups	243

English Summary

A new ontology development paradigm has started; its emphasis lies on the reuse and possible subsequent reengineering of knowledge resources, on the collaborative and argumentative ontology development, and on the building of ontology networks; this new trend is the opposite of building new ontologies from scratch. To help ontology developers in this new paradigm, it is important to provide strong methodological support.

This thesis presents some contributions to the methodological area of the Ontology Engineering field that we are sure will improve the development and building of ontologies networks, and thus,

- ❑ It proposes the *NeOn Glossary of Processes and Activities*, which identifies and defines the processes and activities potentially involved when ontology networks are collaboratively built.
- ❑ It defines a set of two *ontology network life cycle models*.
- ❑ It identifies and describes a collection of *nine scenarios for building ontology networks*.
- ❑ It provides some *methodological guidelines for performing the ontology requirements specification activity*, to obtain the requirements that the ontology should fulfil.
- ❑ It offers some *methodological guidelines for obtaining the ontology network life cycle for a concrete ontology network, as part of scheduling ontology projects*. Additionally, the thesis provides the technological support to these guidelines: a tool called gOntt.
- ❑ It also proposes some *methodological guidelines for the reuse of ontological resources* at two different levels of granularity: as a whole (general ontologies and domain ontologies) and using ontology statements.

Resumen

Un nuevo paradigma de desarrollo de ontologías ha comenzado. Dicho paradigma se basa en la reutilización y posterior reingeniería de recursos de conocimiento disponibles, en la colaboración y argumentación durante el desarrollo de las ontologías, y en la construcción de redes de ontologías, en contraposición a construir nuevas ontologías a la medida y desde cero. En este contexto, es importante proporcionar un soporte metodológico robusto para ayudar a los desarrolladores de ontologías.

Esta tesis doctoral presenta las siguientes contribuciones al área metodológica del campo de la Ingeniería Ontológica:

- ❑ Propone el *Glosario NeOn de Processos y Actividades*, que identifica y define los procesos y actividades que potencialmente están involucradas en el desarrollo colaborativo de redes de ontologías.
- ❑ Define un conjunto de dos *modelos de ciclo de vida para redes de ontologías*.
- ❑ Identifica y describe un conjunto de *nueve escenarios para construir redes de ontologías*.
- ❑ Propone las *guías metodológicas para desarrollar la actividad de especificación de requisitos*, con el objetivo de obtener los requisitos que la ontología debe satisfacer.
- ❑ Propone las *guías metodológicas para obtener el ciclo de vida de una red de ontologías, como parte de la planificación de proyectos ontológicos*. Además, la tesis proporciona el soporte tecnológico a dichas guías, la herramienta llamada gOntt.
- ❑ Propone las *guías metodológicas para la reutilización de recursos ontológicos* a dos niveles distintos de granularidad: reutilización de ontologías como un todo (ontologías generales y ontologías de dominio) y reutilización de *statements* ontológicos.

1. Introduction

Knowledge has become increasingly important to support intelligent process automation and solve problems collaboratively. Thus, the effective management of knowledge and shared data is one of the greatest challenges of current large organizations, both public and private.

Organizations manage crucial data and knowledge, namely, data about products, financial information, etc., that are recorded with the aim of supporting day-to-day company applications and decision making processes.

Additionally, an increase in the demand for resource sharing across different sites connected through networks has led to the evolution of data and knowledge-management systems from centralized to decentralized systems. Current decentralized systems still focus on data and knowledge as their main resource, and this implies to handle very large quantities of data and knowledge in different types of distributed archives. These data are normally structured, but sometimes they are very heterogeneous and, in most of the cases, not necessarily interoperable.

Nowadays, ontologies are used for making explicit the meaning of information and allowing the sharing of information. Thus, the solution to improving the management of data and knowledge is to create an ontology-based application that manages in a proper way all the knowledge of the organization; and this process involves modelling all this knowledge in an ontology network.

1.1. Thesis Context

Ontologies are widely used in (a) Knowledge Engineering, Artificial Intelligence and Computer Science, (b) applications related to knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database design and integration, bio-informatics, education, and (c) the Semantic Web and the Semantic Grid.

The word **ontology** was taken from Philosophy, where it means a systematic explanation of being. There are many definitions about what an ontology is and such definitions have changed and evolved over the years. However, one of the most well-known definition, which is the one used in the present thesis, is provided by Studer and colleagues [Studer et al., 1998]: “*An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group*”.

Research on methodologies for the Ontology Engineering field is in its “adolescence”. The 1990s and the first years of this new century 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 [Gómez-Pérez et al., 2003]. Until the mid-1990s, the ontology development process was an art rather than an engineering activity. Each development team usually followed their own set of principles as well as their own design criteria and phases for manually building the ontology [Gómez-Pérez et al., 2003]. Thus, the absence of common and structured guidelines slowed the development of ontologies within and between teams, the extension of any ontology, the possibility of ontologies of being reused in others, and the use of ontologies in final applications [Gómez-Pérez et al., 2003].

Until now, a large number of ontologies have been developed by different groups, under different approaches, and with different methods and techniques. Ontological Engineering [Gómez-Pérez et al., 2003] refers to the set of activities that concern the ontology development process, the

ontology life cycle, as well as the methodologies, tools and languages required for building ontologies. The progress made in building ontologies, which have to model some agreed-upon views of the reality, is impressive.

A series of methods and methodologies for developing ontologies from scratch have been reported in [Gómez-Pérez et al., 2003] and these 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, the first guidelines were proposed. These were based on the experience gathered in developing the Enterprise ontology and the TOVE (TOnto Virtual Enterprise) project ontology, both belonging to the enterprise modelling domain. In 1996, a method to build an ontology in the domain of electrical networks was presented. This method was part of the Esprit KACTUS project. The METHONTOLOGY methodology [Gómez-Pérez et al., 2003] appeared almost simultaneously, in 1996. In 1997, a new method was proposed for building ontologies based on the SENSUS ontology. And some years later, in 2001, the On-To-Knowledge methodology emerged from the project of the same name. However, all the aforementioned approaches have one important limitation: they do not regard the collaborative and distributed development of ontologies [Gómez-Pérez et al., 2003]. In this respect, in 2004, the DILIGENT methodology [Pinto et al., 2004] was proposed. This new methodology was intended to support domain experts in a distributed setting when they need to engineer and evolve ontologies.

In general, we can say that METHONTOLOGY [Gómez-Pérez et al., 2003] and On-To-Knowledge [Staab et al., 2001] are up to now the most complete methodologies for building ontologies from scratch. Both include guidelines for single ontology construction, ranging from ontology requirements specification to implementation.

1.2. Thesis Main Motivations

The development of ontologies in different national and international projects has revealed that there are different alternative ways or possibilities of building ontologies. For example, and just to name a few: in the Esperonto¹ project ontologies were built from scratch using METHONTOLOGY; in Knowledge Web Network of Excellence² the issue of the aligning and versioning of ontologies was tackled as well as the use of best practices or patterns in the context of W3C activities; in the SEEMP³ project a good ontology requirements specification document helped to find ontologies and consensual knowledge resources that were reused and re-engineered into ontologies; the SEKT⁴ project was focused on argumentative development of ontologies using the DILIGENT methodology; in the UMLS Project [Coté et al., 1994] the experiences gained while transforming the UMLS® Semantic Network into OWL ontology were described; within the UK PRODIGY and Drug Ontology Projects [Hodge, 2000] it was described the transformation of tangled hierarchies (for example, those derived from ambiguous "broader than / narrower than" thesauri in library science) into formal ontologies; and the WonderWeb project⁵, whose goal was to build a modular ontology library around DOLCE, can be seen as an initial attempt to build an ontology network.

Thus, it is not premature to affirm that a new ontology development paradigm is starting, whose emphasis is on (1) the reuse and subsequent reengineering of knowledge resources, (2) the

¹ <http://www.esperonto.net>

² <http://knowledgeweb.semanticweb.org>

³ <http://www.seemp.org/>

⁴ <http://www.sekt-project.com/>

⁵ <http://wonderweb.semanticweb.org/>

collaborative and argumentative ontology development, and (3) the building of ontology networks⁶, as opposed to custom-building new ontologies from scratch.

Furthermore, with the goal of speeding up the ontology development process, ontology practitioners are starting to reuse [Simperl, 2009] as much as possible (a) other ontologies such as DOLCE⁷, SUMO [Pease et al., 2002] and KOWIEN⁸, ontology modules [Cuenca-Grau et al., 2007]; (b) ontology statements and ontology design patterns [Gangemi, 2007; Presutti and Gangemi, 2008]; and (c) non-ontological resources [Jimeno-Yepes et al., 2009] such as thesauri, lexicons, DBs, UML diagrams and classification schemas (e.g., NAICS⁹ and SOC¹⁰) built by others and which already have some degree of consensus.

In this regard, a good example to mention is the case of the FAO (Food and Agriculture Organization)¹¹ fishery department. This department has several information and knowledge organization systems that facilitate and secure the long-term sustainable development and utilization of the world's fisheries and aquaculture. Such systems are developed and maintained by different people, who interact with experts and users worldwide. Each system has its own community, which currently constitutes a separate knowledge collective.

The FAO fishery systems manage different types of data and knowledge (e.g., fishing statistical data, GIS data, information on aquaculture, geographic entities, and description of fish stocks). Much of the data are 'structured', but they are very heterogeneous and, in most of the cases, not necessarily interoperable. In addition, knowledge is embedded in different resources that are not easily available.

In those cases in which different resources are not easily available, the solution is to create an ontology-based application that manages all the knowledge in the organization properly. This process implies first to model all this knowledge in an ontology network. In the case of FAO fishery department, the ontology should represent knowledge about different resources, for example, taxonomic classification of biological entities, ISSCFC¹² and HS classification of fisheries commodities¹³, ISSCAAP classification of species¹⁴ (with links to taxonomic classification of biological entities and ISSCFC commodities), FAO division of water areas¹⁵, large marine ecosystems (with links to FAO divisions)¹⁶, geopolitical ontology¹⁷ [Kim et al., 2009], exclusive

⁶ 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 meta-relationships such as mapping, modularization, version, and dependency relationships [based on Haase et al., 2006].

⁷ <http://www.loa-cnr.it/DOLCE.html>

⁸ Skill Ontology from the University of Essen, which defines concepts representing the competencies required to describe job position requirements and job applicant skills. Available at <http://www.kowien.uni-essen.de/publikationen/konstruktion.pdf>

⁹ North American Industry Classification System, which provides industry-sector definitions for Canada, Mexico, and the United States to facilitate uniform economic studies across the boundaries of these countries. Available at <http://www.census.gov/epcd/www/naics.html>

¹⁰ Standard Occupational Classification, which classifies workers into occupational categories (23 major groups, 96 minor groups, and 449 occupations). Available at <http://www.bls.gov/soc/>

¹¹ http://www.fao.org/index_en.htm

¹² International Standard Statistical Classification of Fishery Commodities: Divisions and Group. Available at ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/ANNEX_RII.pdf

¹³ Harmonized Commodity Description and Coding System. 2007 Edition. Available at http://www.wcoomd.org/ie/En/Topics_Issues/HarmonizedSystem/DocumentDB/TABLE_OF_CONTENTS_2007.html

¹⁴ International Standard Statistical Classification of Aquatic Animals and Plants. Version in use until 1999 available at: <ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/AnnexS1listISSCAAPold.pdf>

¹⁵ Handbook of Fishery Statistical Standards. Fishing Areas for Statistical Purposes. Available at <http://www.fao.org/fi/website/FIRetrieveAction.do?dom=ontology&xml=sectionH.xml>

¹⁶ <http://www.lme.noaa.gov/>

economic zones (with links to geopolitical ontology), ISSCFG classification of gear types¹⁸, ISSCFV classification of vessel type and size¹⁹, and stocks (with links to taxonomic classification of biological entities and FAO divisions).

Thus, developers realized that such an ontology network should not be developed entirely from scratch, but by reusing and possibly reengineering other ontologies, databases, XML schemas, thesauri, classification schemes, and other knowledge resources, as well as by taking into account good practices in the development. Additionally, they also believed that in large organizations the development and maintenance of these knowledge models that are the ontology networks should be performed by people with many different expertises (ontology engineers, software developers, aquaculture specialists, economists, biologists, etc.), who should collaborate in a coherent manner and have the possibility of managing different versions of the knowledge model.

Accordingly, it is logical to predict that the Semantic Web of the future will be characterized by using a very large number of ontologies embedded in ontology networks built collaboratively by distributed teams. Such ontology networks could include ontologies that already exist or that could be developed in different scenarios by reusing either other available ontologies or non-ontological but knowledge resources (thesauri, lexicons, text corpora, DBs, UML diagrams, etc.) [NeOn Consortium, 2006].

The amount of knowledge published on the Semantic Web (i.e., the number of ontologies and semantic documents available on-line) is rapidly increasing, having reached the critical mass required to enable the vision of a truly large scale, distributed and heterogeneous web of knowledge [d'Aquin et al., 2007d].

With this new vision of the ontologies and the Semantic Web, it is important to provide strong methodological support for the collaborative and context-sensitive development of ontology networks.

However, up to date, **there are no methodological approaches that help ontology developers to build large ontologies embedded in ontology networks in complex settings where distributed teams could collaboratively build ontologies by reusing and possibly reengineering knowledge resources, using alignments and having in mind the continuous evolution of the ontologies. On the other hand, the approaches available are not described with a user-oriented approach, but with a style that is more oriented to ontology researchers than to developers.** In summary, the methodologies available for building ontologies have at least four important limitations:

1. They lack detailed and clear guidelines for building ontologies by reusing and reengineering knowledge resources widely agreed upon in a particular domain.
2. They do not consider those scenarios in which the reuse is the key aspect.
3. They do not take into account different life cycle models related to the aforementioned reuse scenarios.
4. They do not explain the ontology building process with the same style and granularity than those methodologies for developing software, which would facilitate the understandability and adoption of such methodologies.

Additionally, there is no consensus about the definitions of activities and processes that potentially could be carried out when ontologies and ontology networks are developed. Consequently, there is a lack of standardization on the Ontology Engineering field terminology.

¹⁷ <http://www.fao.org/countryprofiles/geoinfo.asp?lang=en>

¹⁸ International Standard Statistical Classification of Fishing Gear. Available at <ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/AnnexM1fishinggear.pdf>

¹⁹ International Standard Statistical Classification of Fishery Vessels by Vessel Types, in use until 1995. 1984. Available at <ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/annexLII.pdf>

Our aim is to create the *NeOn Methodology for building ontology networks*, covering the drawbacks of the existing methodologies while, at the same time, benefiting from the advantages included in them.

Thus, the main goal of this thesis is to present a methodology for building networks of ontologies. The main principles that guide the construction of such a methodology are

1. It should define each process or activity precisely; it should 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.
2. It should be presented in a prescriptive way and not only oriented to researchers, to facilitate a promptly assimilation by software developers and ontology practitioners.
3. It should be general enough in the sense that it should help software developers and ontology practitioners to build networks of ontologies with any ontology development tool (NeOn Toolkit²⁰, Protégé²¹, Top Braid Composer²², etc).

1.3. Thesis Main Objective and Tasks

Methodological frameworks are widely accepted in different mature fields, like Software Engineering and Knowledge Engineering. Such methodological frameworks involve the following issues: development process, life cycle models and life cycle, and the methods, techniques and tools to be used. Thus, the methodologies for developing ontologies should also involve the issues aforementioned.

In this context, the main objective of this thesis is to create the **NeOn Methodology** framework, a framework that will support different aspects of ontology network development as well as the reuse and the dynamic evolution of networked ontologies in distributed environments where knowledge is introduced by developers (domain experts, ontology practitioners) at different stages of the ontology development process.

This objective will be achieved through investigating the following tasks:

- Task 1. Identification and description of the *research methodology* used in this thesis.
- Task 2. Identification and definition of the *development process* and *life cycle of ontology networks*.

In order to provide the methodological support for the collaborative and context-sensitive development of ontology networks, we have taken as a starting point some IEEE standard definitions [IEEE, 1990; IEEE, 1997] related to the development process, life cycle models, and life cycle in Software Engineering and we have adapted them to the specific characteristics of the development of single ontologies and ontology networks.

- Task 2.1. Creation of the NeOn Glossary of Processes and Activities.
- Task 3. Creation of the *NeOn Methodology for building ontology networks*. This methodology will include methodological guidelines (methods, techniques and tools) for carrying out the processes and activities identified and defined in the ontology network development process.
 - Task 3.1. Creation of the NeOn Methodology framework, as a scenario-based methodology, for building ontology networks.

²⁰ <http://www.neon-toolkit.org/>

²¹ <http://protege.stanford.edu/>

²² http://www.topquadrant.com/products/TB_Composer.html

- Task 3.2. Creation of the methodological guidelines within the general methodological framework for the following activities and processes:
 - Ontology requirements specification activity.
 - Ontology development project schedule activity.
 - Ontological resources reuse process.

It is worth mentioning that within our research group there are three theses, one already presented and the other two in progress, that deal with the NeOn Methodology framework though they cover different issues from those tackled here. These are

- Collaboration and dynamism [Palma, 2009]
- Reuse and reengineering non-ontological resources [Villazón-Terrazas, in progress]
- Localization [Espinoza, in progress]

1.4. Thesis Main Contributions

The six main contributions of this thesis are the following:

1. The *NeOn Glossary of Processes and Activities*, which identifies and defines the processes and activities potentially involved when ontology networks are built. Part of this work was carried out in collaboration with partners concerned with the NeOn project, as explained in Chapter 5. This glossary is included in Chapter 5.
2. A collection of *nine scenarios for building ontology networks*, included in Chapter 6.
3. A set of two different *ontology network life cycle models*, which is sufficient to allow ontology practitioners to develop ontology networks covering the different possibilities. This set of models is described in Chapter 7.
4. *Guidelines for performing the ontology requirements specification activity*, to obtain the requirements that the ontology should fulfil. These guidelines are explained in Chapter 8.
5. *Guidelines for obtaining the ontology network life cycle for a concrete ontology network*, which is a crucial part of scheduling ontology projects. Additionally, we provide the technological support to these guidelines in the form of a plug-in called gOntt. Both contributions are included in Chapter 9.
6. *Guidelines for the reuse of ontological resources* described in Chapter 1.

1.5. Thesis Structure

The thesis is structured as follows:

- Chapter 2 (*State of the Art*) deals with the state of the art of the development process and life cycle models in different fields (Software Engineering, Knowledge Engineering and Ontology Engineering). This chapter also includes the state of the art of methodologies for building ontologies and some methodological issues related to ontology requirements specification and ontological resource reuse.
- Chapter 3 (*Work Objectives*) presents the main objectives and assumptions of this thesis.
- Chapter 4 (*Research Methodology*) explains the research methodology followed.

- Chapter 5 (*Glossary of Processes and Activities*) presents the NeOn Glossary of Processes and Activities, which identifies and defines the activities and processes potentially involved in ontology network construction.
- Chapter 6 (*Scenarios for Building Ontology Networks*) deals with the identification and definition of different scenarios for building ontology networks collaboratively, with special emphasis on reuse, reengineering and merging.
- Chapter 7 (*Life Cycle Models for Ontology Networks*) provides the definition of several ontology network life cycle models according to those defined in the Software Engineering field.
- Chapter 8 (*Ontology Requirements Specification*) presents the proposed methodological guidelines for carrying out the ontology requirements specification activity. This chapter also includes three examples of how to carry out this activity in different use cases.
- Chapter 9 (*Planning and Scheduling: Obtaining the Ontology Network Life Cycle*) provides the groundings for scheduling ontology network development projects and describes the plug-in gOntt and the methodological guidelines for the scheduling activity.
- Chapter 10 (*Ontological Resource Reuse*) presents the methodological guidelines proposed for carrying out the ontological resource reuse process. Specifically, such guidelines are proposed for the general or common ontology reuse, the domain ontology reuse, and the ontology statement reuse.
- Chapter 11 (*Experimentation*) includes action research and controlled experiments that show the evaluation of the results presented in this thesis.
- Chapter 12 (*Conclusions and Future Work*) offers the main conclusions of this work, emphasizing its main contributions. The chapter also lists future lines of work that might be performed in the field of ontology networks development.

Finally, the thesis includes the bibliographic references used for its elaboration and 8 annexes that provide information relevant to the thesis.

2. State of the Art

2.1. Introduction

This chapter presents a summary of the state of the art that is relevant for this thesis. Bearing in mind that ontologies are part of software products and that sometimes ontologies are considered a kind of software, we provide, in Section 2.2, a glossary of the terms most used in the present thesis, taken from the Software Engineering field. Because this thesis deals with development process, life cycle models and life cycles, Sections 2.3 and 2.4 focus on development process and life cycle models in the Software Engineering and Knowledge Engineering fields, respectively. Additionally, Section 2.5 includes how the most well-known methodologies in the Ontology Engineering field (METHONTOLOGY [Gómez-Pérez et al., 2003], On-To-Knowledge [Staab et al., 2001], and DILIGENT [Pinto et al., 2004]) tackle development process and life cycle models. Section 2.5 gives also an overview of some seminal works on ontology requirements specification and ontological resource reuse, issues important here since this thesis provides guidelines for specifying ontology requirements and reusing ontological resources. Finally, Section 2.6 concludes with a summary of the main gaps in methodologies for building ontologies in the current state of the art.

In a nutshell, we will show that none of the methodological approaches available can help ontology developers to build large ontologies embedded in ontology networks by reusing and possibly reengineering knowledge resources.

2.2. Basic Terminology from Software Engineering

This section includes several terms defined in Software Engineering that are crucial for this thesis. Those terms have been extracted from the following IEEE documents:

- IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990) [IEEE, 1990].
- IEEE Guide for Developing Software Life Cycle Processes (IEEE Std 1074.1-1995) [IEEE, 1995].
- IEEE Standard for Developing Software Life Cycle Processes (IEEE Std 1074-1997) [IEEE, 1997].

The terminology is presented in an incremental way, from the simplest term to the most complex one.

- A **task** can be defined as the smallest unit of work subject to management accountability. Another definition of **task** [IEEE, 1995] can be a well-defined work assignment for one or more project members. Related tasks are usually grouped to form activities.
- An **activity** [IEEE, 1997] is a defined body of work that is to be performed, including its required input and output information.
- A **process** [IEEE, 1990] is a sequence of steps performed for a given purpose. A process is composed of activities.
- The **software development process** [IEEE, 1990] is the process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing

the design in code, testing the code, and sometimes, installing and checking out the software for operational use.

- A **software life cycle model** [IEEE, 1997] is the framework, selected by each organization, on which to map the activities identified in the software development process to produce the software life cycle. For example, waterfall model [Royce, 1970], evolving prototyping model [Davis et al., 1988], etc.
- The **software life cycle** [IEEE, 1990] is defined as the period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. Furthermore, the software life cycle [IEEE, 1997] is the project-specific sequence of activities created by mapping the activities identified in the software development process onto a selected software life cycle model.

Throughout the literature, the terms *methodology*, *method*, *technique*, *process*, *activity*, etc. are used indiscriminately [de Hoog, 1998]. To make clear the use of these terms, in this thesis²³ we have adopted the IEEE definitions for such terms.

The IEEE [IEEE, 1990b] 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”. Methods and techniques are parts of methodologies. A **method** [IEEE, 1990b] is a set of “orderly processes or procedures used in the engineering of a product or performing a service”. A **technique** [IEEE, 1990] is “a technical and managerial procedure used to achieve a given objective”. De Hoog [de Hoog, 1998] explores relationships between methodologies and methods. According to this author, methodologies and methods are not the same because “methodologies refer to knowledge about methods”. Methodologies state “what”, “who” and “when” a given activity should be performed. Greenwood [Greenwood, 1973] also explores the differences between methods and techniques. A method is a general procedure, whereas a technique is the specific application of a method and the way in which the method is executed. Several techniques are used for applying a given method, and each technique specifies what means should be used to execute the method.

Methods and techniques are strongly related because both are used to carry out tasks inside the different processes a methodology comprises. The IEEE defines a **process** [IEEE, 1995] as a “function that must be performed in the software life cycle. A process is composed of activities”. An **activity** [IEEE, 1995] is “a constituent task of a process”. Another definition of the latter [IEEE, 1997] is that an **activity** 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 [IEEE, 1995] 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 [Gómez-Pérez et al., 2003], where it can be observed that a methodology is composed of methods and techniques. Methods, in turn, are composed of processes and are detailed with techniques. Processes are composed of activities. And finally, activities are made up of groups of tasks.

²³ This section is a summary taken from [Gómez-Pérez et al., 2003].

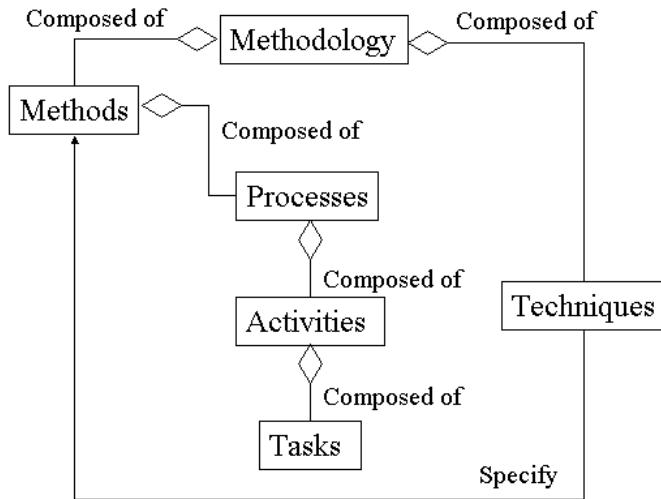


Figure 1. Graphical representation of terminological relationships in methodologies [Gómez-Pérez et al., 2003]

2.3. State of the Art in Software Engineering

This section summarizes the main results of the development process and life cycle models in the Software Engineering field.

2.3.1. Development Process

A **software development process** is a process that transforms a user's need into a computable product that satisfies the user's need.

This section briefly summarizes three well-known Software Engineering development processes: IEEE development process [IEEE, 2006] (Section 2.3.1.1), ISO development process [ISO, 2006] (Section 2.3.1.2), and RUP development process [Kruchten, 2000] (Section 2.3.1.3).

2.3.1.1. IEEE development process

The 69 activities identified in the software project development process are grouped administratively, according to IEEE [IEEE, 2006], into five main activity groups. The full set of activities covers the entire software life cycle, from concept exploration through the eventual retirement of the software system. The main activity groups [IEEE, 2006], also shown in Table 1, are the following:

1. Project Management Activity Group [IEEE, 2006]. This group includes activities that initiate, monitor, and control a software project throughout its life cycle. For example, Project Monitoring and Control activities are used to track and manage the project. During these activities, the actual project performance is tracked, reported, and managed against the planned performance. In this group special consideration is given to risk management. The group is divided into three subgroups: Project Initiation, Project Planning, and Project Monitoring and Control.

2. Pre-Development Activity Group [IEEE, 2006]. This group includes the activities that explore and allocate system requirements before the software development can begin. For example, the feasibility study serves to analyse the idea or need, the potential approaches, and all the life cycle constraints and benefits. Justification for each recommendation should be fully documented and formally approved by all the organizations concerned (including the user and the developer). This

group is divided into three activity subgroups: Concept Exploration, System Allocation, and Software Importation.

3. Development Activity Group [IEEE, 2006]. This group includes the activities performed during the development and enhancement of a software project. For example, Design activity has as main goal to develop a coherent, well-organized representation of the software system that meets the software requirements. This group is divided into three subgroups: Software Requirements, Design, and Implementation.

4. Post-Development Activity Group [IEEE, 2006]. This group includes the activities that install, operate, support, maintain, and retire a software product. For example, the Installation activity consists in the transportation and installation of a software system from the development environment to the target environment(s); and the Operation and Support activities involve the user's operation of the system and ongoing support. Support includes providing technical assistance, consulting the user, and recording user's support requests by maintaining a Support Request Log. This group is divided into four subgroups: Installation, Operation and Support, Maintenance, and Retirement.

5. Support Activity Group [IEEE, 2006]. This group includes activities that are necessary to assure the successful completion of a project, but that are considered supporting activities rather than activities directly oriented to the development effort. In other words, the Support Activity Group includes the set of activities necessary to ensure that a system fulfils its original requirements and any subsequent modifications to those requirements. For example, the Documentation Development activity for software development and usage has as a purpose to plan, design, implement, edit, produce, distribute, and maintain the documents that are needed by developers and users. This group is divided into four subgroups: Evaluation, Software Configuration Management, Documentation Development, and Training.

Section	Clause	Activity groups
Project Management	A.1	Project Initiation Project Planning Project Monitoring and Control
Pre-Development	A.2	Concept Exploration System Allocation Software Importation
Development	A.3	Software Requirements Design Implementation
Post-Development	A.4	Installation Operation and Support Maintenance Retirement
Support	A.5	Evaluation Software Configuration Management Documentation Development Training

Table 1. Activity groups from IEEE [IEEE, 2006]

2.3.1.2. ISO development process

The international standard ISO 12207 [ISO, 2006] establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a software product or service and during the supply, development, operation, and maintenance of software products. Each process within the life cycle has a set of outcomes associated with it.

This international standard also provides a process that can be employed for defining, controlling, and improving software life cycle processes.

The standard emphasises two basic principles: *modularity* and *responsibility*.

- Modularity means processes with minimum coupling and maximum cohesion.
- Responsibility means establishing a responsibility for each process, facilitating the application of the standard to projects in which many people can be involved.

This international standard [ISO, 2006] groups the activities that may be performed during the life cycle of a software system into seven process groups. Each of the processes within those groups is described in terms of its purpose and desired outcomes and lists activities and tasks that need to be performed to achieve those outcomes. The purposes and outcomes of these processes constitute a process reference model.

The seven process groups, shown in Figure 2, are the following:

1. **Agreement Processes** define the activities necessary to establish an agreement between two organizations. These processes are the acquisition process and the supply process.
2. **Project-Enabling Processes** manage the organization's capability to acquire and supply products or services through the initiation, support and control of projects.
3. **Project Processes**. There are two categories of Project Processes: project management processes and project support processes. The Project Management Processes are used to plan, execute, assess and control the progress of a project. The Project Support Processes support specialized management objectives, which refer to the management of any undertaking, ranging from a complete organization down to a single life cycle process and its tasks.
4. **Technical Processes**, which include eleven processes, are used to define the requirements for a system, to transform such requirements into an effective product, to permit consistent reproduction of the product where necessary, to use the product, to provide the required services, to sustain the provision of those services, and to dispose of the product when it is retired from service.
5. **Software Implementation Processes**, which include seven processes, are used to produce a specified system element (software item) implemented in software. Those processes transform specified behaviour, interfaces and implementation constraints into the implementation actions resulting in a system element that satisfies the requirements derived from the system requirements.
6. **Software Support Processes**, which include eight processes, provide a specific focused set of activities for performing a specialized software process. A supporting process assists the Software Implementation Process as an integral part with a distinct purpose, contributing to the success and quality of the software project.
7. **Software Reuse Processes**, which include three processes (domain engineering process, reuse program management process, and reuse asset management process) that support an organization's ability to reuse software items across project boundaries.

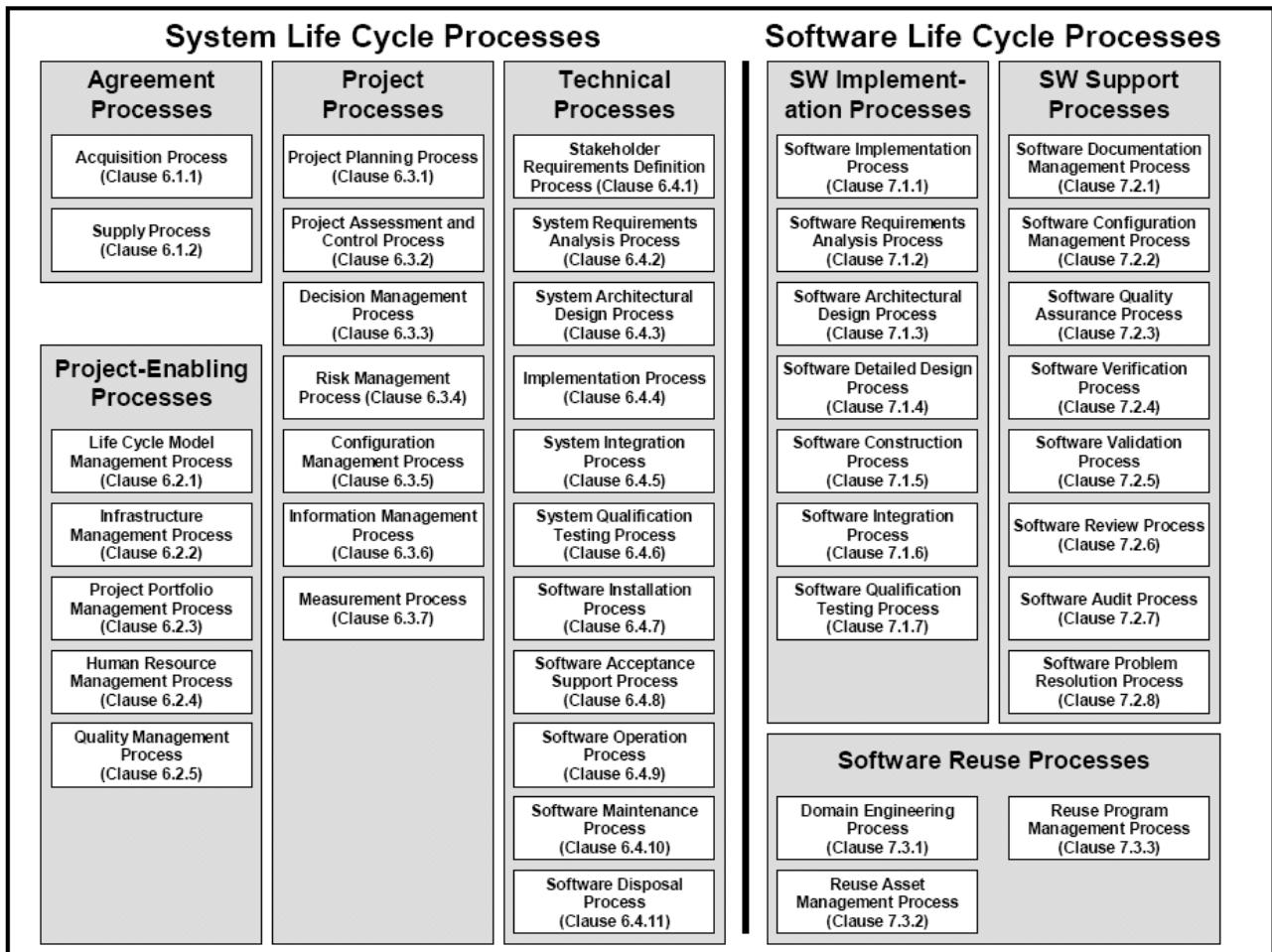


Figure 2. Process groups in the software development process [ISO, 2006]

2.3.1.3. RUP development process

One of the best-known methodologies is the Rational Unified Process (RUP) [Kruchten, 2000]. This methodology is an iterative software development framework originally created in the late 1980s by the Rational Software Corporation, which became a division of IBM in 2002. Since then, the methodology is sometimes referred to as IBM RUP process.

RUP is not a single concrete process that would prescribe definite steps for the software development but rather an adaptable framework. It is intended to be tailored and customized by development organizations and project teams, who are expected to select the elements of the process appropriate for their needs. RUP is also a fully-fledged software product in its own right (it includes a hyperlinked knowledge base with sample artefacts and detailed descriptions for many different types of activities).

RUP is based on six key principles for the so-called business-driven development:

1. *Adapting the process.* There is not a single ‘typical’ software development process; organizations must have options to tailor the process to their needs. In this regard, RUP provides pre-configured process templates for small, medium and large projects, which can be used for easier adoption.
2. *Balancing stakeholder priorities.* The key principle of this tenet is to address the traditional requirements gathering and analysis and to see these activities as a part of a broader relationship to business goals and stakeholder interests, which are often contradictory.

3. *Collaborating across teams.* Since RUP is built upon an assumption of iterative and incremental development, the direct effect of this assumption is that a wide range of developers will be involved.
4. *Demonstrating the added value iteratively.* As above, software projects are assumed to be delivered in increments. The increment, which includes the value of the past iteration, is used as a measure of the project progress. That increment is also used to encourage feedback about the direction of the project from stakeholders. All this allows the provision of a timely feedback and rectification of identified shortcomings.
5. *Elevating the level of abstraction.* RUP motivates the use of reusable assets such as software patterns. Its importance lies in that it prevents the engineers from ‘doing hacks’; i.e., going directly from the requirements to custom-made software code. A higher level of abstraction also allows discussions on different architectural levels, which are, in practice, often accompanied by visual representations, for example using UML.
6. *Focusing continuously on quality.* Quality checks are not only at the end of each iteration but also form a continuous activity, often performed daily and supported by the entire team. Automating test scenarios (scripts) are increasingly popular for coping with the increasing amount of tests due to the iterative development.

The RUP development process recognizes the following four broad phases:

- ❑ *Inception phase.* In this initial phase, the business case (including the context of the product and its success factors) is established. At least two aspects are crucial for this phase: (1) an initial use case and (2) an initial project description comprising the key requirements, constraints and features. The purpose of this phase is to appreciate the scope of the definition, the understanding of the requirements and their satisfaction in the proposed use case(s), and the overall credibility of the project.
This is also the phase where one usually defines base milestones related to satisfying the key aspects discussed in this phase.
- ❑ *Elaboration phase.* This is where the project starts taking its shape; the key activity in this step is the problem domain analysis, and the key outcome is usually the initial architecture. The purpose of this phase is to extend and elaborate the initial use case(s) and use case descriptions that start driving the formulation of key architectural modules.
- ❑ *Construction phase.* In this phase the main focus shifts from the analysis to the synthesis and development of components and other features of the system being designed. This is the phase where the bulk of the coding, modelling and also testing take place. In larger projects, several construction iterations may be developed in an effort to divide the use cases into manageable segments. Thus, two key outcomes come from this phase: (1) a realistic project plan (the familiar Gantt chart is a good example of this) and (2) a suite of demonstrable prototypes. It is also in this phase where the first software releases are produced; hence, this phase is often associated with the operational capability milestones.
- ❑ *Transition phase.* Here the project is moving from the development organization to the end user(s). The activities of this phase include training end users and maintainers and beta testing the system to validate it against end users’ expectations. The product is also checked against the quality level set in the inception phase. If the product does not meet this level or the end users’ standards, the entire cycle may begin again or may return to the appropriate earlier phase.

2.3.2. Life Cycle Models

The life of a software product can be modelled by a life cycle model consisting of stages [ISO, 2006].

Software life cycle models define how to develop and maintain a software system, i.e., they describe different ways of organizing the activities to be carried out [Pfleeger, 2001]. Software life cycle models are abstractions of the phases or states that a software product passes through along its life. These models could also determine the order of the phases and establish the transition criteria between phases.

It is well known that there is no a life cycle model valid for all the software development projects [Pfleeger, 2001], and that each software life cycle model is appropriate for a concrete project, depending on the following characteristics:

- Software organization culture. The organization is enterprising or conservative (that is, it likes to assume or not risks in projects).
- Previous experience of the software development team.
- Software application area.
- Comprehension and volatility of the software requirements.

Once a particular software life cycle model has been chosen for the software project, then the concrete software life cycle can be obtained. The **software life cycle** describes the software product life from its conception until its implementation, deliver, use, and maintenance. That is, the software life cycle describes the states that a software product passes through from its conception until its death.

This section briefly summarizes the most commonly used and well-known software life cycle models. Such models are based on the literature, but it was difficult to summarize them because different authors explain some of the models in a different way. For this reason, we include in this section our own understanding of the literature available.

- The sequential models, such as the waterfall (Section 2.3.2.1) and the 'V' model (Section 2.3.2.2).
- The gradual models, such as the incremental development (Section 2.3.2.3) and the iterative development (Section 2.3.2.4).
- The prototype-based models, such as evolutionary prototyping model (Section 2.3.2.5) and rapid throwaway prototyping approach (Section 2.3.2.6).
- The spiral model (Section 2.3.2.7).
- The extreme programming model (Section 2.3.2.8).
- The reuse components model (Section 2.3.2.9).

2.3.2.1. Waterfall life cycle model

The pure waterfall life cycle model was defined by Royce [Royce, 1970] to help cope with the growing complexity of the development of software projects. The use of such a model encourages the developer to specify what the system is supposed to do (i.e., to define the requirements) before building the system and to plan how components are going to interact (i.e., designing) before building the components (i.e., implementing). This model enables project managers to track progress more accurately and demands that the development process generate a series of documents that can later be used to test and maintain the system. The use of this model reduces development and maintenance costs for all of the above reasons [Davis, 1988].

The pure waterfall life cycle model [Pfleeger, 2001] is divided into sequential phases (analysis, design, implementation and testing) and represents the stages as a waterfall, from a particular stage through the following one. The model represents the stages of a project, starting with the requirements plan, such as the problem that the user has or thinks he has, through the analysis and definition of requirements, design, coding, testing, and operation to maintenance. The result of each phase is one or more documents that are approved ('signed-off') [Sommerville, 2007]. The following phase should not start until the previous one has finished, as shown in Figure 3. For example, the development team should not begin with the activities related to the system design until all the requirements have been identified and analysed with the purpose of checking their integrity and consistency and finally documenting them.

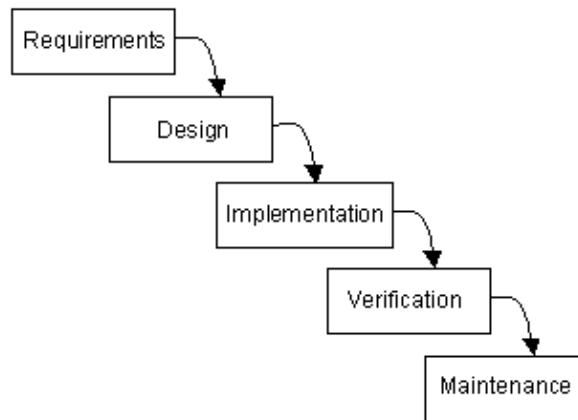


Figure 3. Pure waterfall life cycle model

In practice, the waterfall life cycle model has a lineal order in the transitions between phases, with one review for each phase. Thus, this model permits to backtrack to previous phases if errors or mistakes are detected [Sommerville, 2007]. Figure 4 shows the usual waterfall life cycle model.

The waterfall life cycle model assumes that the requirements are completely known and without ambiguities at the beginning of the development process. Such requirements do not change along the project. Therefore, this model should only be used when the requirements are well understood and unlikely to change radically during system development. This model is still used, particularly when the software project is part of a larger systems engineering project [Sommerville, 2007].

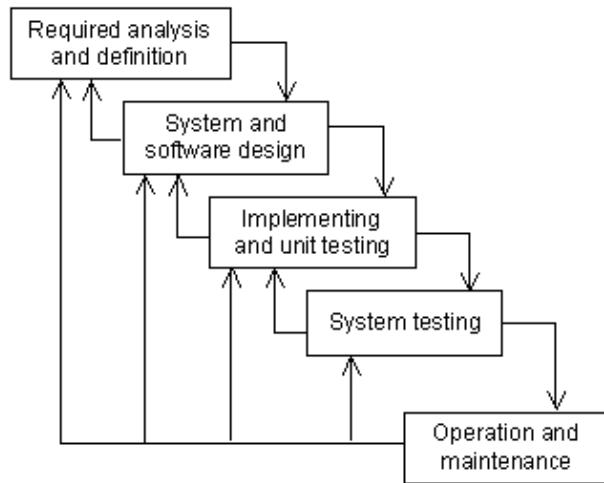


Figure 4. Waterfall life cycle model [Sommerville, 2007]

The advantages of the waterfall model are the following:

- The model is easy to understand.
- The model facilitates to plan a project according to this sequential approach.
- Documentation is produced at each phase.

The main inconveniences with this type of model are

- It does not reflect clearly how the software code is developed [Pfleeger, 2001] because real projects rarely follow the sequential flow that the model proposes.
- It is inflexible with respect partitioning the project into distinct stages. Commitments must be made at an early stage of the process, which makes it difficult to respond to changes in the customers' requirements [Sommerville, 2007].
- The requirements can be obsolete at the end of the project.
- No software product is shown to the user until the end of the project.

2.3.2.2. 'V' life cycle model

The most commonly known variant of the waterfall model, the 'V' life cycle model [Pfleeger, 2001; German Ministry of Defense, 1992], is a sequential path of execution of processes where each phase must be completed before the next phase begins. This model explicitly includes the observation that the result of each development task must be verified in a corresponding test task. This life cycle model gives emphasis to the validation of the products in each phase.

Figure 5 shows the graphical representation of the 'V' life cycle model. The symmetry between the left and right sides of the model reflects the relationship between the steps on the left and the steps on the right. The system definition generated on the left is ultimately used to verify the system on the right. The connections between left and right are indicated by the arrows that cross the "V", showing how plans developed on the left drive the process on the right. These connections provide continuity between the beginning and end of the project development and ensure that the engineers are focused on the completion of the project from the beginning [US Department of Transportation, 2007].

The relation between the right and left sides of the V-model, shown in Figure 5 implies that if problems are found during the verification and validation, then the left part of the 'V' can be performed again to solve the problems and improve the requirements, the design and the code before redoing the tests.

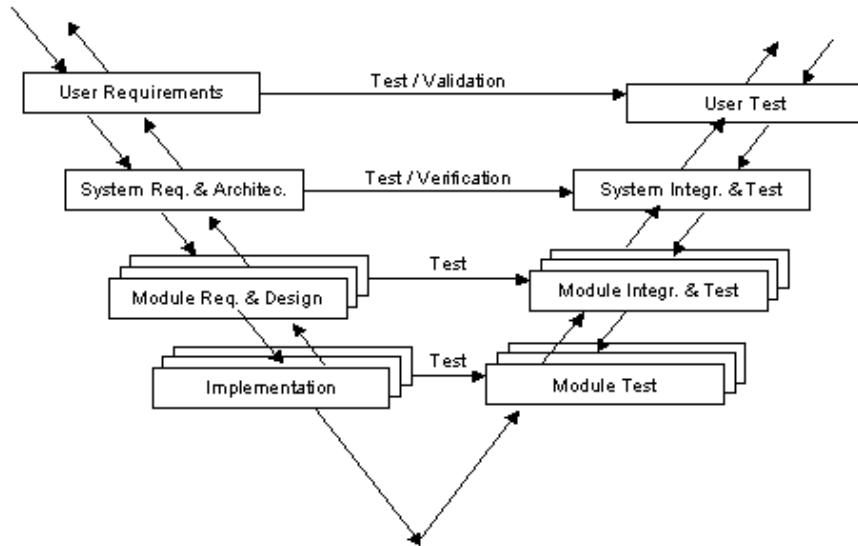


Figure 5. 'V' life cycle model

2.3.2.3. Incremental development life cycle model

Incremental development model [Sommerville, 2007] involves producing and delivering the software in increments rather than in a single package.

In the incremental development model [Pfleeger, 2001], the system, specified in the requirement document, is divided into functional subsystems, which are completely developed in different cycles. Each cycle follows the waterfall model. The system development starts with a small functional subsystem that is fully developed. In each cycle a functional subsystem is developed and a new version is delivered to the user at the end of each cycle. Following this model, the software development process begins with the most essential functionalities and continues and ends with the less important ones.

In this model, the complete system is built gradually through different iterations. The model can be seen as a repetition of the waterfall model for each functional subsystem. Therefore, in each iteration each subsystem passes through the requirements, design, implementation, and testing phases.

Incremental development implies that we understand upfront most of our requirements and simply choose to implement them in subsets of increasing capability. With the incremental development life cycle model, the risk of developing the wrong element is reduced by breaking the project into a series of small subprojects (increments).

In the incremental development model, as in the waterfall one, the problem and the overall requirements of the final product are well-known at the start of the development. Here, however, a limited set of requirements is allocated to each increment, and with each successive release more requirements are addressed until the final release satisfies all requirements.

The main advantages of this approach are

- It accelerates the delivery of customer services and user's engagement with the system [Sommerville, 2007].
- It produces an operational system more rapidly, thus reducing the possibility that the user's needs change during the development process [Davis, 1988].

One risk of this approach is that the first releases address such a limited set of requirements that customers could be dissatisfied; on the other hand, one benefit is that wrong or missing requirements can be corrected in time.

There are some types of systems for which the incremental development and delivery is not the best approach. These are very large systems in which development may concern (a) teams working in different locations, (b) some embedded systems where the software depends on hardware development, and (c) some critical systems where all the requirements must be analysed to check for interactions that may compromise the safety or security of the system [Sommerville, 2007].

2.3.2.4. Iterative development life cycle model

In the iterative development model, as in the waterfall one, the problem and the overall requirements of the final product are well-known at the start of the development. In iterative models, however, a limited set of requirements is allocated to each increment, and with each successive release more requirements are addressed until the final release satisfies all requirements.

The iterative development [Pfleeger, 2001] also divides the system into small parts, as the incremental development (Section 2.3.2.3). However, instead of building and delivering a new and complete subsystem in each cycle, the iterative model proposes building and delivering in each cycle a system with all the subsystems partially developed, being the functionalities of each subsystem improved in each new cycle. At the end of the cycle, an improved new version of the complete system is delivered.

This life cycle model has as purpose to reduce the risk between the user's needs and the final product; it consists in the iteration of several waterfall life cycles. At the end of each iteration, an improved or refined version of the software product is delivered.

2.3.2.5. Evolutionary prototyping life cycle model

The evolutionary prototyping model [Davis, 1988] is based on the assumption that it is often difficult to know all the system requirements at the beginning of a project, and that the requirements can change during the project life. This assumption is the main difference of this model with respect to the previous ones, in which requirements were supposed to be known at the beginning of the project. Following an evolutionary prototyping life cycle model [Davis, 1988], the developers construct a partial implementation of the system that meets the requirements known. The prototype is rigorously developed by incorporating the best requirements understood. The prototype is then evaluated and used by its intended users and the requirements are refined according to the evaluation.

This kind of model is used (1) when the user does not know exactly what he wants, or (2) when what the user wants is not understood by the development team, or (3) when the feasibility of the solution is not clear.

2.3.2.6. Rapid throwaway prototyping life cycle model

The rapid throwaway prototyping approach [Davis, 1988] addresses the issue of ensuring that the software product being proposed really meets the users' needs. The approach involves constructing a "quick and dirty" partial implementation of the system prior to (or during) the requirements stage. The potential users use and evaluate this prototype for a period of time and supply the developers with feedback concerning its strengths and weaknesses. This feedback is then used to modify the software requirements specification to reflect the users' real needs. At this point, the developers can proceed with the actual system design and implementation with the confidence that they are building the "right" system (except in those cases where the users' needs evolve).

The objective of throwaway prototyping is to understand [Sommerville, 2007] the software requirements. The development should start with requirements that are not well understood or are doubtful in order to find out more about them. Requirements that are straightforward may never need to be prototyped [Sommerville, 2007].

Throwaway prototypes have a very short lifetime. It is possible to change them rapidly during development, but long term maintainability is not required. Poor performance and reliability may be acceptable in a throwaway prototype so long as it helps everyone understand the requirements [Sommerville, 2007].

This kind of model is used (1) when the user does not know exactly what he wants, (2) when what the user wants is not understood by the development team, (3) when the feasibility of the solution is not clear, or (4) when the development team should learn a new technology.

2.3.2.7. Spiral life cycle model

The spiral model [Pfleeger, 2001] was designed to include the best features from the waterfall and prototyping models, but it introduces a new component: the concept of risk, which appears because of the uncertainties in the requirements and its assessment. This life cycle model is based on the idea of showing the historical cycles of the Earth's geology [Juristo and Pazos, 1993].

The spiral life cycle model [Pfleeger, 2001; Boehm, 1988; Boehm, 1976], shown in Figure 6, proposes combining the development activities with the risk management activities to minimize and control the risk. This model divides the software engineering space into four quadrants: management planning, formal risk analysis, engineering, and customer assessment. The development of the system is divided into n-cycles. Each cycle consists of waterfall phases. The result of each cycle is a prototype, and the development cycle continues until the final version of the product is reached. With each iteration around the spiral (beginning at the center and working outward), more complete versions of the system are progressively built. Furthermore, user feedback is considered at the output of each cycle, and new additional features are included into the next cycle.

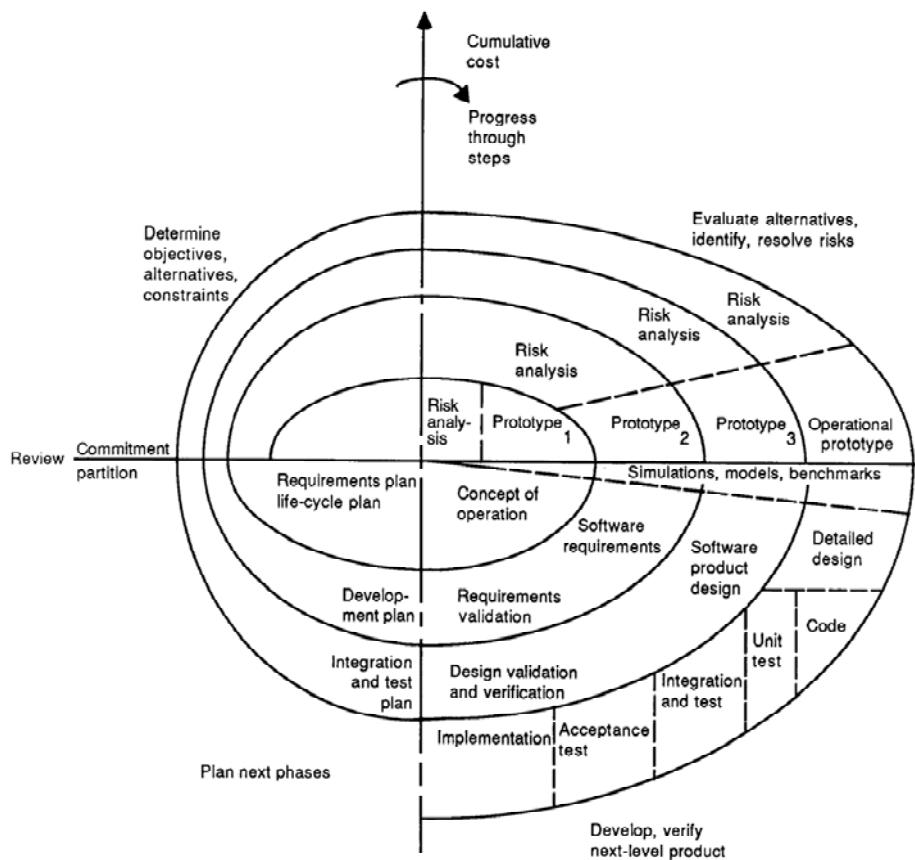


Figure 6. Spiral life cycle model [Boehm, 1976]

This life cycle model consists of the following four repetitive phases:

- Planning: it is based on the requirements.
- Risk analysis: the decision of continuing with the development is taken according to the requirements.

Risk analysis is included as a step in the development process. This analysis serves as a way of evaluating each version of the system to determine whether the development should continue. If the customer decides that any identified risk is too great, the project may be halted. For example, if a substantial increase in cost or completion time is identified during one phase of risk analysis, the customer or the developer may decide that it does not make sense to continue with the project, since the increased cost or lengthened timeframe may make the continuation of the project impractical or unfeasible.

Risk management [Sommerville, 2007] is increasingly seen as one of the main jobs of project managers. It involves, on the one hand, anticipating risks that might affect the project schedule or the quality of the software being developed and, on the other hand, taking action to avoid these risks. The results of the risk analysis should be documented in the project plan along with an analysis of the consequences of a risk if this appears.

There are three related categories of risk:

- Project risks are risks that affect the project schedule or resources (e.g., the loss of an experienced designer).
- Product risks are risks that affect the quality or performance of the software being developed (e.g., the failure of a purchased component to perform as expected).

- Business risks are risks that affect the organization developing or procuring the software (e.g., when a competitor introduces a new product in the market).
- Implementation: development of a prototype based on the requirements.
- Evaluation: the user evaluates the prototype. If the user is satisfied, the project ends; if not, new requirements are included in the following iteration.

In a nutshell, the main difference between the spiral model and other life cycle models is the explicit recognition of risk. Informally, risk simply means that something can go wrong [Sommerville, 2007]. This model has an approach driven by risk. In each cycle, a risk analysis is carried out to identify possible situations that may cause the project failure and to focus first on the software aspects with more risk.

2.3.2.8. Extreme programming life cycle model

The Extreme Programming (XP) [Beck and Andres, 2004] is a lightweight methodology for software development, which emphasizes customer involvement and team work and targets small to medium sized teams that build risky software projects with dynamic requirements.

The key aspects of XP are customer satisfaction, daily releases, and tests. The software project is managed as a small pieces puzzle, which is a way to approach software development that mainly differentiates XP from other traditional software development methodologies.

Software requirements are collected by means of user stories written by the customers as “things that the system has to do”. User stories consist of two or three sentences and drive the creation of acceptance tests.

The four basic activities in the XP life cycle model are *coding*, *testing*, *listening*, and *designing*. There is no special order for them since all happen in parallel.

- Coding* means constantly creating the simplest possible solutions for tasks based on single user stories or possibly on a few related stories at once. Coding is performed by pairs of programmers working together.
- Testing* means creating executable test scripts even before coding starts. They are run daily, and the number of them that succeed measures the velocity of the project.
- Listening* means meeting users regularly and collecting their own stories (the so-called user stories). These form the basis of both coding and testing.
- Designing* means applying design patterns, going over the code and ‘refactoring’ it in order to remove any duplications or any inelegant constructs.

XP methodology is an iterative process. It consists in applying 12 basic rules, which are also called XP practices.

- Planning game²⁴. It is related to the project planning activity and is aimed at defining a release plan (the user stories to be implemented for each release, and their dates). The planning activity is treated as a game with a goal, playing pieces, players, and rules for allowable moves. The playing pieces are the user stories. They are estimated, and based on such estimates a number of them are put into production. The goal of the planning game is to put the larger number of stories in production over the game time. The Planning game is an iterative process itself, and has three phases: *exploration*, *commitment*, and *steering*. The release plan provides the set of stories from which the customer can choose during the iteration planning meeting, whose objective is to plan releases for the next iteration.

²⁴ <http://c2.com/cgi/wiki?PlanningGame>

- Small releases. The approach consists in releasing new versions of the system quickly and continuously.
- Metaphor. The project development is guided by a shared metaphor for the system. This approach simplifies the definition of naming/coding conventions and allows developers to share the intuition of how the overall system works.
- Simple design. The design should be simple and intuitive; the use of design pattern is crucial in this methodology.
- Testing. The development is *test-driven*. Developers write and run tests first, and then code. Each user stories has its set of acceptance tests to be satisfied. Until this happens, the user story is not considered complete.
- Refactoring. This practice has to be performed whenever is needed and possible to keep the code clean and to improve the system design.
- Pair programming. It means that two programmers develop software side by side in one computer. Costs and benefits of pair programming are discussed and shown in [Cockburn and Williams, 2000]. Specifically, the authors show through a project experience that many mistakes are caught on the fly; the end result contains less defects, the design is better and the code is shorter. Besides, people learn more about software development and share knowledge about all pieces of the system, they profit from teamwork and enjoy their work more.
- Collective ownership. In theory, all programmers work on all the pieces of the system.
- Continuous integration. Integration is performed every few hours as well as testing and building.
- 40-hours weeks. Never work overtime.
- Onsite customer. Direct interaction with and involvement of the customer.
- Coding standards. The use of coding patterns and design patterns is crucial in this methodology. They help the entire team to read and to refactor the code.

Figure 7 (based on the figure appearing in the XP web site²⁵) shows the XP methodology. As a first step, the team develops a *Spike Architecture*, a disposable prototype. It is a very simple program used to explore potential solutions and evaluate risks. Based on the *Release Planning*, the needed iterations are performed. Iterations are meant to address a number of user stories that are implemented by following the four basic life cycle model activities, i.e., by following a test-driven approach. Iterations include *Iteration Plan* activities for the next iteration to perform. The software developed during one iteration is presented for judgment. The evaluation is performed by means of the *Acceptance Tests* that have been previously defined by the customers for the user stories. If the software passes the acceptance test, it is then released.

²⁵ <http://www.extremeprogramming.org/>

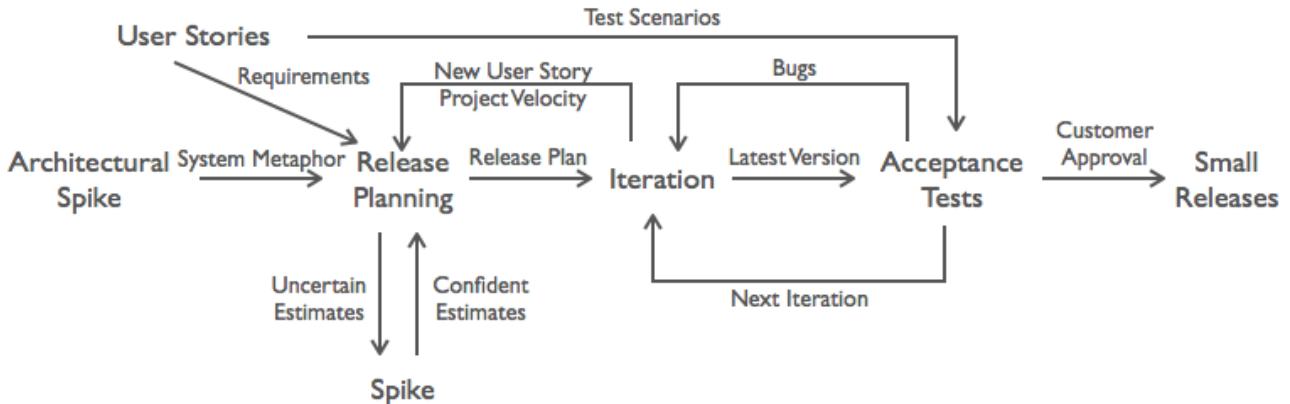


Figure 7. Extreme programming methodology

XP methodology is particularly targeted to small and medium sized teams building software with vague and/or rapidly changing requirements. XP is an agile and very light methodology; it emphasizes customer satisfaction but does not stress the production of detailed design documentation.

2.3.2.9. Reuse components life cycle model

The reuse components life cycle model [Davis, 1988; Jones, 1984] attempts to reduce the development costs since it incorporates previously proven designs and codes into new software products. The basic premise behind this model is that systems should be built using available components, the opposite of custom-building new components. The net effect of reusing components should be shorter development schedules and more reliable software since the developer is using components that have been previously “shaken down”.

The reuse-oriented approach relies on a large base of reusable software components and some integrating framework for these components. While both the initial requirements specification stage and the validation stage are comparable with other processes in other life cycle models, the intermediate stages in a reuse-oriented approach are different. Such intermediate stages include component analysis, requirements modification, system design with reuse, and development and integration.

The advantages of reusing software components are obvious [Sommerville, 2007]. If it is possible to find and use a software component that fulfils the requirements at hand, it will, in general, be cheaper and will include more functionalities than an in-house component would. Furthermore, its quality is known and it is immediately available. With this approach, the amount of software to be developed is reduced, so costs and risks are also reduced. In addition, it usually leads to faster delivery of the software.

However, requirements compromises are inevitable and this may lead to a system that does not meet the real needs of the users. Furthermore, some control over the system evolution is lost as new versions of the reusable components are not under the control of the organization using them [Sommerville, 2007].

Clearly, what is needed are techniques to create reusable components, techniques and tools that store and retrieve reusable components, and component specification techniques that help catalog and locate relevant components.

2.4. State of the Art in Knowledge Engineering

The major difference between Knowledge Engineering, the field of developing knowledge-based systems (KBSs), and Software Engineering is the requirement for knowledge engineers to capture, represent, analyse and exploit knowledge in order to produce a successful knowledge-based system [Kingston, 1994].

Both disciplines (Software Engineering and Knowledge Engineering) have similar aims, that is, to turn the process of developing systems (classical and knowledge-based, respectively) from an art into an engineering discipline. This shift requires to analyse the building and maintenance processes themselves and to develop appropriate methods, languages, and tools for developing systems [Studer et al., 1998].

This section summarizes the main results in development process and life cycle models within the Knowledge Engineering field.

2.4.1. Development Process

Some of the methodologies proposed for building KBSs are based on Software Engineering methodologies and tackle the features of this kind of systems. Such methodologies provide a complete life cycle for the development process and include the guidelines to be followed in the different activities of the process.

In this section we briefly include the development processes proposed by two Knowledge Engineering methodologies: CommonKADS and IDEAL.

In the **CommonKADS methodology** [Schreiber et al., 1994], KBS development entails constructing a set of six engineering models of problem solving behaviour in its concrete organization and application context. This modelling concerns not only expert knowledge but also the various characteristics of how that knowledge is embedded and used in the organizational environment. Hence, a KBS is a computational realization associated with a collection of these models. The models to be developed are

- The *organization model*, which models the main features of the organization developing the KBS.
- The *task model*, which models the different tasks to be supported by the application being developed.
- The *expertise model*, which models the problem solving behaviour of an agent in terms of the knowledge applied to perform a certain task.
- The *agent model*, which models the features and capabilities of the agents (people, information systems, etc.) carrying out tasks.
- The *communication model*, which models the communication between the agents involved in the resolution of a task.
- The *design model*, which models the technical aspects of the KBS.

The aforementioned models are considered not as “steps along the way”, but as independent products in their own right that play an important role during the development process of the KBS [Schreiber et al., 1994].

In the **IDEAL methodology** [Gómez et al., 1997], KBS development and maintenance processes consist of the following five main phases, which includes several stages:

- Phase I: Task Identification.* In this phase the KBS objectives and problem characteristics are defined. Based on them, a feasibility study is also carried out to determine whether knowledge engineering techniques are needed for the task. If so, a first version of systems requirements is specified.

- *Phase II: Development of the different prototypes.* This development allows defining and refining system requirements. This phase includes the following main activities for each prototype: (a) knowledge acquisition, (b) conceptualization and formalization, (c) implementation and (d) evaluation and validation.
- *Phase III: Development of the complete and integrated system.* In this phase, the integration of the KBS with other systems takes place.
- *Phase IV: Perfective maintenance.* This phase includes the maintenance of the whole system, the maintenance of the knowledge bases, and the acquisition of new knowledge.
- *Phase V: Proper technology transfer.* In this phase, the technology transfer is organized and the documentation of the KBS is completed.

2.4.2. Life Cycle Models

When building an expert system (ES), which is a special type of KBS, the objective is to model the behaviour of the experts working in their domain of expertise. In this type of systems, it is very difficult to establish the requirements to design a system that models the subjective expert behaviour [Juristo and Pazos, 1993]. This implies that it is difficult to establish *a priori* the requirements, and less so a definition of specifications. Therefore, it is necessary to use a process of progressive improvement to define the requirements gradually. In addition, the operative product will never be completely finished since its use by the experts should improve their services, which should, in turn, lead to a new version of the system, thus producing a feedback cycle which, at least to begin with, although toned down, is positive and therefore unending.

In this section we include briefly the KBS life cycle models proposed by two Knowledge Engineering methodologies: CommonKADS and IDEAL.

In CommonKADS methodology [Schreiber et al., 1994], the life cycle model proposed is a **cyclic, risk-driven model similar to Boehm's life cycle model** (presented in Section 2.3.2.7). For each project a specialized life cycle, based on the life cycle model proposed, is configured depending on specific project objectives and risks.

In a CommonKADS project, the KBS development [Schreiber et al., 1994] usually consists of several cycles, depending on the identification of new objectives and risks. Steps within a cycle can be repeated many times. The CommonKADS model set (described in 2.4.1) provides a comprehensive and organized collection of aspects that can be relevant in a KBS project. However, this does not mean that in an actual project all models have to be fully developed; only those model components and states that affect the project objectives and risks are selected. Whenever possible, parallel development of models is encouraged. Models must be maintained over the life cycle of the KBS. Control of quality and progress is integrated through the regular checking of model states that must be reached in each cycle.

Figure 8 gives a stylized representation of how project management and development works are connected through model states. At the start of a management cycle, objectives for the cycle are defined, and associated risks are identified. From these objectives and risks, a set of model states is derived, which must be realized within the cycle. These target model states are projected onto development activities that should result in "filling" elements of the CommonKADS models.

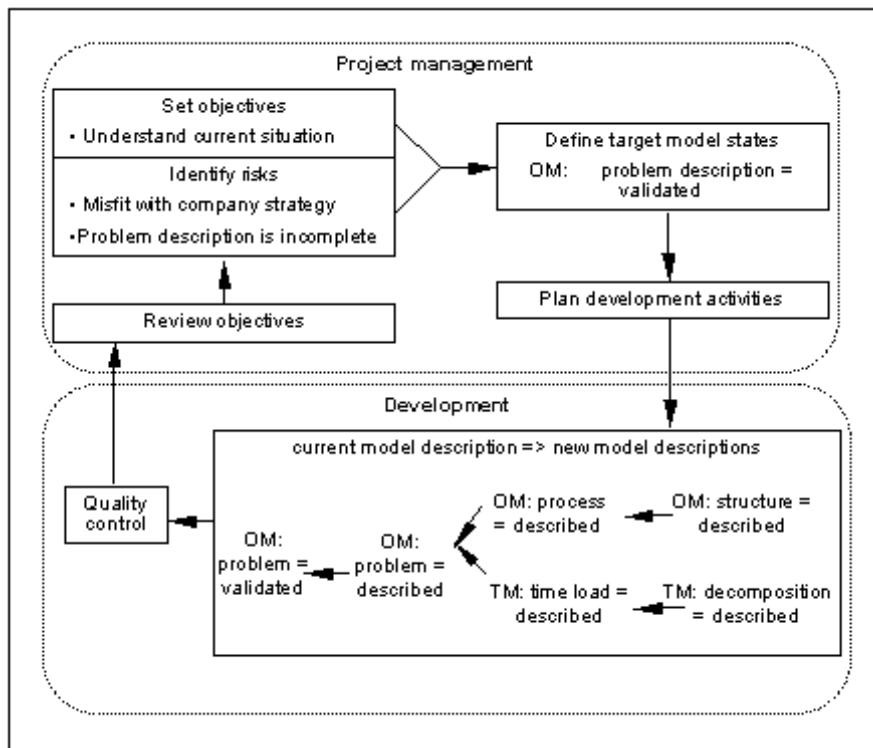


Figure 8. Example of CommonKADS cycle²⁶ [Schreiber et al., 1994]

The IDEAL methodology [Gómez et al., 1997] proposes a **conical-spiral life cycle model** (shown in Figure 9), based on Boehm's spiral model but in three dimensions, for the case of KBS [Juristo and Pazos, 1993]. The authors [Juristo and Pazos, 1993] claim that a KBS life cycle model cannot be defined by means of the two axes of the classical software (cost and time), but by the addition of a new axis, which reflects the intrinsic characteristic of knowledge. This new axis could be the quality of knowledge which, like cost and time, increases during the life of a KBS. The third dimension of the conical-spiral life cycle model would correspond to *adaptative or perfective maintenance*, i.e., the incorporation of new knowledge into the system, which the expert acquires in time. Since knowledge generates always new knowledge, a kind of adaptative maintenance, where new knowledge is added to the system as experts obtain new experience, is always necessary. Perfective or adaptative maintenance entails building in the new knowledge acquired by the expert at a later stage. This would call for another spiral life-cycle with shorter stages and smaller prototypes, located on another plane, to produce the spiral cone [Alonso et al., 1995].

²⁶ OM: Organization Model; TM: Task Model.

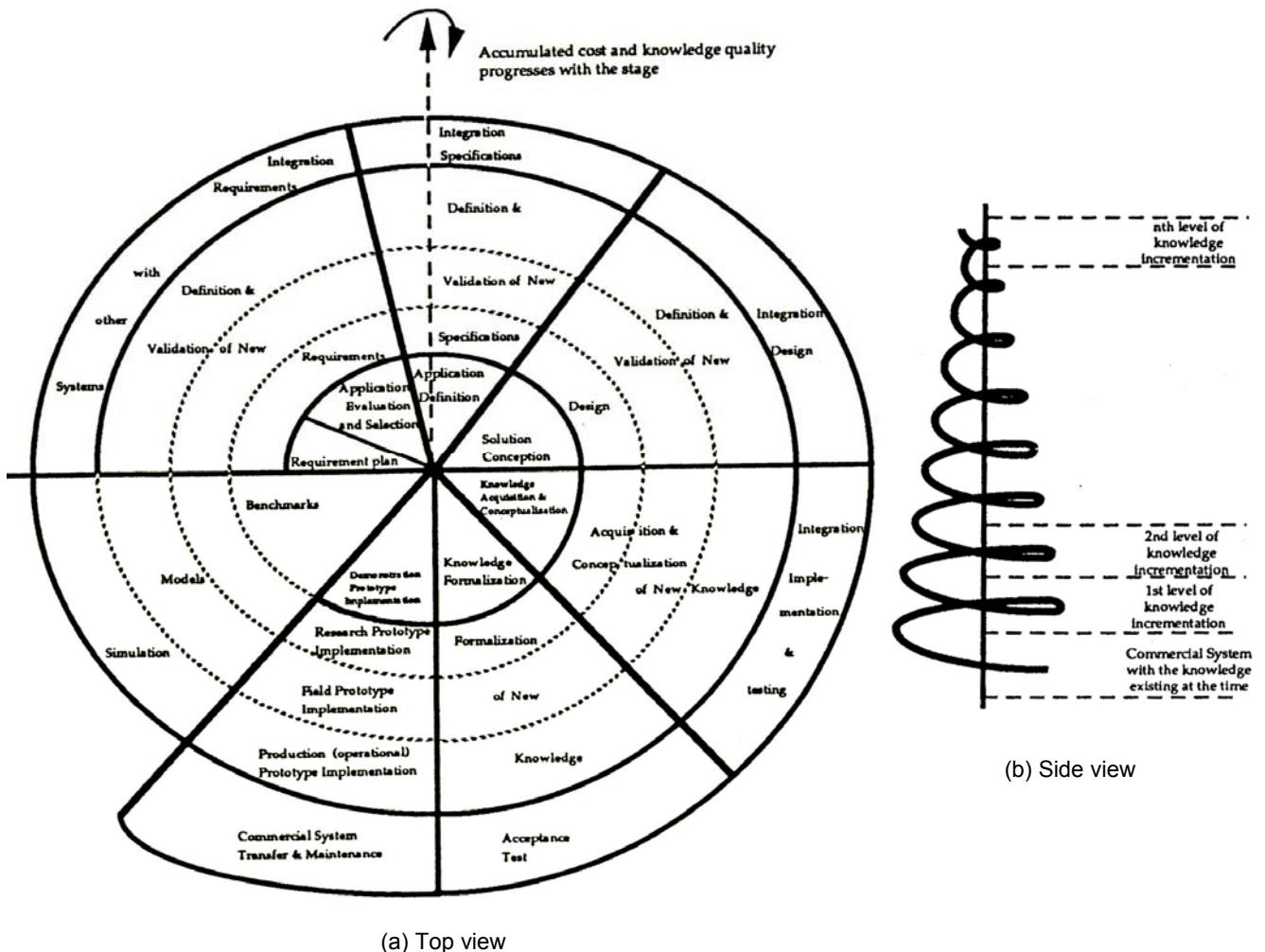


Figure 9. Conical-Spiral life cycle model [Alonso et al., 1995]

The conical-spiral life cycle model would have the development of the ES on one plane, based on a specific level of expert knowledge. Its development would involve the use of prototypes to define the initial system and to refine the requirements and specifications, obtaining a review and operational prototype system [Alonso et al., 1995]. The number of prototypes to be developed will depend on cost analysis and problem complexity. Later, the ES would be integrated into a general purpose system to make a commercial system [Alonso et al., 1995].

In short, according to [Alonso et al., 1996], a life cycle model for an ES with a spiral structure and conical prototype, based on a modular design with small well-defined modules and on modules which are easily related to the problem and independent, would make it possible to obtain a highly maintainable, adaptable, and flexible product, i.e., a minimum-cost system, which is the aim of all methodologies.

2.5. State of the Art in Ontology Engineering

This section includes a summary of the most well-known methodologies for building ontologies, paying special attention to how such methodologies propose carrying out the establishment of the ontology life cycle (Section 2.5.1). The section also includes the most significant work on ontology requirements specification (Section 2.5.2) and ontological resource reuse (Section 2.5.3).

2.5.1. Main Methodologies

This section includes chronologically a brief description of the three most used methodologies for building ontologies (METHONTOLOGY, On-To-Knowledge and DILIGENT). A detailed explanation can be found in [Gómez-Pérez et al., 2003].

In the methodologies dealt with, we pay special attention to the ontology development process and to the establishment of the ontology life cycle as part of the scheduling activity.

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

- Collaboration in the ontology development.
- Degree of coverage of the processes or activities included in this thesis (that is, ontology requirements specification, scheduling, and reusing ontological resource) and detail of the guidelines provided.
- Observance of whether the three methodologies mentioned are targeted to software developers and ontology practitioners or to ontology researchers.

2.5.1.1. METHONTOLOGY

METHONTOLOGY methodology [Fernández-López et al., 1997; Blázquez et al., 1998] was developed within the Ontology Engineering Group at Universidad Politécnica de Madrid. This methodology enables the construction of ontologies at the knowledge level.

This methodology includes the identification of the ontology development process (which is the set of activities to be carried out to build ontologies), a life cycle based on evolutionary prototyping, and techniques to carry out each activity during the management, development-oriented, and support activities.

To give technological support to METHONTOLOGY, ODE [Blázquez et al., 1998] and WebODE [Arpírez et al., 2003] were built within the same group at Universidad Politécnica de Madrid. Some other ontology tools and tool suites can also be used to build ontologies following this methodology, for example, the NeOn Toolkit, Protégé, etc. It should be mentioned that METHONTOLOGY has been proposed²⁷ for ontology construction by the Foundation for Intelligent Physical Agents (FIPA), which promotes inter-operability across agent-based applications.

The ontology development process [Fernández-López et al., 1997; Blázquez et al., 1998] was identified on the framework of the METHONTOLOGY methodology for ontology construction. Such a proposal was based on the IEEE standard for software development [IEEE, 1997]. The ontology development process refers to *which* activities are performed when building ontologies. If an agreement is to be reached on ontologies built by geographically distant cooperative teams, it is crucial to identify these activities. Such activities are organized into the three following categories: management, development oriented, and support, as Figure 10 shows.

²⁷ <http://www.fipa.org/specs/fipa00086/> (last access, January 16, 2008)

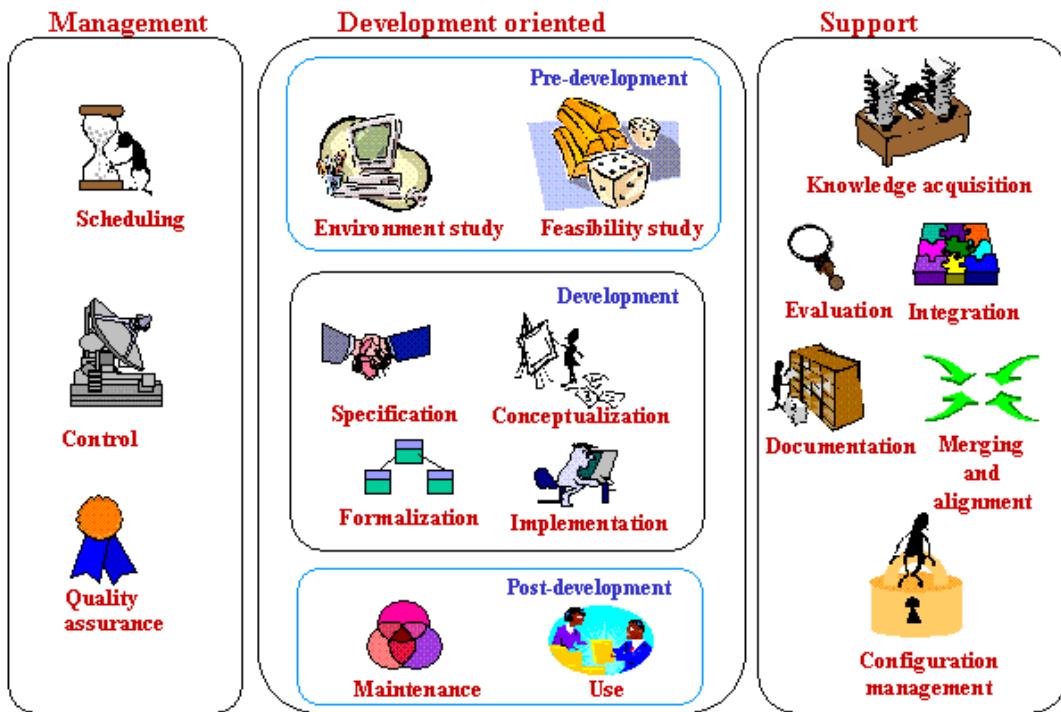


Figure 10. METHONTOLOGY ontology development process

1. **Ontology management activities** include activities that initiate, monitor, and control an ontology project throughout its life cycle. These activities are
 - ❑ The *scheduling* activity, which identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion.
 - ❑ The *control* activity, which guarantees that scheduled tasks are completed in the manner intended to be performed.
 - ❑ The *quality assurance* activity, which guarantees that the quality of each and every product output (ontology, software, and documentation) is satisfactory.
2. **Ontology development oriented activities** are grouped, as presented in Figure 10, into pre-development, development and post-development activities.
 - a) The *pre-development activities* include activities that explore and allocate requirements before the ontology development can begin. These activities are
 - ❑ An *environment study* is carried out to know the project environment, the organization where the ontology will be developed, the participants in the ontology development, etc.
 - ❑ A *feasibility study* answers questions such as: Is it possible to build the ontology? Is it suitable to build the ontology? etc.
 - b) The *development activities* include activities performed during the development and enhancement of an ontology project. Such activities are
 - ❑ The *specification* activity states why the ontology is being built, which its intended uses are, and who are the end-users.
 - ❑ The *conceptualization* activity structures the domain knowledge as meaningful models at the knowledge level [Newell, 1982].
 - ❑ The *formalization* activity transforms the conceptual model into a formal or semi-computable model.

- The *implementation* activity builds computable models in an ontology language.
 - c) The **post-development activities** include the *maintenance* activity, which updates and corrects the ontology if needed, and the *(re)use* activity, which refers to the *(re)use* of the ontology by other ontologies or applications.
3. **Ontology support activities** include activities that are necessary to assure the successful completion of an ontology project. This group includes a series of activities performed at the same time as the development-oriented activities, without which the ontology could not be built. Such activities are
- The *knowledge acquisition* activity whose goal is to acquire knowledge from experts of a given domain or through some kind of (semi)automatic process, which is called ontology learning.
 - The *evaluation* activity makes a technical judgment of the ontologies, of their associated software environments, and of the documentation [Gómez-Pérez, 2004]. This judgment is made with respect to a frame of reference during each stage and between stages of the ontology's life cycle.
 - The *integration* activity is required when building a new ontology by reusing other ontologies already available.
 - The *merging* activity consists in obtaining a new ontology from several ontologies on the same domain. The resulting ontology is able to unify concepts, terminology, definitions, constraints, etc., from all the source ontologies. The merge of two or more ontologies can be carried out either in run-time or design time.
 - The *alignment* activity establishes different kinds of mappings (or links) between the ontologies involved. Hence this option preserves the original ontologies and does not merge them.
 - The *documentation* activity details, clearly and exhaustively, each and every one of the completed stages and products generated.
 - The *configuration management* activity records all the versions of the documentation and the ontology code to control the changes.

As already mentioned, the ontology development process does not identify the order in which the activities should be performed. This is the role of the ontology life cycle. The ontology life cycle identifies *when* the activities should be carried out, that is, it identifies the *set of stages* through which the ontology moves during its life time, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.). METHONTOLOGY proposes an ontology building life cycle based on what the authors call evolutionary prototyping life cycle model because it allows adding, changing, and removing terms in each new version of the ontology (prototype). For each prototype, METHONTOLOGY proposes beginning with the schedule activity that identifies the tasks to be performed, their arrangement, and the time and resources needed for their completion. After that, the ontology requirements 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 to the development activities (specification, conceptualization, formalization, implementation and maintenance) during the whole life cycle of the ontology.

In the case of the **ontology requirements specification activity**, METHONTOLOGY proposes the use of competency questions or intermediate representations for describing the requirements that the ontology should fulfil. However, this methodology does not provide detailed guidelines for carrying out this activity. 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. METHONTOLOGY provides detailed guidelines for carrying out this conceptualization activity. Then the formalization and implementation activities are carried out. If some problem is detected after any of these activities, we can return to any of the previous ones 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 11 shows the ontology life cycle proposed in METHONTOLOGY; it 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.

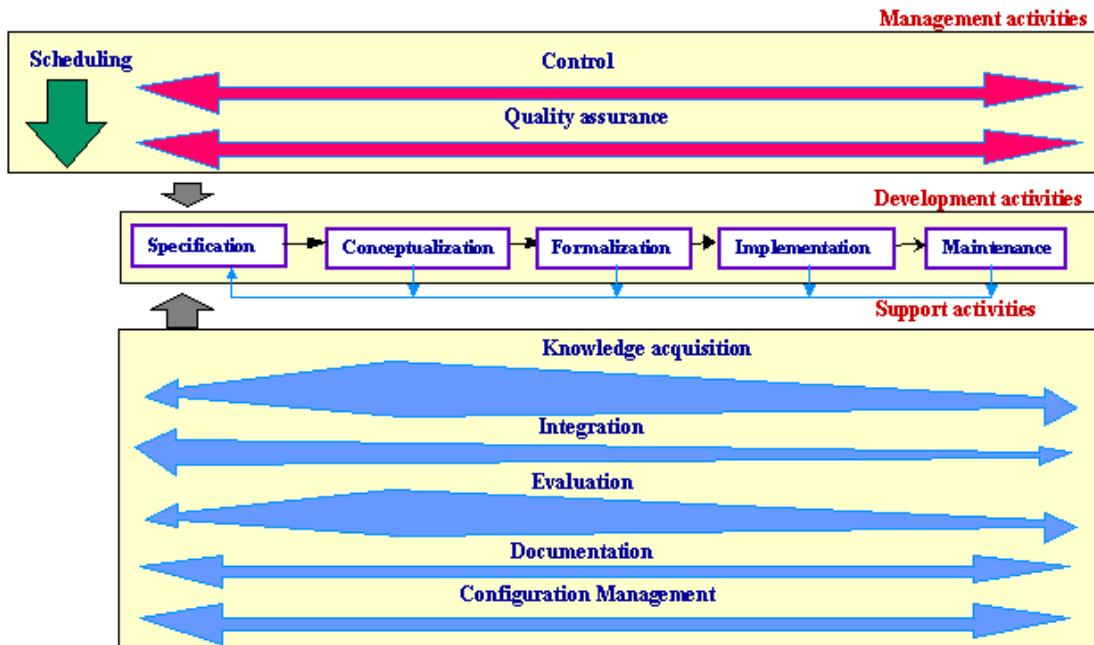


Figure 11. METHONTOLOGY ontology life cycle

The idea of integrating an ontology into a new one is related to the **reuse of 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 and does not provide detailed guidelines. The activities proposed are shown in Figure 12.

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

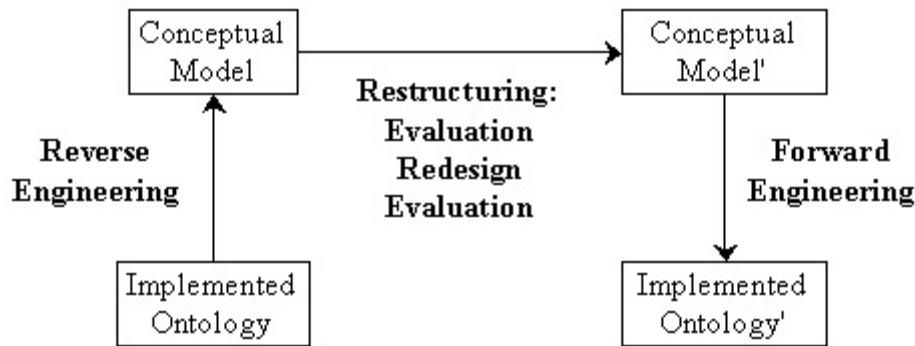


Figure 12. Ontology reengineering activities [Gómez-Pérez and Rojas-Amaya, 1999]

Related to the support activities, Figure 11 also shows that knowledge acquisition, integration and evaluation are greater during the ontology conceptualization, and that they decrease 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. The integration at the knowledge level should be carried out before the integration at the implementation level.
- 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 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 [Fernández-López et al., 2000]. Therefore, METHONTOLOGY takes into account 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, most of the times and before integrating an ontology into a new one, the ontology to be reused is modified or merged with other ontologies of the same domain.

We can also say that METHONTOLOGY does not mention anything about **collaboration**.

The main METHONTOLOGY contributions to the area were

- Identification of the ontology development process.
- Identification of the life cycle.
- Provision of 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 to ontology engineers and researchers.

2.5.1.2. On-To-Knowledge

The aim of the On-To-Knowledge project [Staab et al., 2001] was to apply ontologies to electronically available information for improving the quality of knowledge management in large and distributed organizations. The partners involved in this project developed a methodology and some tools for the intelligent access to large volumes of semi-structured and textual information sources in intra-, extra-, and internet-based environments. The **On-To-Knowledge methodology** for building ontologies proposes building ontologies taking into account their future use in further knowledge management applications. Consequently, the ontologies developed with this methodology are highly dependent of the application.

Another important characteristic is that On-To-Knowledge proposes ontology learning for reducing the efforts made to develop the ontology. However, On-To-Knowledge does not regard the **collaboration** dimension.

The methodology also includes the identification of goals to be achieved by knowledge management tools and is based on an analysis of usage scenarios [Staab et al., 2001].

The processes proposed by this methodology can be summarized as follows:

1. **Feasibility study.** On-To-Knowledge adopts the kind of feasibility study described in the CommonKADS methodology [Schreiber et al., 1999]. 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.
2. **Kickoff.** The result of this process is the ontology requirements specification document that describes the following issues: domain and goal of the ontology; 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.

On-To-Knowledge proposes competency questions (CQs) [Grüninger and Fox, 1994] for carrying out the **ontology requirements 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 on the inclusion or exclusion of concepts in the ontology and on 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 nor for reusing them. Apart from that, this methodology does not explicitly provide guides for the **reuse and reengineering of non-ontological resources**, nor for the **reuse of ontology design patterns**.

3. **Refinement.** The goal here is to produce a mature 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 the interaction with domain experts. Once this activity is performed, axioms are identified and modelled. During the elicitation, the concepts are gathered on one side and the terms to label the concepts on the other. Then, both elements are mapped.

The On-To-Knowledge methodology proposes the use of intermediate representations to model the knowledge. In this respect, it follows METHONTOLOGY's basic ideas.

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. The language is selected according to the specific requirements of the application envisaged.

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, but other ontology editors that perform similar functions can also be used.

4. **Evaluation.** The evaluation process serves as a proof of the usefulness of the ontologies developed 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 may be needed 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 level envisaged.

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.

Finally, it is important to mention that ontologies developed with this methodology are highly dependent on the application. However, the methodology is not **targeted to software developers nor ontology practitioners**.

The **On-To-Knowledge methodology** proposes an incremental and cyclic ontology life cycle, based on evolutionary prototyping life cycle model [Gómez-Pérez et al., 2003]. This ontology life cycle is shown in Figure 13.

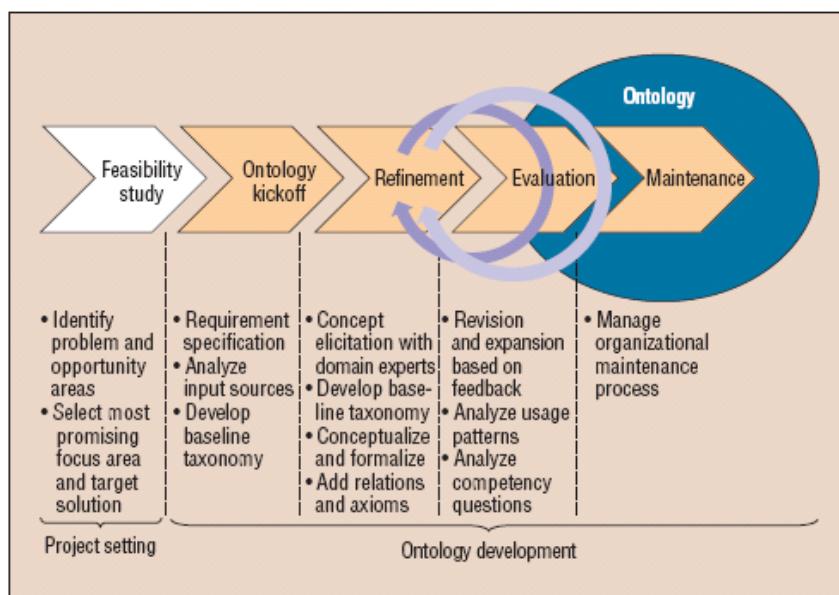


Figure 13. On-To-Knowledge ontology life cycle [Staab et al., 2001]

2.5.1.3. DILIGENT

The **DILIGENT methodology** [Pinto et al., 2004] was developed by the Institute AIFB of the University of Karlsruhe and the *Instituto Superior Técnico of Lisboa*. This methodology is intended to support domain experts in a distributed setting in order to engineer and evolve ontologies. This methodology is focused on **collaborative** ontology engineering, and its central issue is to keep track of the change arguments.

The ontology development process proposed by this methodology includes the following five main phases:

1. **Build.** The build phase aims to create an initial version of the ontology quickly, so that the stakeholder can start using the ontology soon [Engler et al., 2006]. Domain experts, users, knowledge engineers and ontology engineers collaboratively create an initial version of the ontology. However, the team involved in building the initial ontology should be relatively small in order to find more easily 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 carrying out the **ontology requirements specification activity** nor does it propose **reusing and reengineering available knowledge resources**.
2. **Local adaptation.** During the next phase, users locally adapt the ontology according to their own needs, while the ontology is in use, for example, to organize knowledge [Engler et al., 2006]. Once the shared ontology is made available, users can start using it and locally adapting it for their own purposes. Typically, due to new business requirements or user and organization changes, their local ontologies evolve in a similar way to folder hierarchies do in a file system. In their local environment, they are free to change the local copy of the shared ontology. However, they are not allowed to directly change the ontology shared by all users. All local changes of the shared ontology are collected by a central control board.
3. **Analysis.** The analysis phase requires the ontology control board to evaluate the changes suggested by the stakeholders [Engler et al., 2006]. The input from users provides the necessary arguments to underline change requests, whereas the control board analyses the local ontologies and the change requests and tries to identify similarities in users' ontologies. A crucial activity of the board is to decide which changes are going to be introduced in the next version of the shared ontology. Then, a balanced decision that takes into account the different needs of the users and meets the user's evolving requirements has to be found. The goal of this phase is to develop a core-shared ontology because otherwise its size will grow fast, and it will become un-maintainable.
4. **Revision.** Then, the board revises the ontology by deciding which changes should be applied to the ontology [Engler et al., 2006]. The board should regularly revise the shared ontology in order to avoid a larger divergence of the local ontologies from the shared ontology. Therefore, the board should have a well-balanced and representative participation of the different types of participants involved in the process, the different needs of the users, and their evolving requirements.
5. **Local update.** In the last step the stakeholders update their local ontologies based upon the revised version of the ontology [Engler et al., 2006]. Once a new version of the shared ontology is released, users can update their own local ontologies to better use the knowledge represented in the new version. Even if the differences are small, users may rather reuse the new concepts, for example, instead of using their previously locally defined concepts that correspond to the new concepts represented in the new version.

The shared ontology may contain several of the changes that were introduced in the local adaptation phase; other changes, however, may be missing. Because the control board tries to balance the different users' needs, 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.

The **DILIGENT methodology** proposes an ontology life cycle model based on evolutionary prototyping life cycle model, shown in Figure 14.

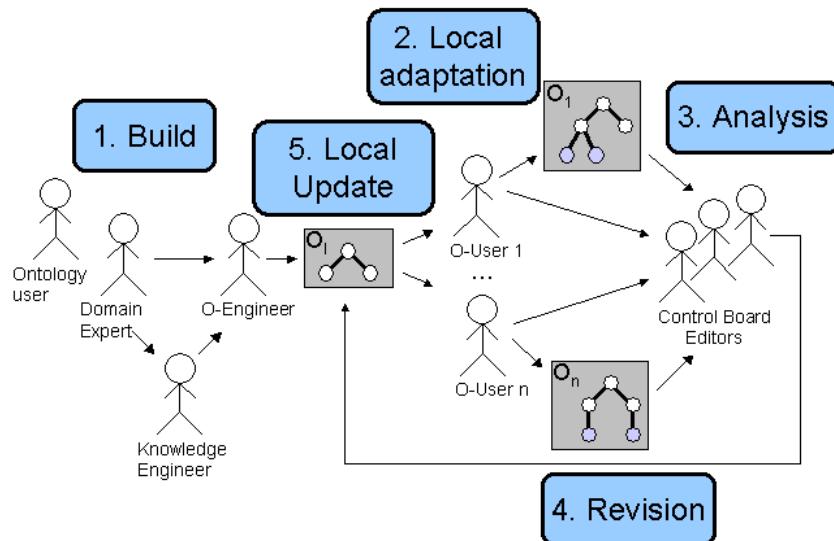


Figure 14. DILIGENT ontology life cycle [Pinto et al., 2004]

As already mentioned, the DILIGENT methodology is an argumentation framework that facilitates discussions about the design rationale of changes introduced in the different phases of the life cycle. The arguments exchanged help the control board in understanding the reasons for specific changes, especially in the analysis and revision phases [Engler et al., 2006].

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, for example, a change was made into the new revision of the shared ontology whereas other was not. More details about the DILIGENT argumentation framework and how it facilitates collaborative ontology engineering are available in [Dellschaft et al., 2008].

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

2.5.2. Ontology Requirements Specification

As we will define in Section 5.3, **Ontology Requirements Specification** is the activity of collecting the requirements that the ontology should fulfil. The output of this activity is the ontology requirements specification document (ORSD), which includes the purpose, the level of formality and scope of the ontology, the target group and the intended uses of the ontology, as well as a set of requirements, which are those needs that the ontology to be built should cover.

In this section we present a brief summary of the methods, techniques and tools available for ontology requirements specification.

2.5.2.1. Methods and Techniques

In this section we present some general methods and techniques for carrying out the ontology requirements specification activity.

METHONTOLOGY [Gómez-Pérez et al., 2003; Fernández-López et al., 1997; Blázquez et al., 1998] proposes the goals of the ontology requirements specification activity; however, it does not propose any method for carrying out the activity.

Grüninger and Fox methodology [Grüninger and Fox, 1995], On-To-Knowledge methodology [Staab et al., 2001], and Uschold [Uschold, 1996] propose the following steps for obtaining the so-called ontology requirements specification document:

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

There are different techniques that can be applied for collecting requirements. Examples of these techniques are brainstorming, joint application development (JAD) [Pressman, 2001], exploit scenarios and use cases using templates, interviews with users and domain experts, and competency questions.

Most of the methodologies or methods [Grüninger and Fox, 1995; Staab et al., 2001; de Hoog, 1998; Hristozova and Sterling, 2003; Uschold, 1996] and guides [Noy and McGuinness, 2001] for developing ontologies suggest identifying **competency questions** as a technique for establishing the ontology requirements. Competency questions (CQs) were proposed for the first time in [Grüninger and Fox, 1995], where they were defined as questions that the ontology to be built should be able to answer.

Next, we present how the different methodologies and guides abovementioned propose carrying out the ontology requirements specification activity by means of competency questions.

- Grüninger and Fox's methodology* [Grüninger and Fox, 1995] 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 can be used. Such scenarios describe a set of the ontology requirements that the ontology should satisfy after being formally implemented. The scenarios may be presented by industrial partners and concern problems encountered in their enterprises. The motivating scenarios often have the form of story problems or examples 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 intended informal 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 that are 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 (complex queries) requiring the solution of lower level ones (simple queries). Competency questions are not well designed 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. Specific competency questions can be included into more general questions that are answered by composing answers associated to specific competency questions.

- ❑ *On-To-Knowledge methodology* [Staab et al., 2001] states 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.
- ❑ *Uschold* [Uschold, 1996] proposes identifying (a) the purpose of the ontology and, specifically, identifying and characterizing the range of the intended users, (b) the uses of the ontology, and (c) (fairly general) motivating scenarios and competency questions, and producing a user requirements document for the target software system. After that, the methodology recommends deciding how formal the ontology needs to be. This decision is determined in large part by the users and by the purpose of the ontology. Finally, this methodology proposes identifying the scope of the ontology by means of (a) creating the detailed motivating scenarios that arise in the applications, which is also proposed by Grüninger and Fox's [Grüninger and Fox, 1995] or (b) using brainstorming to do a more thorough and accurate job of scoping.
- ❑ *The EXPLODE methodology* [de Hoog, 1998; Hristozova and Sterling, 2003] integrates ideas from the eXtreme Programming methodology. This methodology is particularly suitable for dynamic and open environments thanks to its focus on immediate feedback and evaluation. It proposes fetching the requirements of the system and defining the competency questions.
- ❑ *The “Ontology Development 101” guide* [Noy and McGuinness, 2001] proposes determining the domain and scope of the ontology by answering a set of basic questions (“What is the domain that the ontology will cover?”, “What will be the uses of the ontology?”, “Who will use and maintain the ontology?”, etc.) and identifying the ontology competency questions.

2.5.2.2. Tools

After analysing the state of the art, we realized that there is only one tool for supporting the creation of ontology requirements. This tool is called **OntoKick** [Sure et al., 2002] and is used to create the requirement specification document and to extract relevant structures for building the semi-formal ontology description.

OntoKick is an OntoEdit [Sure et al., 2002] plug-in that supports the collaborative generation of requirement specifications for ontologies. OntoKick can be used for 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), potential users, use cases, and applications supported by the ontology. This tool employs 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 an explicit requirement for the ontology. OntoKick takes further advantage of 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, it establishes and maintains links between CQs and concepts derived from them. This permits a better traceability of the origins of the concept definitions in later stages.

2.5.3. Ontological Resource Reuse

As will be defined in Section 5.3, ***Ontology Resource Reuse*** is the process of using available ontological resources (ontologies, modules, statements, or ontology design patterns) for the solution of different problems (e.g., the development of different ontology-based applications, the activity of ontology aligning (as background knowledge)). We distinguish between ontology reuse, ontology module reuse, ontology statement reuse, and ontology design pattern reuse.

In this section we present a brief summary of the methods, techniques and tools available for reusing ontological resources.

2.5.3.1. Methods and Techniques

Next we present different approaches to carry out the reuse of ontological resources.

- ❑ *METHONTOLOGY* [Gómez-Pérez et al., 2003] propose some activities to reuse ontologies in a particular domain. These are the following:
 - a. Finding candidate ontologies to be reused.
 - b. Inspecting the content and granularity of the candidate ontologies.
 - c. Selecting the ontologies to be reused.
 - d. Evaluating the selected ontologies from knowledge representation point of view.

This reuse process is illustrated in the methodology with a real example.

- ❑ *Uschold et al.* [Uschold et al., 1998] identify the main tasks involved in reusing available ontologies. Reusing ontologies involve the following steps:
 1. Understanding the ontology and finding a kernel to reuse.
 2. Translating the ontology.
 3. Specifying and refining the ontology into executable code. The goal here is to define refinements of the specifications produced in the above steps, and produce the executable code.
 4. Verifying the ontology refined, which will guarantee that the executable code is true to the original specification.
 5. Integrating the ontology with the application.
- ❑ *Pinto and Martins* [Pinto and Martins, 2001] propose the following activities for reusing ontologies as part of the integration process:
 1. Searching and selecting the candidate ontologies. The ontologies to be analysed, and perhaps reused, are chosen from those available in libraries that meet a series of requirements. The ontologies chosen as candidates must also be compatible with the requirements specified for the resulting ontology. The authors divide requirements into (a) strict²⁸ requirements, such as 'candidate ontology should represent a similar domain' and (b) desirable²⁹ requirements, such as 'candidate ontology should have been evaluated'. Strict requirements eliminate the ontologies that are not appropriate to be reused. Desirable requirements distinguish between those ontologies that are more likely to be better candidates, but they do not exclude candidate ontologies.

²⁸ Strict means hard or "must have"

²⁹ Desirable means soft or "good to have"

2. Evaluating with an integration approach. The candidate ontologies should be evaluated from an integration point of view. The authors have identified a series of criteria that domain experts should take into account when analyzing an ontology for integration. The domain experts should evaluate the ontology paying special attention to what knowledge is missing; what knowledge should be removed; what knowledge should be relocated; and which knowledge sources, documentation, terminology, definitions, and practices should be changed.

These evaluation criteria show the weaknesses and strong points of the candidate ontology from the domain expert point of view.

3. Assessing with an integration approach. The candidate ontologies should be assessed from an integration point of view. The authors have identified a series of criteria that ontologists should take into account when analyzing an ontology for integration. Ontologists should assess the ontology paying special attention to the following items of the ontology: general structure, basic distinctions, structuring relation, naming convention rules, definitions, documentation, and knowledge pieces represented.
4. Selecting the adequate ontology. After the analysis of various candidate ontologies, and given the fact that they may not perfectly match the ontology needed, another choice must take place. This selection must be made among the candidate ontologies that passed strict requirements and among those that scored best in integration oriented evaluation and assessment; then, the ontology chosen must be the one that best suits our needs, or that can more easily or better be adapted to them. This selection also depends, to some extent, on the other ontologies that are going to be reused, since more than one ontology can be reused in an integration process.
5. Integrating the selected candidate ontology.
6. Evaluating and assessing the resulting ontology.

- *Paslaru and Mochol* [Paslaru and Mochol, 2005] propose an incremental reuse process that consists in
1. Considering the vocabulary of the candidate ontologies (concepts, relations, and axioms) and computing a common vocabulary depending on the natural language in which the ontological primitives have been originally denominated.
 2. Merging the candidate ontologies vocabularies, generating separate lists of ontological primitives according to the degree of formality of the considered models, and eliminating duplicates in order to avoid unnecessary computations.
 3. Computing syntactical similarities between concept names belonging to different source ontologies [Cohen et al., 2003].
 4. Improving the accuracy of the similarity computation, generating a bag of terms for each concept name in the source vocabularies.
 5. Computing the ranking of the concepts by considering frequencies (concept names occurring in several source ontologies are ranked higher), source priority (relevance measure of the corresponding source to the target application domain) and application requirements.
 6. Identifying relevant concepts. The user selects the relevant relationships, which can be added incrementally to the ontology until a certain level of complexity is achieved.

2.5.3.2. Tools

CORE (Collaborative Ontology Reuse and Evaluation) [Fernández et al., 2006] receives as input an informal description of a semantic domain and determines which ontologies, from an ontology repository, are the most appropriate to describe the given domain. For this task, the tool is divided into three main modules.

1. The first component receives the problem description, represented as a set of terms and permits the user to refine and enlarge it using WordNet.
2. The second module applies a set of criteria to evaluate the ontologies of the repository and then determines which ones fit best the problem description. A ranked list of ontologies is returned for each criterion; the lists are combined by means of rank fusion techniques that combine the selected criteria.
3. The third component of the system uses manual user evaluations of the ontologies in order to incorporate a human, collaborative assessment of the quality of ontologies.

ProSe [Jiménez-Ruiz et al., 2008; Jiménez-Ruiz et al., 2008b] is a Protégé plug-in for reusing ontologies. This tool supports the underlying formalisms to reuse ontologies in a safe and economic way. Safety guarantees that the semantics of imported concepts is not changed, whereas economy guarantees that only the relevant part of the ontology will be imported. ProSe intends to guide the user in the proper reuse of external symbols (safety), that is, without adding new implication to them. ProSe also provides a module extraction engine that obtains modules with a reasonable size and with all the necessary axioms to guarantee the coverage of the terms to be reused and. ProSe allows the user to add to the ontology such ontology modules.

Watson [d'Aquin et al., 2007; d'Aquin et al., 2007b] is a gateway to the Semantic Web that collects, analyses and gives access to ontologies and semantic data available on-line. Its objective is to support the dynamic exploitation of ontologies by semantic applications. Watson, which is already coupled with ontology editors, is a specific tool for reusing ontologies.

Watson implements a number of techniques that enhance the basic keyword-based search in ontology repositories by taking into account compound terms for example. Additionally, rather than providing no results when a query is only partially matched, Watson returns a combination of ontologies that together cover the terms considered. It also implements mechanisms that select only relevant ontology modules corresponding to the user/application needs.

2.6. Conclusions

After analysing the state of the art, we can state that

1. The degree of maturity of the Ontology Engineering field is very low when is compared with the Knowledge Engineering field and, specially, with the Software Engineering field. The long term goal of the Ontology Engineering field will be to reach a similar degree of maturity to that of the Software Engineering field today.
2. None of the methodological approaches available in the Ontology Engineering field can help ontology developers to build large ontologies embedded in ontology networks by reusing and possibly reengineering knowledge resources.

Thus, unlike what happens in the Software Engineering field, in the Ontology Engineering field

- **We need a glossary that identifies and defines the processes and activities that could potentially be carried out when single ontologies and ontology networks are developed.**

None of the most well known methodologies (METHONTOLOGY, On-To-Knowledge and DILIGENT) includes shared definitions for the activities they propose. Only METHONTOLOGY provides activity definitions, but they are not obtained by means of consensus process, nor do they include relations to ontology networks.

Additionally, it is also remarkable that in the last few years, new activities have been identified when building ontologies.

This situation is the result of a lack of standardization of the Ontology Engineering field terminology, in contrast with the Software Engineering field that boasts the *IEEE Standard Glossary of Software Engineering Terminology* [IEEE, 1990].

- **We need to fill the gaps with respect to the development process and the life cycle of single ontologies and ontology networks.**

METHONTOLOGY proposes explicitly the ontology development process that identifies a set of activities performed during ontology development, but the set of activities includes also the maintenance and use of the ontology, which is not compliant with the software development process that only refers to development issues [IEEE, 1999]. Furthermore, On-To-Knowledge and DILIGENT do not include an explicit definition for both the development process and the life cycle.

As already mentioned, up to date, there are no methodological approaches that help ontology developers to build large ontologies embedded in ontology networks in complex settings. Thus, in this regard, we can mention that in the Ontology Engineering field

- **We need new methodologies since those available do not cover the complex scenarios in which reuse and reengineering of ontological and non-ontological resources are needed.**

- a. 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 requirements specification to the implementation. However, neither of them allows for 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 proceed quickly with the building of a single ontology and the tracking of all relevant discussions about the conceptualization activity [Engler et al., 2006].
- b. Only METHONTOLOGY considers that the activities performed during the development of an ontology may involve performing other activities in other ontologies already built or under construction [Fernández-López et al., 2000]. Therefore, METHONTOLOGY allows for not only intra-dependencies but also inter-dependencies. Inter-dependencies are defined as the relationships between activities carried out when building different ontologies.
- c. DILIGENT does not take into account the reuse of information and knowledge resources available for speeding up the ontology building process. On-To-Knowledge considers the use of ontology learning methods from textual resources for reducing the efforts made to develop the ontology. METHONTOLOGY proposes a general method for reusing ontologies but does not include prescriptive methodological guidelines.
- d. Of the methodologies mentioned, only METHONTOLOGY provides a definition and initial guidelines for reengineering ontologies [Fernández-López et al., 2000; Gómez-Pérez and Rojas-Amaya, 1999].

- **Most of the methodologies available propose simple methods for carrying out the ontology requirements specification activity, which consist of high level steps.** However, these methodologies do not provide detailed guidelines explaining how to carry out each step, what is needed for obtaining a good Ontology Requirements Specification Document (ORSD), and how the ORSD can be used later on in the ontology development, for instance, (a) to search the knowledge resources to be reused, and (b) to verify the ontology content.
- **We need guidelines that help software developers and ontology practitioners to select a specific life cycle model and thus to create a particular ontology life cycle, as part of the scheduling activity.** Life cycle models defined in Software Engineering have not been seriously analysed, taking into account the new approach in ontology development. Additionally, no new life cycle models have been proposed yet taking into account the special features of ontology networks and the different scenarios in ontology building.
- **We need detailed guidelines for carrying out the ontological resource reuse since most of the methods available only consist of high level steps.** Additionally, such methods do not take into account different granularity during the reuse.

Finally, Table 2 summarizes the presented methodologies (METHONTOLOGY, On-To-Knowledge and DILIGENT) according to the following characteristics:

- Collaboration dimension in the ontology development.
- Degree of coverage of the process or activities included in this thesis (ontology requirements specification, scheduling and ontological resource reuse) by means of providing detailed guidelines.
- Observance of whether the three methodologies mentioned are targeted to software developers and ontology practitioners or to ontology researchers.

	METHONTOLOGY	On-To-Knowledge	DILIGENT
Dimension			
Collaboration	Not mentioned	Not mentioned	Treated
Detailed Guidelines for Processes and Activities			
Ontology Requirements Specification	Not provided Only Competency Questions are proposed	Not provided Only Competency Questions are proposed	This activity is not proposed by the methodology
Scheduling	Not provided	Not provided	Not provided
Reusing Ontological Resources	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
Audience			
Targeted to Software Developers and Ontology Practitioners	Targeted to ontology engineers and researchers	Targeted to ontology engineers and researchers	Intended to domain experts and users

Table 2. Summary of conclusions

Regarding the collaboration dimension, ***none of the analysed methodologies allows for 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 proceed quickly with the building of a single ontology and the tracking of all relevant discussions about the conceptualization activity [Engler et al., 2006].

Table 2 shows that ***none of the analysed methodologies provide detailed guidelines for the process or activities*** (scheduling, ontology requirements specification, and ontological resources reuse) included in this thesis. Based on this fact, we can say that the analysed methodologies are more descriptive than prescriptive since they do not provide instructions to carry out processes or activities.

As a final comment, it should be observed that ***none of the methodologies analysed and described here target software developers and ontology practitioners.***

For all the above reasons, our objective is to create the NeOn Methodology, a methodology for building ontologies and ontology networks, emphasizing the reuse of available knowledge resources (ontological and non-ontological), generalizing from previous experiences, covering the drawbacks of the methodologies available, and taking into account the new trends on collaboration and dynamism. Thus, to solve the above limitations, in this thesis we propose

- Creating the NeOn Glossary of Processes and Activities, which identifies and defines the processes and activities potentially involved when ontology networks are collaboratively built.
- Defining a collection of life cycle models for building ontologies and ontology networks.
- Creating the NeOn Methodology framework for building ontology networks as a scenario-based methodology.
- Providing guidelines for specifying ontology requirements.
- Providing guidelines for establishing the ontology network life cycle and scheduling ontology development projects.
- Providing guidelines for reusing ontological resources.

3. Work Objectives

This chapter presents the goals of our work, together with terminological clarifications and the open research problems, identified in Chapter 2, that we aim to solve. It also describes our contributions to the current state of the art, the work assumptions (premises and hypotheses included) considered as a starting point for this work, and the restrictions of the results obtained.

3.1. Terminological Clarifications

Based on our experience in developing ontologies within different types of projects, we can identify three different possibilities when building ontologies: (1) building single ontologies; (2) building sets of interconnected single ontologies; and (3) building ontology networks.

- A *single ontology* is an ontology that does not have any kind of relationship (domain dependent or independent) with other ontologies.
- A *set of interconnected single ontologies* includes a set of ontologies that have some kind of domain dependent relation among them.
- An *ontology network or a network of ontologies* is a collection of single interconnected ontologies related to each other via a variety of different meta-relationships (based on [Haase et al., 2006]). Examples of these meta-relationships are the following:
 - *hasPriorVersion*: if the ontology to be developed is a new version of an existing one. This meta-relationship is related to the ontology evolution activity defined in Chapter 5.
 - *useImports*: if the ontology is importing any other ontology because the later contains definitions whose meaning are considered to be part of the meaning of the former ontology. This meta-relationship is related to the ontological resource reuse process defined in Chapter 5.
 - *isExtension*: if the ontology extends another existing ontology. This meta-relationship is related to the ontology enrichment activity, defined in Chapter 5. With regard to this meta-relationship, we can also mention *isSpecialization* and *isGeneralization*.
 - *containsModules*: if the ontology to be developed is composed of a number of modules. This meta-relationship is related to the ontology modularization activity defined in Chapter 5.
 - *hasMapping*: if some ontology components have mappings with other existing ontologies. This meta-relationship is related to the ontology mapping activity defined in Chapter 5.

To sum up, if software developers and ontology practitioners explicitly define meta-relationships such as mapping, modularization, version, and dependency between a set of interconnected ontologies and/or between an ontology and their components, then we can say that they are developing an ontology network.

To clarify the difference among a single ontology, a set of interconnected single ontologies, and an ontology network, we provide here some examples.

- An isolated ontology A_1 is a single ontology, as Figure 15 (a) shows.
- N single ontologies related among them by means of domain dependent relations between concepts included in the ontologies are considered a set of interconnected single ontologies, as Figure 15 (b) shows.

- Figure 15 (c) shows the ontology network associated to the set of interconnected single ontologies presented in Figure 15 (b). In this ontology network, the meta-relationships (“useImports” and “hasPriorVersion”) among the different ontologies have been explicitly expressed.

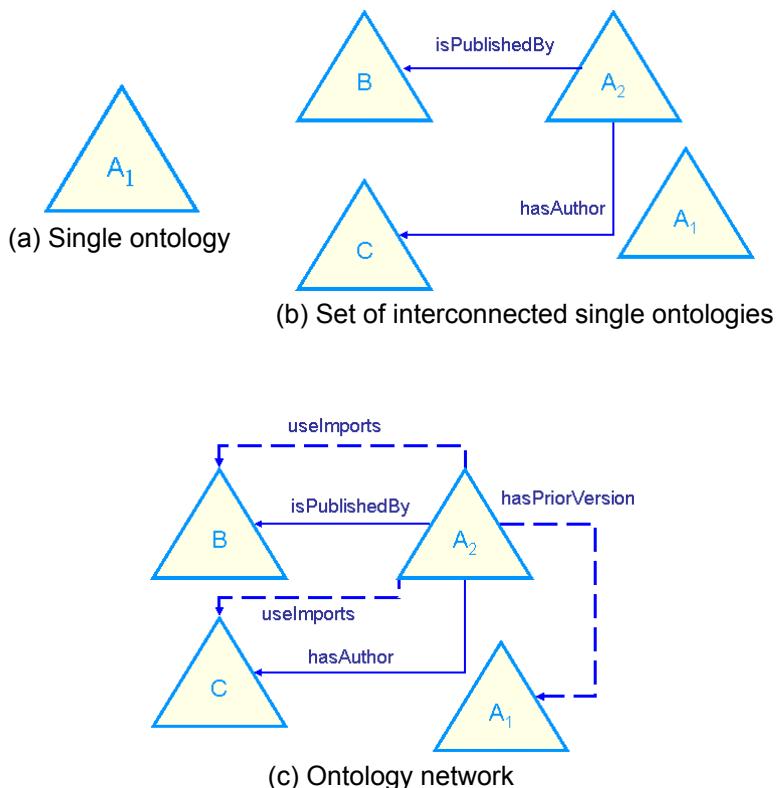


Figure 15. Simple graphical examples of single ontologies, of a set of interconnected single ontologies, and of ontology networks

3.2. Goals and Open Research Problems

The goal of this work is to advance the current state of the art in the Ontology Engineering field, specifically in the methodological area. To attain this goal, three subgoals have been pursued:

- O1. To provide the identification and definition of the development process and life cycle for networks of ontologies
- O2. To propose the NeOn Methodology framework, based on scenarios for building ontology networks. This framework should include methods, techniques and tools for carrying out the activities identified and defined in the ontology network development process.
- O3. To propose detailed methodological guidelines for ontology requirements specification, scheduling, and ontological resource reuse.

In order to achieve the *first objective*, a list of open research problems, derived form the analysis included in Chapter 2, must be solved; these problems are the following:

1. There is neither consensus on nor standardization of the definitions of the processes and activities that potentially could be carried out when single ontologies and ontology networks are developed.

2. There is only one ontology life cycle model (evolving prototypes), in contrast with the set of software life cycle models.

With regard to the *second* and *third objectives*, the following list of open research problems, derived from the analysis included in Chapter 2, must be solved:

1. None of the methodologies here described targets software developers and ontology practitioners, they only target ontology researchers.
2. None of these methodologies can explain the ontology building process with the same style and granularity than those of the methodologies for developing software.
3. None of these methodologies provide detailed prescriptive guidelines for carrying out the ontology requirements specification activity.
4. None of these methodologies provide detailed prescriptive guidelines for selecting a specific life cycle model and creating a particular ontology life cycle (as part of the scheduling activity).
5. None of these methodologies provide detailed prescriptive guidelines for reusing ontological resources at different levels of granularity (as a whole, by modules, or by statements).
6. Finally, none of these methodologies can cover complex scenarios in which the reuse and reengineering of ontological and non-ontological resources are needed.

3.3. Contributions to the State of the Art

In this document, we intend to give solutions to the aforementioned open research problems. Chapters 5 and 7 describe the solutions we propose for objective O1; Chapter 6 describes the solutions for objective O2; and Chapters 8, 9, and 10 describe the solutions related to objective O3.

With regard to the *first objective*, the thesis presents new advances in the state of the art concerning the following aspects:

- ❑ It deals with ontology network development process, ontology network life cycle models and ontology network life cycle, issues all necessary to provide some methodological guidelines for the development of ontology networks.
- ❑ It presents the **NeOn Glossary of Processes and Activities**, which identifies and defines the processes and activities potentially involved when ontology networks are collaboratively built.
- ❑ It also proposes **two different ontology network life cycle models**: waterfall and iterative-incremental, which are related to the scenarios identified in Chapter 6.

The *second* and *third objectives* tackle the NeOn Methodology for building ontology networks and are focused on (1) ontology requirements specification; (2) scheduling; and (3) ontological resource reuse. This thesis also presents advances in the current state of the art in the following aspects:

- ❑ It identifies a set of **nine scenarios for building ontologies and ontology networks**, emphasizing the reuse of knowledge resources (ontological and non-ontological).
- ❑ It provides **prescriptive methodological guidelines for carrying out the ontology requirements specification activity**.
- ❑ It proposes **prescriptive methodological guidelines for obtaining the ontology network life cycle for a concrete ontology network, as part of the scheduling activity**. Additionally, a tool named gOntt, which supports such guidelines, is also provided.

- It offers **prescriptive methodological guidelines for reusing ontological resources at different level of granularity.**

All these contributions are backed up by a large number of experiments. These experiments, included in Chapter 11, show how the solutions proposed have been applied to real-world problems in the context of research projects.

3.4. Work Assumptions: Premises, Hypotheses and Restrictions

The work here described is based on the set of assumptions appearing below. These assumptions, which are divided into premises, hypotheses, and restrictions, help to explain the decisions taken for the development of the solutions and for the relevance of the contributions presented.

We have identified two premises, P1 and P2, both related to the three objectives of this thesis.

- P1. Some claims valid for software engineering are also valid for ontology engineering.

The IEEE standard glossary of Software Engineering terminology [IEEE, 1990] defines software as “computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system”. Ontologies are part (sometimes only potentially) of software products. Therefore, ontologies should be developed according to the standards generally proposed for software, which should be adapted to the special characteristics of ontologies [Fernández-López and Gómez-Pérez, 2002].

For such reasons, we will use and adapt Software Engineering terminology and methodological principles (taken from different IEEE standards) as basis for the current methodological research in the Ontology Engineering field.

- P2. Explaining the ontology building process with the same style and granularity than those methodologies for developing software benefits the understanding and adoption of ontology development methodologies.

Once the premises have been identified and presented, the set of hypotheses of our work are described. This set of hypotheses covers the main features of the solutions proposed.

Hypotheses H1 and H2 are related to objective O1, whereas hypothesis H3 is related to objective O2, and hypotheses H4, H5 and H6 are related to objective O3.

- H1. Despite the discrepancies found in the literature, ontology experts have reached a certain implicit consensus on what activities should be undertaken in ontology development. This implicit consensus can be made explicit by means of establishing a glossary of processes and activities in the Ontology Engineering field.
- H2. Despite the diversity of life cycle models proposed in the literature of Software Engineering and Knowledge Engineering fields, it should be pointed out that a collection of ontology network life cycle models, consisting of the waterfall model and the iterative-incremental model, is enough to allow ontology practitioners to develop ontology networks.
- H3. Although the current methodologies pose ontology development as a process that mainly takes into account the ontology being built, our experience shows that there are alternative scenarios in which ontologies are built by reusing and reengineering knowledge resources, merging ontological resources and/or localizing ontological resources. Thus, we can hypothesize that the ontology network development entails combining different scenarios, which in turn involve reusing ontological and non-ontological resources, reengineering, merging, restructuring, and localizing.

- H4. While so far there have not been substantial contributions to Grüninger and Fox's [Grüninger and Fox, 1994] proposal on the specification of ontology requirements based on CQs, a precise method, based on this approach, can be established. This method for specifying ontology network requirements can be applicable by software developers and ontology practitioners, and allows identifying the main terms to be included in the ontology.
- H5. Despite the fact that up to the present nobody has made a proposal on how to carry out the scheduling of ontology development projects, we here propose a set of identified processes and activities and life cycle models to bridge this gap. That means that it is possible to establish a clear, simple, and useful procedure to select and instantiate an ontology network life cycle model, as part of the scheduling activity. This procedure can be applicable by software developers and ontology practitioners.
- H6. It is possible to establish a clear, simple, and useful procedure to reuse ontological resources at different level of granularity. And again, as in the previous hypothesis, such a procedure can be applicable by software developers and ontology practitioners.

Finally, the following set of restrictions defines the limits of our contributions and allows determining future research objectives. Restriction R1 is related to O1; restrictions R2, R3, R4, R5, and R6 are related to O2 and O3; finally, restriction R7 is a general restriction.

- R1. This work presents a glossary that includes the current processes and activities used in ontology engineering. New terms are still appearing.
- R2. This work is focused on a subset of processes and activities performed during the ontology network development. Such a subset includes ontology requirements specification, scheduling, and ontological resource reuse. Out of the scope of this thesis are the rest of the processes and activities identified in the NeOn Glossary (included in Chapter 5).
- R3. This work presents methodological guidelines for ontology requirements specification, which are based on the so-called competency questions (CQ). It is out of the scope of this thesis the rest of the techniques identified in Section 2.5.2.1 for collecting requirements.
- R4. This work presents methodological guidelines for reusing ontological resources at different levels of granularity: (1) reusing domain ontologies and general or common ontologies as a whole and (2) reusing ontology statements. However, it does not provide guidelines for reusing ontology modules nor for reusing and reengineering knowledge resources that are not ontologies, since these issues are out of its scope.
- R5. This work does not propose guidelines for the collaboration and evolution of ontology networks nor for creating mapping among ontological resources, because they are out of the scope of this thesis.
- R6. This work presents methodological guides to develop ontology networks. However, it is assumed that the technological support to develop such ontologies is in charge of creating the meta-relationships for having ontology networks.
- R7. The evaluation of the work is restricted to the use of the results in real cases, which provide feedback, and to the execution of controlled experiments that use the results of this thesis.

In summary, the main aim of this thesis is to create the *NeOn Methodology for building ontology networks*, covering the drawbacks presented in the three methodologies analysed in Chapter 2, and benefiting from the advantages included in such methodologies. Concretely, we will take into account the proposal given by METHONTOLOGY and On-To-Knowledge about the use of competency questions for the ontology requirements specification activity within the methodological guidelines proposed for this activity and presented in this document. With respect to the reuse of ontologies, we will consider as starting point the list of activities proposed by METHONTOLOGY; however, we have tried to improve and extend them in order to provide the corresponding methodological guidelines in the NeOn Methodology.

4. Research Methodology

This chapter describes the research methodology adopted for creating the NeOn Methodology, a methodology for building ontology networks, which is presented in this thesis. Additionally, it explains the main requirements and conditions that have guided the creation of this methodology.

4.1. General Framework for Describing the NeOn Methodology

This thesis work is grounded on the following approaches, as presented graphically in Figure 16:

- *Software Engineering methodological work* (terminology, models, ideas, etc.). Methodological frameworks are widely accepted in different mature fields, like Software Engineering. Since single ontologies and ontology networks could be seen as “software artifacts” and as part of software products, the current software methodologies could be relevant to the Ontology Engineering field, or at least, be taken as inspiration.
- *Methodologies and methods for building ontologies*. We have employed METHONTOLOGY [Gómez-Pérez et al., 2003], On-To-Knowledge [Staab et al., 2001], DILIGENT [Pinto et al., 2004] and other methods available (e.g., [Grüninger and Fox, 1995]) to provide guidelines for carrying out a particular process or activity.
- *Practices and previous experiences*. Members of the NeOn consortium and of the OEG group (Ontology Engineering Group) have built a large number of ontologies in different domains across several European and National funded projects. From these experiences, we obtained a preliminary set of informal steps, which were refined, improved and completed to provide detailed methodological guidelines for each process or activity. As an example, we can mention the ontology requirements specification activity, whose guidelines are based on previous experiences drawn from the SEEMP project (FP6-27347).

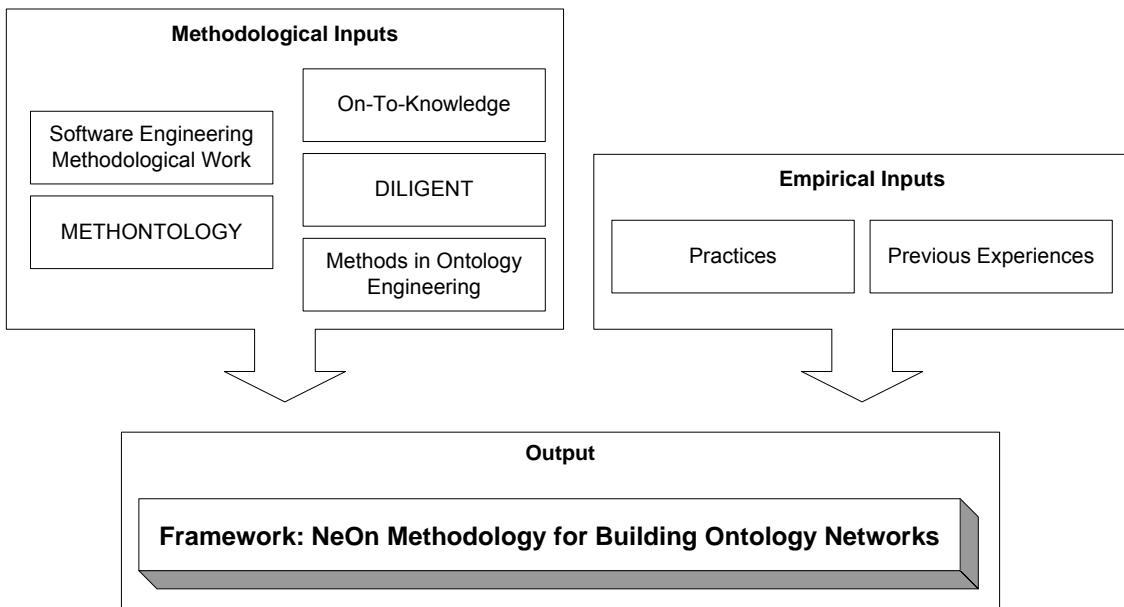


Figure 16. Inputs taken into account for creating the NeOn Methodology

For building the NeOn Methodology we used a “*divide and conquer*” strategy. That is, we decomposed the general problem to be solved in different subproblems. To obtain the solution to the general problem, i.e., the development of an ontology network, some solutions to the different subproblems should be combined. In our case, the subproblems are the nine scenarios identified in Chapter 6, which are composed of processes and activities. For each subproblem, we provide different strategies and alternatives to find the solution. Additionally, and whenever possible, we present the results through a Software Engineering approach with different levels of complexity, which facilitates a promptly assimilation by software developers and ontology practitioners.

The NeOn Methodology for building ontology networks is a *scenario-based methodology* that provides a set of nine scenarios that can be combined among them. Each scenario is decomposed in different processes and activities of those included in the NeOn Glossary. For each process or activity detailed methodological guidelines are provided. In this sense, the thesis is written with a *process or an activity-centric approach*, and in a *prescriptive way* rather than in a descriptive one.

- The scenarios are described in Chapter 6, where we have also included assumptions for the scenarios, prerequisite resources, sequence of the processes and activities to be carried out, and the main outcomes of applying the scenarios.

For identifying the scenarios (the different ways or paths we can follow to build ontologies and ontology networks), we founded on the different studies carried out to revise the state of the art of ontology development, on the building of ontologies in different international and national projects (Esperonto, Knowledge Web, SEEMP, etc.), and on the analysis of the NeOn European project use cases.

- The NeOn Glossary includes activities and processes composed, in turn, of activities based on the terminology included in Section 2.2. The activities can be divided into zero or more tasks. Tasks, which are the smallest units of work, are used to decompose activities and provide more detailed information. Figure 17 shows how the processes and activities are decomposed.

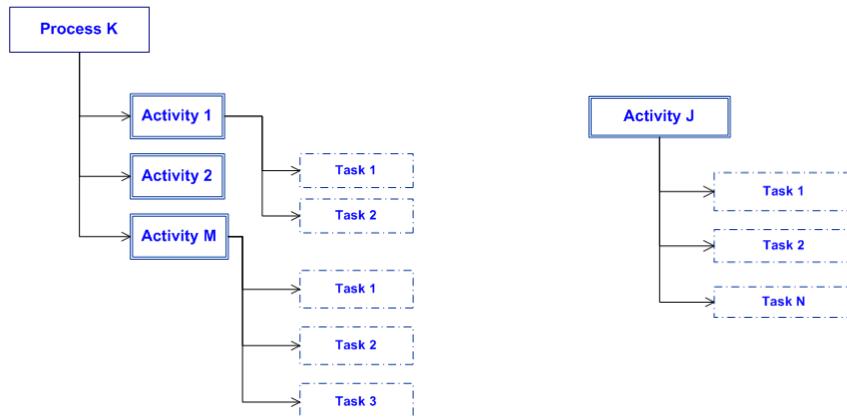


Figure 17. Process, activities, and tasks

For describing each of the processes and activities included in the NeOn Methodology presented in this thesis, we use the following structure:

- A general introduction to the process or activity. In this introduction we show the necessity and/or utility of the process or activity.
- The detailed guidelines proposed for carrying out the process or the activity, including the following fields:
 - Definition, which is taken from the NeOn Glossary of Processes and Activities and included in Section 5.3.

- Goal, which explains 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 the people or teams involved in the process or the activity.
- When, which explains in which moment the process or the activity should be carried out.

We provide all the aforementioned information in the so-called *Filling Cards*. These filling cards allow us to explain the information of each process and activity of the NeOn Methodology in a practical and easy way. Each card is filled according to the *filling card template* shown in Table 3.

- A graphical *workflow* that shows how the process or the activity should be carried out is also included. This workflow contains the inputs, outputs, actors involved, and details for carrying out the process or the activity in a prescriptive manner. Additionally, methods, techniques and tools supporting the process or the activity are proposed.
- Some examples explaining the guidelines proposed, whenever possible, are given. The examples are based on previous experiences and/or NeOn use cases.

Process or Activity Name	
<i>Definition</i>	
<input type="text"/>	
<i>Goal</i>	
<input type="text"/>	
<i>Input</i>	<i>Output</i>
<input type="text"/>	<input type="text"/>
<i>Who</i>	
<input type="text"/>	
<i>When</i>	
<input type="text"/>	

Table 3. Template for the process and activity Filling Card

4.2. Requirements of the NeOn Methodology for Building Ontology Networks

The main principles that guided the construction of the NeOn Methodology for building ontology networks were

1. The methodology should define each process or activity precisely; it should state clearly its purpose, inputs and outputs, the actors involved, when its execution is more convenient, and the set of methods, techniques and tools to be used.
2. The methodology should facilitate a promptly assimilation by software developers and ontology practitioners. That means that the methodology should be presented in a prescriptive way and non-oriented to researchers. It should be easy to understand and to learn in order to facilitate its success and its generalized use.

Additionally, according to [Paradela, 2001], any methodology must fulfil a set of characteristics that can be grouped into two main types:

- The suitability formal characteristics, that is, the **necessary conditions** (also called formal conditions) that any methodology in any activity domain must fulfil. These conditions are independent of the domain where the methodology is applied.
- The suitability material characteristics, that is, the **sufficient conditions** (also called material conditions) that the methodology must fulfil in the domain where it is applied. These conditions are specific to each methodology and are determined by factors such as the following: the domain where the methodology is applied; cases, situations or problems to be dealt with; characteristics of the material (economic, technological, etc.); human or temporal resources; etc.

We use here requirements instead of characteristics or conditions because (1) the characteristics abovementioned are needed for any methodology, and therefore they should be considered as requirements³⁰, and (2) the term condition is mainly used in the logic field, whereas the term requirement is used in the engineering field.

This section presents the necessary and sufficient requirements in the development of the NeOn Methodology for building ontology networks. The section is based on Paradela's conditions [Paradela, 2001], which were adapted whenever it was necessary.

4.2.1. Necessary Requirements

- **Generality.** A methodology should be general enough and should not be driven to solve ad-hoc cases or problems.
- **Completeness.** A methodology must consider all the cases presented and propose solutions to all of them.
- **Effectiveness.** A methodology should solve adequately the cases proposed that have a solution, with independency of the person that applies the methodology. Therefore, it should be more prescriptive than descriptive.
- **Efficiency.** A methodology must be efficient, that is, it should be able to achieve its objective or goal. This means that the methodology should allow the construction of ontologies.

³⁰ Requirement is something required, wanted or needed (definition at <http://www.merriam-webster.com/dictionary/requirement>).

- ❑ **Consistency.** A methodology must produce the same set of products for the same problem, independently of who applies the methodology. The content of such a set of products will be different depending on the problem treated.
- ❑ **Finiteness.** The number of the elements that compose a methodology and the number of activities must be finite, i.e., they should consume a reasonable period of time.
- ❑ **Discernment.** A methodology must be composed of a small set of structural, functional and representational components.
- ❑ **Environment.** Methodologies can be classified into scientific and technological ones. In scientific methodologies ideas are validated, whereas in technological ones artefacts are built and evaluated. A technological methodology must regard the life cycle of the product that is guiding its development.
- ❑ **Transparency.** A methodology must be like a white box that permits us to know in every moment the active processes or activities that are being performed, who is performing them, etc.
- ❑ **Essential Questions.** In each activity included in the methodology, the following six questions: "what", "who", "why", "when", "where", and "how" must be dealt with.

4.2.2. Sufficient Requirements

- ❑ **Domain or Scope.** The area in which the methodology can be applied.
- ❑ **Perspectives.** A methodology must facilitate its application following different approaches.
- ❑ **Understanding.** A methodology must be easy to understand and to learn in order to facilitate its success and its generalized use.
- ❑ **Usability.** The degree of difficulty in using the methodology must be minimal.
- ❑ **Grounded on existing practices.** A methodology must be founded on previous methodological and practical works.
- ❑ **Flexibility.** A methodology must be adaptable to concrete needs and users.
- ❑ **Tool-Independent.** A methodology must be independent of the technology available.

5. Glossary of Processes and Activities

5.1. Introduction

As stated in Chapter 3, one of the goals of this PhD work is to provide the identification and definition of the development process for ontology networks. In this regard, the following open research problem identified in Chapter 2 must be solved:

- Up to date, there is neither consensus nor standardization about the definitions of activities and processes that potentially could be carried out when single ontologies and ontology networks are developed.

Therefore, this chapter treats of the definition of ontology network development process and the agreement on the activities and processes involved in the ontology development and their definitions.

Despite the discrepancies found in the literature, ontology experts have reached a certain implicit consensus on which processes activities should be undertaken in ontology development and which are their definitions; thus, we should probe that

- It is possible to make such a consensus explicit and thus to obtain a glossary of processes and activities of the Ontology Engineering field.

5.1.1. Ontology Network Development Process

Ontologies are artefacts designed for the purpose of satisfying certain requirements and needs that are emerging in the real world.

The *ontology development process* can be defined (inspired by [IEEE, 1990] and according to the Software Engineering terminology) as the process by which user's needs are translated into an ontology.

Thus, we can define the ***ontology network development process*** as the process by which user's needs are translated into an ontology network. This means that the ontology network development process can be seen as a specific case of the software development process.

The main goal of this development process is to identify and define which processes and activities are carried out when ontology networks are built collaboratively. Such a development process is related to the consensus of processes and activities involved in the ontology development.

5.1.2. The Need for Reaching an Agreement on Processes and Activities

Ontology researchers, technology developers, and users have noticed, during the different meetings held, that they use different terminology to name the processes and activities involved in the ontology development; that is, that they have not reached a consensus yet on the definitions for ontology engineering processes and activities. For instance, it is not clear enough the difference between ontology modification [Stojanovic, 2004] and ontology update [Stojanovic et al., 2002], whereas other activities have multiple definitions in natural language (e.g., ontology merging [Fernández-López et al., 1997; Kalfoglou and Schorlemmer, 2003; Kotis and Vouros, 2004]). We have also observed that in the last recent years, new processes and activities have emerged when large ontologies are built in complex settings. This situation is the result of a lack of standardization in the ontology engineering terminology, which clearly contrasts with the Software Engineering field

that boasts the *IEEE Standard Glossary of Software Engineering Terminology* [IEEE, 1990], a consensual glossary.

Thus, in order to unify the terminology used by the NeOn partners, it was decided by members of the NeOn consortium³¹ to build the **NeOn Glossary of Processes and Activities**. The idea here was to reach consensus on the identification and definition of the processes and activities involved when developing ontology networks. The glossary, which is the result of the agreement, is a key asset for creating the NeOn Methodology framework.

5.1.3. Chapter Structure

This chapter presents the following issues:

1. The consensus reaching process for the identification and definition of the processes and activities involved in the ontology network development process (Section 5.2).
2. The NeOn Glossary of Processes and Activities (Section 5.3), which identifies and defines the processes and activities involved in ontology network construction. Definitions in the glossary have been collaboratively built and agreed upon by all NeOn partners through a long and difficult consensus reaching process, as explained in Section 5.2.
3. Processes and activities are classified into those required for the development of ontology networks and those applicable (depending on the cases), but not required, and, therefore, non-essential or dispensable. This classification is summarized in the “Required-If Applicable” table (Section 5.4).
4. The processes and activities included in the NeOn Glossary of Processes and Activities are classified into groups according to the IEEE classification [IEEE, 2006] (Section 5.5).
5. The relation between processes and activities included in the NeOn Glossary and the ontology networks (Section 5.6).

5.2. Consensus Reaching Process

Within the NeOn consortium, it was decided to build the **NeOn Glossary of Processes and Activities** with the purpose of unifying the terminology used by the NeOn partners. The idea was to reach a consensus on the identification and definition of the processes and activities involved when developing ontology networks.

The definitions in the glossary have been collaboratively built and agreed upon by all NeOn partners. To reach a consensus in the process and activity terminology, we decided to use the wiki technology [Leuf and Cunningham, 2001] because it supports a higher level of consensus building by community members; thus, if a user disagrees with a statement, he can very easily modify it, delete it, comment it, etc. [Viégas et al., 2004]. Then, we created a non public space in the NeOn wiki where the partners involved in the NeOn project could discuss the ontology engineering terminology, express and exchange different opinions, and reach a final agreement. We also used meetings and mailing lists for agreeing on the process and activity definitions at the final stages.

This section sketches the roles to be played in the NeOn consortium and the overall process involved in reaching a consensus on the processes and activities for developing ontology networks. During the consensus reaching process, we have tried to agree on the list of processes and activities and on the process and activity definitions.

³¹ <http://www.neon-project.org/>

Roles in the process: a varied number of skilled people (known as the ‘NeOn Glossary’ team), geographically dispersed, collaborated on our consensus reaching process. This team has a well-balanced and representative participation of the different people involved in the process: ontology engineers, ontology editors, and users within the NeOn project. Its different members had the following specific roles (or functions):

- The author of this PhD thesis, from the *Universidad Politécnica de Madrid* (UPM), led the whole consensus reaching process.

She, personally, was in charge of creating the NeOn wiki page. She also created and published a template, in the wiki, for gathering information about processes and activities. Finally, she created the initial list of processes and activities with definitions, which are founded on the study of the state of the art in ontology engineering.

- The following organisms introduced several comments in the process and activity definitions: *Consiglio Nazionale delle Ricerche* (CNR), United Nations Food & Agriculture Organization (FAO), *Institut National de Recherche en Informatique et en Automatique* (INRIA), Intelligent Software Components, S.A. (iSOCO), Josef Stefan Institute (JSI), The Open University (OU), *Universitaet Karlsruhe* (UKARL), UPM, and University of Sheffield (USFD).
- Some of these organisms, i.e., CNR, FAO, JSI, OU, UKARL, UPM, and USFD, participated in some of the ad-hoc meetings carried out to reach a consensus.

Process Stages: Before beginning the consensus reaching process, the meanings of consensus and consensus reaching process were explained to the people concerned.

- Consensus is defined as a state of mutual agreement among members of a group where all opinions have been heard and addressed to the satisfaction of the group [Saint and Lawson, 1994].
- Consensus reaching process is defined as a dynamic and iterative process composed of several rounds, where the experts express and discuss their opinions in order to reach the maximum agreement about a set of alternatives before taking a decision [Herrera-Viedma et al., 2005].

Additionally, the ‘NeOn Glossary’ team explained and reviewed the process proposed for achieving consensus on the processes and activities. Then, they agreed on a targeted time period (one year) for reaching a consensus. Finally, they followed the general process, shown in Figure 18, to achieve consensus on the identification and definition of the processes and activities involved in the development of ontology networks, which they finally succeeded after the third round.

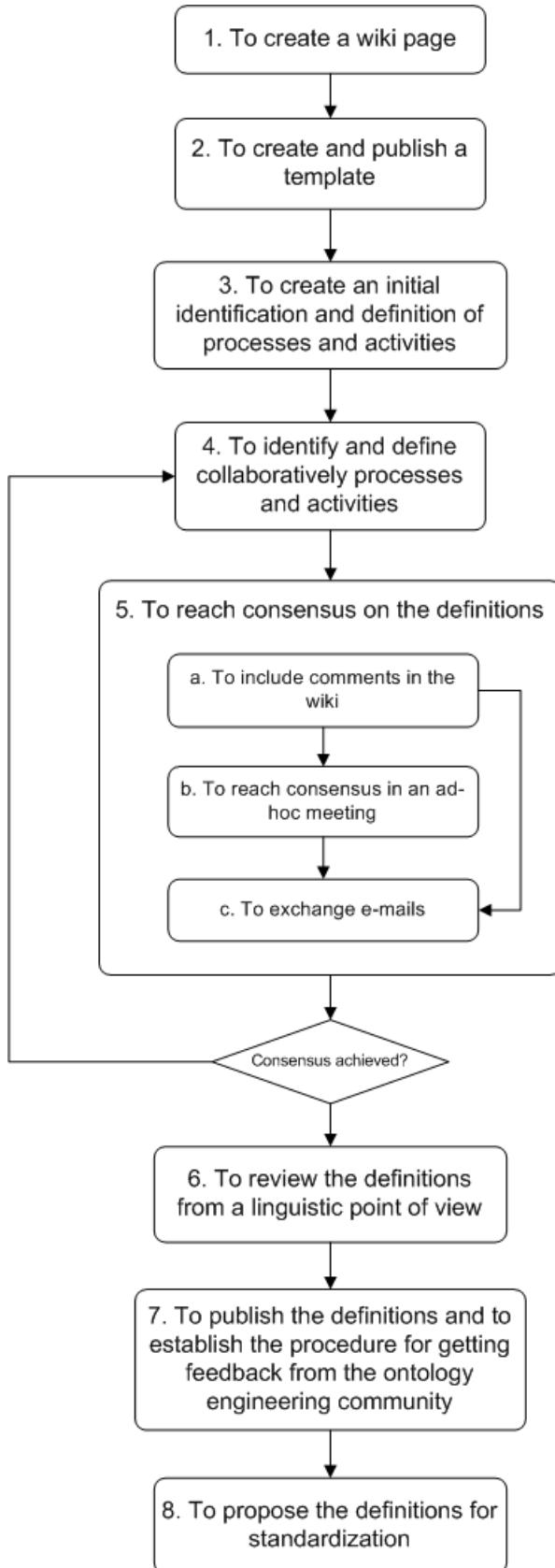


Figure 18. Consensus reaching process

The detailed stages of the consensus reaching process for the NeOn Glossary of Processes and Activities are the following:

1. *To create a NeOn wiki page dedicated to the consensus reaching process, within the NeOn consortium.* We used this wiki page, which was created in mid-June 2006, within the NeOn consortium to build collaboratively and by consensus the NeOn Glossary of Processes and Activities. This technology was chosen because wiki web sites facilitate community members the reach of a consensus [Viégas et al., 2004]. At that moment, the argumentation tool Cicero³² [Dellschaft et al., 2008] was not available for the consensus reaching process.
2. *To create a template for gathering information about processes and activities and to publish this template in the dedicated wiki page.* The template shown in Table 4 gathers not only possible process and activity definitions but also other general information useful for the methodology, for example, the input and output of the process or activity. It was built and published in mid-June 2006.

Template Slot	Description
<i>Process or Activity Name</i>	Name of the process or activity
<i>Definition</i>	One or several natural language (NL) definitions (with the corresponding references)
<i>Type of Process/Activity according to IEEE</i>	<p>In the Software Engineering field, activities are grouped administratively into five main activity groups [IEEE, 2006]. Based on that, the following groups are proposed:</p> <ul style="list-style-type: none"> (a) Ontology Management (b) Ontology Pre-Development (c) Ontology Development (d) Ontology Post-Development (e) Ontology Support Activity <p>The type for a concrete process or activity should be unique</p>
<i>Input</i>	A list of the information required to be input of the process or activity
<i>Output</i>	A list of the information that is required to be output of the process or activity
<i>References</i>	References for the NL definitions
<i>Comments</i>	Other comments about the process or activity

Table 4. Template for processes and activities in the NeOn Glossary

3. *To create an initial list of processes and activities with definitions.* An initial NeOn Glossary of Processes and Activities (the initial identification and definition of the main activities to be included in the ontology network development process) was made available in the wiki following the template presented in Table 4. This initial glossary was created having as starting point the ontology processes and activities found in the sources that follow: METHONTOLOGY [Gómez-Pérez et al., 2003], On-To-Knowledge [Staab et al., 2001], DILIGENT [Pinto et al., 2004], NeOn use cases [Iglesias et al., 2006; Gómez-Pérez et al., 2006], the Semantic Web Framework (SWF) from Knowledge Web Network of Excellence [García-Castro et al., 2007], and inputs from ontology engineering papers and ontology experts, as can be seen in Figure 19. The initial list of processes and activities, definitions and groups contained 39 processes and activities, and was uploaded on the wiki in mid-June 2006.

³² <http://cicero.uni-koblenz.de>

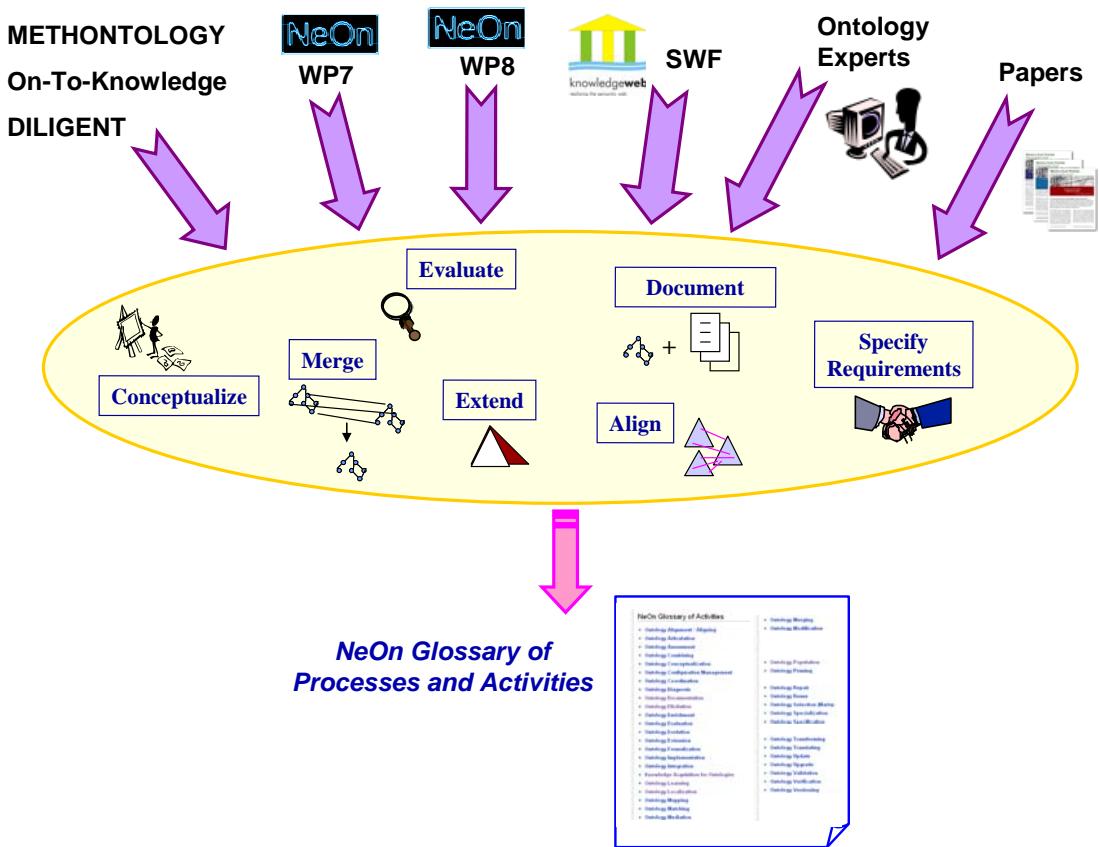


Figure 19. Approach to creating initially the NeOn Glossary of Processes and Activities

4. *To identify and define collaboratively processes and activities in the wiki, according to the initial list.* The ‘NeOn Glossary’ team was given total freedom to incorporate more processes and activities and/or definitions in the initial glossary, and also to include more general information about the process or activity (such as input and output, classification following the groups based on IEEE [IEEE, 2006], etc.).
 5. *To reach a consensus on process and activity definitions.* For this stage, the team used the wiki, held ad-hoc meetings and interchanged e-mails. Then, we adopted the following process:
 - a. *To include comments in the wiki.* The ‘NeOn Glossary’ team introduced their comments for each process and activity in a table where the institutions participating appeared. This table was uploaded on the wiki at mid-October 2006. In the table participants could include comments about activity definitions, i.e., the definition they would prefer if there were more than one; they could also add a proposal for a new definition. Figure 20 shows an example of a table of preferences for the ‘knowledge acquisition’ activity.

Preference Table

Partner Name	Partner Preferences
CNR	<p>Definition: agreement with OU's definition -- two remarks: 1) does knowledge acquisition yield "formalized knowledge"? The original definition aimed at a pre-development activity, while OU's definition shifts the balance on development and/or use (population being use rather than development) 2) in d2.1.1 OU's definition seems to correspond to a number of functionalities (learning, upgrading database content, data annotation). Activity Group: Development and Maintenance and Use (by OU's definition). Type of Activity according to IEEE: Ontology Management and Development (by OU's definition). Input: Knowledge Resource or Ontology Element (as in C-ODO). Output: Ontology Element (as in C-ODO).</p>
OU	<p>For me knowledge acquisition is a rather broad task that encompasses a couple of specialised tasks which seem to be mixed up in the current definition. On one hand "knowledge elicitation" is often used when knowledge is derived from a domain expert. Then, ontology learning (acquiring TBOX) and ontology population (acquiring ABOX) are seen as two aspects of knowledge acquisition that involve extraction of ontologies (resp. instances) from structured, semi-structured and unstructured data sources.</p> <p>Therefore, I would rephrase the definition as follows:</p> <p>"Knowledge acquisition stands for a family of tasks that deal with deriving formalized knowledge from a variety of sources. In particular, we distinguish knowledge elicitation when knowledge is acquired from a domain expert. Then, ontology learning and ontology population are knowledge acquisition tasks that rely on (semi)automatic methods to transform unstructured, semi-structured and structured data sources into ontologies and their associated instance data respectively."</p> <p>Development Development Input: some source of knowledge (expert, documents, etc) Output: formalized knowledge</p>

Figure 20. Knowledge acquisition activity: table of preferences

- b. *To reach a consensus in ad-hoc meetings and via e-mail.* Two ad-hoc meetings to review and agree on the definitions of processes and activities of the NeOn Glossary were held; the first one took place at the Bled plenary meeting (January 23rd 2007), and the second at the Dubrovnik plenary session (June 29th 2007).

During these meetings, the team took into account some informal rules for accepting or rejecting a concrete process and activity definition; these rules were the following:

- If the team's comments were generally positive and no major objections were raised, then the definition was considered as final.
- If the team's comments were generally positive, but someone had a major objection to the definition, the definition was modified until no major objection was encountered.
- If the team's comments were generally negative, the definition was ruled out.
- If the team's comments were mixed, there were three possibilities:
 - discussions continued until positive or negative results were achieved;
 - discussions were postponed until the next meeting; and
 - the issue was postponed until more information was available in the wiki.
- If discussions seemed to be going on forever without any possibility of reaching an agreement, the team could
 - decide to drop the definition, or the process or activity; or
 - move onto approval by voting the definition. The selected voting procedure was based on absolute majority.

During the ***first round*** of the process, which took place in Bled (January 23rd 2007), the team reached consensus on 25 definitions out of 46 identified processes and activities. The partners participating in this meeting were JSI, OU, UKARL, and UPM. Figure 21 shows the consensual definitions during the first round. Some processes and activities have been classified following a “sub-type” approach, such as ontology validation and ontology verification, which can be seen as ontology evaluation activities. This classification can be seen also in Figure 21.

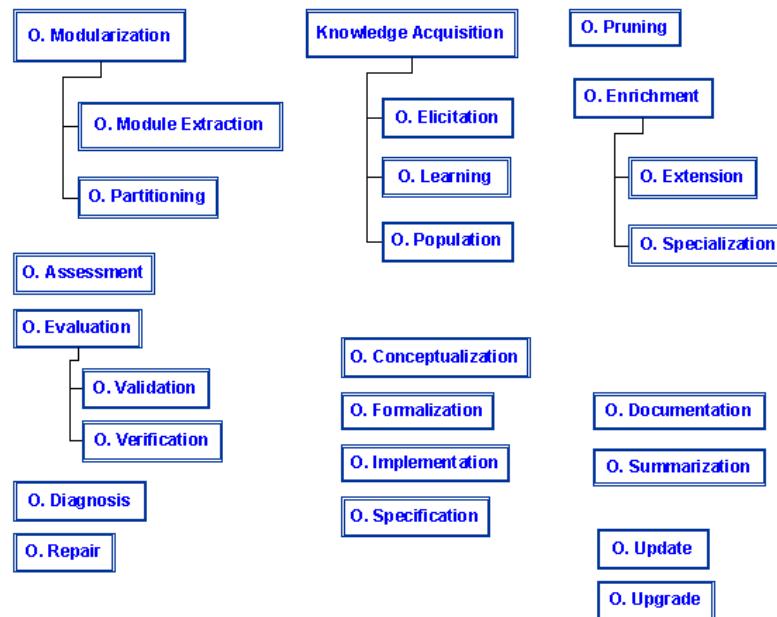


Figure 21. Processes and activities agreed upon in the first round, on 23rd January at Bled

During the ***second round*** of the process, which took place in Dubrovnik (29th June 2007) there were 61 identified processes and activities, 36 of them without consensus. At this meeting we reached a consensus on 4 activity definitions; we also agreed on deleting 5 activities, and almost achieved a consensus on 8 activity definitions (waiting for concrete feedback on minor details and to be agreed upon in the next round). The partners participating in this meeting were CNR, FAO, JSI, UKARL, UPM, and USFD. Figure 22 shows the activity definitions agreed upon during the second round. The 5 deleted activities appear in Figure 22 surrounded by a dotted line, whereas the synonymous activities are linked by a bi-directional arrow.

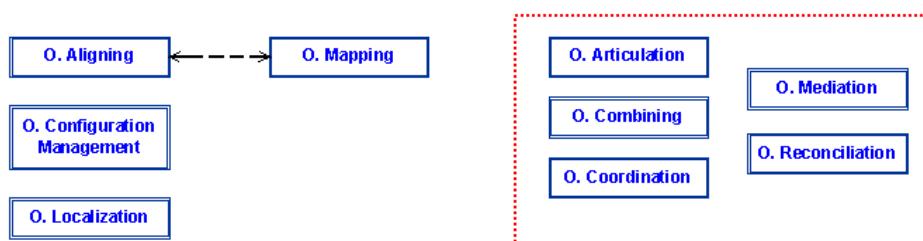


Figure 22. Activities agreed upon in the second round, on 29th June at Dubrovnik

Furthermore, in the ***third round***, it was decided to finish via email the consensus process on the remaining 27 activity definitions. Therefore, in July 2007, several experts provided feedback on the 8 abovementioned activities that did not reach a final consensus; on the other hand, UKARL and UPM exchanged several emails and included several comments in the wiki with the following results: 22 processes and activity definitions agreed upon and 5 activities deleted. Figure 23 shows the processes and activities agreed upon during the third round. The 5 activities deleted appear surrounded by a dotted line in Figure 23.

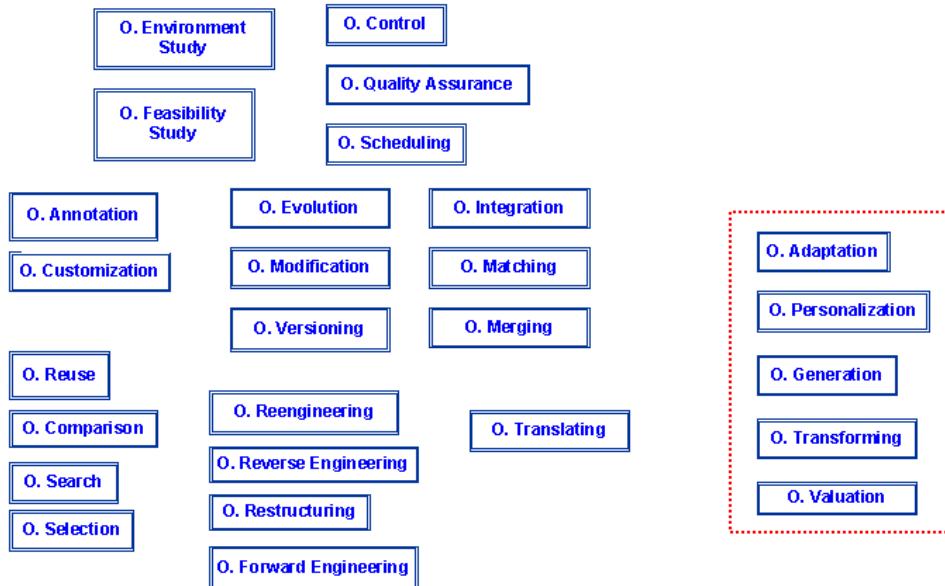


Figure 23. Processes and activities agreed upon in the third round during July 2007

- c. *To exchange e-mails.* Several e-mails were sent to the NeOn consortium to encourage other WPs to put forward their ideas.

Once the two ad-hoc meetings were held, WP5 partners decided that the existing definitions should be reviewed by other WPs within the NeOn consortium. Then, we identified which processes and activities were related to which WPs, taking into account the WP main objectives, and we created and sent a table for obtaining feedback from WPs about the different related processes and activities. Several processes and activities were reviewed by more than one WP.

The comments received were mainly about differentiating or making equal some process and activity definitions, including missing aspects in some activity definitions, and deleting some processes and activities. All the comments received were considered during the second and third round of the process.

- 6. *To review the definitions from a linguistic point of view.* The definitions and their corresponding comments were checked and reviewed from a linguistic point of view by an expert terminologist.

7. To publish the definitions in the NeOn website³³ and in a public wiki page and to establish the procedure for getting feedback from the ontology engineering community, using the argumentation tool Cicero.

After finishing the NeOn Glossary of Processes and Activities, we published it in the NeOn website and delved into the idea of obtaining feedback from the ontology engineering community (outside NeOn). Thus, we created a public wiki page with all the processes and activities of the glossary with the aim of getting comments from other people. We used the argumentation tool Cicero [Dellschaft et al., 2008], which is already a wiki, so that the OE community could comment on the activity definitions for about a year. Our long term goal is to have a more complete and consensual glossary that could become the terminological reference in the field.

8. To propose the standardization of the NeOn Glossary of Processes and Activities. If the Ontology Engineering community supports the activity of providing feedback on the NeOn Glossary, then we could think of approaching IEEE or W3C for the standardization of the NeOn Glossary of Processes and Activities.

Figure 24 shows the history of the consensus reaching process (from stage 1 to stage 5.b), including the main results in each time point.

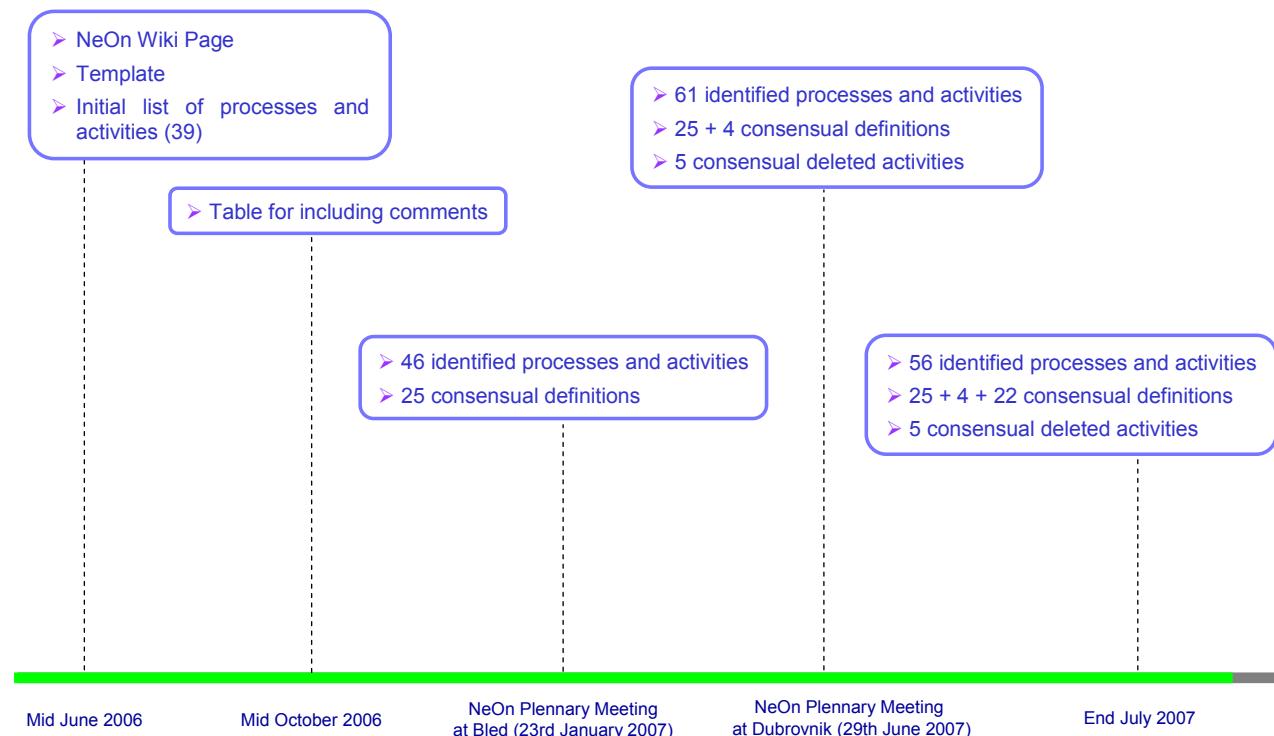


Figure 24. History of the consensus reaching process

³³ <http://www.neon-project.org/>

5.3. NeOn Glossary of Processes and Activities

According to ISO 12200:1999 [ISO, 1999] and ISO 1087-1:2000 [ISO, 2000], a **glossary** can be defined as

- a. A terminological dictionary containing the terminology of a specific subject field or of related fields and based on terminology work [ISO, 1999].
- b. A terminological dictionary that contains designations and definitions from one or more specific subject fields. The vocabulary may be monolingual, bilingual or multilingual [ISO, 2000].

According to [Landau, 1984] there are three basic principles for **defining** a term: “avoid circularity, define every word in a definition, and make sure that every word’s definition says what the word means”. However, applying these basic rules does not necessarily produce good definitions.

The following principles, widely accepted in lexicographic practice, are also common:

- Conciseness, i.e., every definition should say the most in the least number of words.
- Clarity, or avoidance of ambiguity, i.e., words should be used unambiguously.
- Appropriateness, i.e., the definition should be appropriate to the target reader.
- Priority of essential traits, i.e., a definition should highlight the essential features of meaning.

Following the aforementioned definitions and principles, the *Neon Glossary of Processes and Activities* collects the main processes and activities involved in the ontology network development process (within the specific field of ontology engineering) and provides some commonly agreed natural language definitions and explanations. The vocabulary included in the glossary is monolingual (English).

As a result of the process explained in Section 5.2, we obtained the main processes and activities to be performed in the ontology network development process. Fifty one activities have been identified and defined as part of the NeOn Glossary of Processes and Activities. Moreover, other processes and activities and their corresponding definitions (e.g., *Non-Ontological Resource Reengineering* and *Non-Ontological Resource Reuse*) have been also included as a result of identifying the different scenarios for building ontology networks (presented in Chapter 6) and revising the glossary.

5.3.1. Definitions in the NeOn Glossary

The **NeOn Glossary of Processes and Activities**, which is ordered alphabetically, includes 59 processes and activities. Three situations can be distinguished here:

- a. Some of the processes and activities have maintained their definition taken from the literature (as shown in the list below). This is the case, for example, of ontology configuration management, ontology evolution, and ontology formalization.
- b. For most of the processes and activities, the definition has been changed and adapted according to past definitions found in the literature and based on NeOn partners’ comments and discussions. This is the case, for example, of ontology conceptualization, ontology documentation, ontology matching, and ontology selection.
- c. New processes and activities and their definitions have been created during the consensus building process, either because the process or activity did not previously exist in the literature, or because the process or activity had not been defined, as it happens in the following processes and activities: ontology annotation, ontology comparison, ontology

customization, ontology diagnosis, ontology elicitation, ontology enrichment, ontology extension, ontology learning, ontology localization, ontology module extraction, ontology partitioning, non-ontological resource reengineering, ontology repair, non-ontological resource reuse, ontology search, and ontology upgrade.

The terms (and their definitions) included in the NeOn Glossary are

- **Ontology Aligning.** This term refers to the activity of finding the correspondences between two or more ontologies and storing/exploiting them. A synonym for this activity is Ontology Mapping.
- **Ontology Annotation.** It refers to the activity of enriching the ontology with additional information, e.g., metadata or comments.
- **Ontology Assessment.** It refers to the activity of checking an ontology against the user's requirements, such as usability, usefulness, abstraction, quality.
- **Ontology Comparison.** It refers to the activity of finding differences between two or more ontologies or between two or more ontology modules.
- **Ontology Conceptualization.** It refers to the activity of organizing and structuring the information (data, knowledge, etc.), obtained during the acquisition process, into meaningful models at the knowledge level and according to the ontology requirements specification document. This activity is independent of the way in which the ontology implementation will be carried out.
- **Ontology Configuration Management** [Gómez-Pérez et al., 2003]. It refers to the activity of recording all the versions of the documentation, software and ontology code, and of controlling the changes.
- **Control** [Gómez-Pérez et al., 2003]. It refers to the activity of guaranteeing that the activities scheduled in the ontology development process are completed and performed in the manner intended.
- **Ontology Customization.** It refers to the activity of adapting an ontology to a specific user's needs.
- **Ontology Design Pattern Reuse.** It refers to the process of using available ontology design patterns in the solution to different modelling problems during the development of new ontologies.
- **Ontology Diagnosis.** It refers to the activity of identifying parts of the ontology directly responsible for incorrectness and incompleteness. Ontology diagnosis is triggered by ontology validation.
- **Ontology Documentation.** It refers to the collection of documents and explanatory comments generated during the entire ontology building process.

Examples of the documents external to the implemented ontology include ontology requirement specification document, sources used for acquiring knowledge, ontology conceptualization document, design and decision criteria, etc.

The information inside the implemented ontology includes natural language comments, ontology metadata, and implementation code.

In summary: anything that could be useful to help users, who did not build the ontology, to understand and learn how the ontology was built. Note that the level of granularity of descriptions can help the understanding of the ontology.

- **Ontology Elicitation.** It is a knowledge acquisition activity in which conceptual structures (e.g., T-Box) and their instances (e.g., A-Box) are acquired from domain experts.
- **Ontology Enrichment.** It refers to the activity of extending an ontology with new conceptual structures (e.g., concepts, roles and axioms).
- **Ontology Environment Study.** It refers to the activity of analyzing the environment in which the ontology is going to be developed.
- **Ontology Evaluation.** It refers to the activity of checking the technical quality of an ontology against a frame of reference.
- **Ontology Evolution** [Stojanovic, 2004]. It refers to the activity of facilitating the modification of an ontology by preserving its consistency.

Ontology Evolution can be seen as a consequence of different activities during the development of the ontology.

- **Ontology Extension.** It is an ontology enrichment activity for stretching the ontology in width.
 - **Ontology Feasibility Study** [Gómez-Pérez et al., 2003]. It refers to the activity of answering questions such as “Is it possible to build the ontology? and/or “Is it suitable to build the ontology?”
 - **Ontology Formalization** [Gómez-Pérez et al., 2003]. It refers to the transformation of a conceptual model into a formal or semi-computable model according to a knowledge representation paradigm (e.g., description logics, frames, and rules).
 - **Ontology Forward Engineering** [Gómez-Pérez and Rojas-Amaya, 1999]. It refers to the activity of outputting a new implementation of the ontology on the basis of the new conceptual model.
 - **Ontology Implementation.** It refers to the activity of generating computable models according to the syntax of a formal representation language (e.g., RDF(S), OWL, and FLogic).
 - **Ontology Integration.** It refers to the activity of including one ontology in another ontology.
 - **Knowledge Acquisition for Ontologies.** It comprises activities for capturing knowledge (e.g., T-Box and A-Box) from a variety of sources (e.g., documents, experts, data bases, etc.). We can distinguish between Ontology Elicitation, Ontology Learning and Ontology Population.
 - **Ontology Learning.** It is a knowledge acquisition activity that relies on (semi-) automatic methods to transform unstructured (e.g., corpora), semi-structured (e.g., folksonomies and html pages) and structured data sources (e.g., data bases) into conceptual structures (e.g., T-Box).
 - **Ontology Localization.** It refers to the adaptation of an ontology to a particular language and culture.
 - **Ontology Mapping.** It refers to the activity of finding the correspondences between two or more ontologies and storing/exploiting them. A synonym for this activity is Ontology Aligning.
 - **Ontology Matching.** It refers to the activity of finding or discovering relationships or correspondences between entities of different ontologies or ontology modules.
- Ontology Matching can be seen as the first stage of Ontology Aligning.
- **Ontology Merging.** It refers to the activity of creating a new ontology or an ontology module from two or more, possibly overlapping, source ontologies or ontology modules.
 - **Ontology Modification** [Stojanovic, 2004]. It refers to the activity of changing the ontology without considering the consistency.

- **Ontology Modularization.** It refers to the activity of identifying one or more modules in an ontology with the purpose of supporting reuse or maintenance.
We can make distinctions between: Ontology Module Extraction and Ontology Partitioning.
- **Ontology Module Extraction.** It refers to the activity of obtaining from an ontology some concrete modules that could be used for a particular purpose (e.g., to contain a particular sub-vocabulary of the original ontology).
- **Ontology Module Reuse.** It refers to the process of using available ontology modules in the solution of different problems
- **Ontology Partitioning.** It refers to the activity of dividing an ontology into a set of (not necessarily disjoint) modules that together form an ontology but that can be treated separately.
- **Ontology Population.** It is a knowledge acquisition activity that relies on (semi-) automatic methods to transform unstructured (e.g., corpora), semi-structured (e.g., folksonomies and html pages) and structured data sources (e.g., data bases) into instance data (e.g., A-Box).
- **Ontology Pruning.** It refers to the activity of discarding conceptual structures (e.g., part of T-Box) of a given ontology that are not or no longer relevant.
Pruning can be used in combination with ontology learning methods to discard potentially irrelevant learned concepts/relations.
- **Ontology Quality Assurance** [Gómez-Pérez et al., 2003]. It refers to the activity of assuring that the quality of each and every process carried out and each an every product built (ontology, software and documentation) is satisfactory.
- **Ontology Reengineering** [Gómez-Pérez and Rojas-Amaya, 1999]. It refers to the process of retrieving and transforming a conceptual model of an implemented ontology into a new, more correct and more complete conceptual model, which is re-implemented.
- **Ontology Repair.** It refers to the activity of solving errors (incompleteness, incorrectness) in the ontology. This activity is triggered by ontology diagnosis.
- **Ontology Requirements Specification.** It refers to the activity of collecting the requirements that the ontology should fulfil (for example, reasons to build the ontology, identification of target groups and intended uses). Such requirements may be reached through a consensus process.
- **Non-Ontological Resource Reengineering.** It refers to the process of retrieving and transforming an non-ontological resource³⁴ (data bases, controlled vocabularies, etc.) into an ontology.

This process could be compared with the ontology learning activity with the difference that in this activity the knowledge is only transformed into conceptual structures, whereas in the process of reengineering non-ontological resources the sources can be transformed into conceptual structures and instance data.

- **Non-Ontological Resource Reverse Engineering.** It refers to the activity of analyzing a non-ontological resource in order to identify its underlying components and creating a representation of the resource at higher levels of abstraction.
- **Non-Ontological Resource Transformation.** It refers to the activity of generating an ontological model at different levels of abstraction from the non-ontological resource.

³⁴ Non-ontological resource is defined as a knowledge resource whose semantics has not yet been formalized by means of an ontology. Elements in this set are glossaries, dictionaries, lexicons, classification schemes and taxonomies, and thesauri.

- **Ontology Restructuring** [Gómez-Pérez and Rojas Amaya, 1999]. It refers to the activity of correcting and reorganizing the knowledge contained in an initial conceptual model, and detecting missing knowledge.

This activity contains two different tasks: analysis and synthesis. The goal of the analysis is to evaluate the ontology technically, that is, to check that the hierarchy of the ontology and its classes, instances, relations and functions are complete (contain all the definitions required for the domain of chemical substances), consistent (there are no contradictions in the ontology and with respect to the knowledge sources used), concise (there are no explicit and implicit redundancies) and syntactically correct. On the other hand, the synthesis task seeks to correct the ontology after the analysis phase and to document any changes made.

- **Non-Ontological Resource Reuse.** It refers to the process of taking the available non-ontological resources (data bases, controlled vocabularies, etc.) for the development of ontologies.
- **Ontological Resource Reuse.** It is defined as the process of using available ontological resources³⁵ (ontologies, modules, statements, or ontology design patterns) for solving different problems (e.g., the development of different ontology-based applications, the activity of ontology aligning (as background knowledge)
- **Ontology Reuse.** It refers to the process of using available ontologies for solving different problems.
- **Ontology Reverse Engineering** [Gómez-Pérez and Rojas-Amaya, 1999]. It refers to the activity of outputting a possible conceptual model on the basis of the code in which the ontology is implemented.
- **Scheduling** [Gómez-Pérez et al., 2003]. It refers to the activity of identifying the different activities and processes to be performed during the ontology development, their arrangement, and the time and resources needed for their completion.
- **Ontology Search.** It refers to the activity of finding candidate ontologies or ontology modules to be reused.
- **Ontology Selection.** It refers to the activity of choosing the most suitable ontologies or ontology modules among those available in an ontology repository or library, for a concrete domain of interest and associated tasks.
- **Ontology Specialization.** It is an ontology enrichment activity for extending the ontology in depth.
- **Ontology Statement Reuse.** It refers to the process of using available ontology statements in the solution of different problems
- **Ontology Summarization.** It refers to the activity of providing an abstract or summary of the ontology content.

The summary can include, for example, a couple of top level concepts in the ontology class hierarchy (perhaps a graphical representation of these top-level concepts and the links between them).

- **Ontology Translation.** It refers to the activity of changing the representation formalism or language of an ontology to another.
- Ontology Translation can be part of an ontology reengineering process.
- **Ontology Update.** It refers to minor changes carried out in an ontology that could not be considered an upgrade.

³⁵ *Ontological resource* is defined as 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.

- **Ontology Upgrade.** It refers to the activity of replacing an ontology with a new version.
- **Ontology Validation.** It is the ontology evaluation that compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize.
It answers the question “Are you producing the right ontology?”
- **Ontology Verification.** It is the ontology evaluation that compares the ontology against the ontology requirement specification document (ontology requirements and competency questions), thus ensuring that the ontology is built correctly (in compliance with the ontology requirements specification).
It answers the question “Are you producing the ontology right?”
- **Ontology Versioning** [Stojanovic, 2004]. It refers to the activity of handling ontology changes by creating and managing different versions of the ontology.

The 59 definitions and their corresponding comments included in the NeOn Glossary of Processes and Activities have been checked and reviewed, from a linguistic point of view, by an expert terminologist. Figure 25 shows the whole NeOn Glossary of Processes and Activities, where the different types of arrows have the following meanings: (1) arrows with dashed lines mean “a process is divided into activities”, (2) arrows with solid lines mean “type of”, and (3) arrows with dotted lines mean “synonymy”.

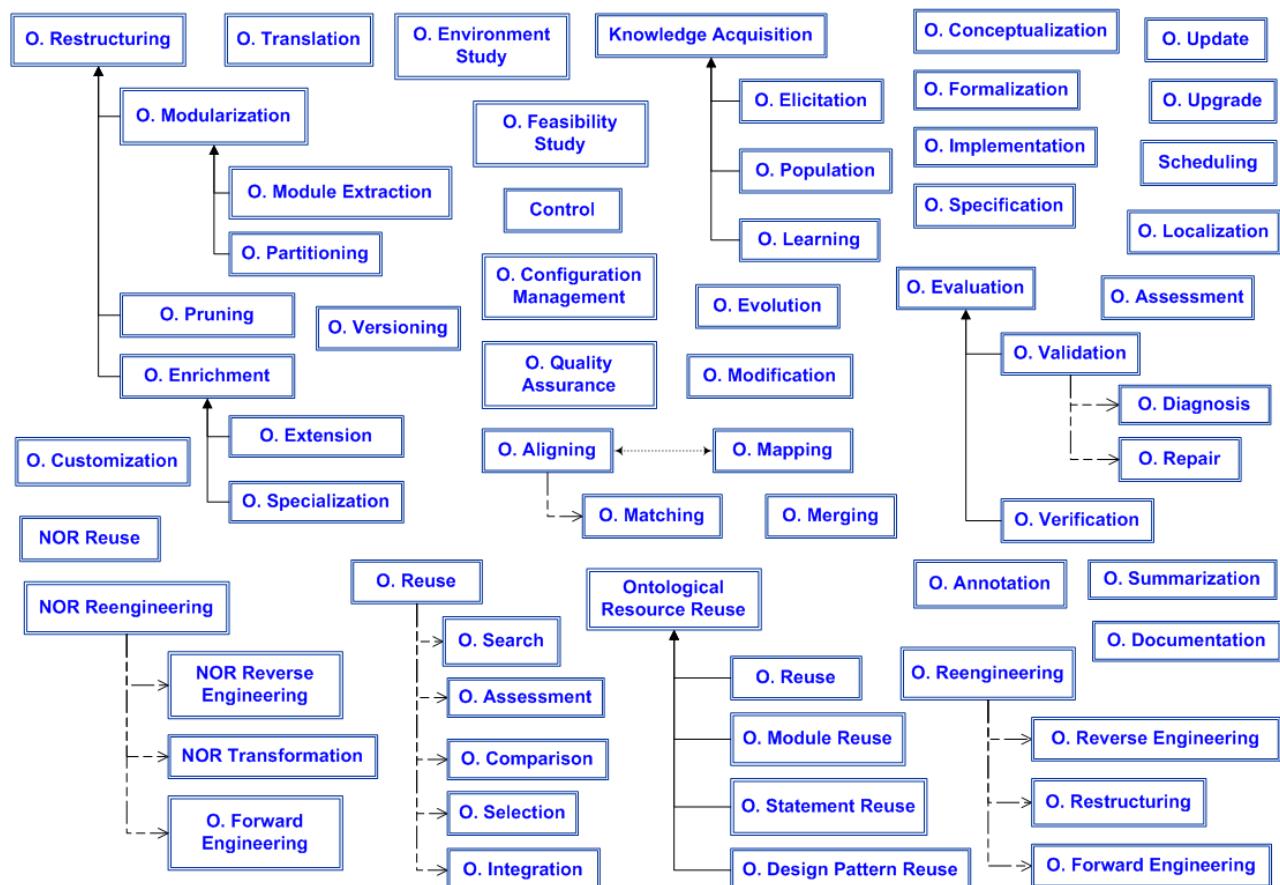


Figure 25. NeOn Glossary of Processes and Activities

5.3.2. Getting Feedback from the Ontology Engineering Community

After having achieved consensus on the definitions for the processes and activities involved in the development process of ontology networks within the NeOn consortium, and having built the NeOn Glossary of Processes and Activities, one important goal we have is to make the NeOn Glossary as much complete as possible, and so to have a commonly agreed collection of the most important terms in the field of Ontology Engineering. Because the terminology is continually evolving, it is essential to establish an approach to collecting feedback of the terms and the suggestion of new terms, which will maintain the NeOn Glossary up to date. For this reason, we have promoted an initiative to obtain feedback from ontology engineering people outside the NeOn project with the aim of having richer definitions and a more complete glossary.

For this purpose, we set up a *discussion project*³⁶ in the Cicero argumentation tool. The *discussion project* is used for collecting the feedback from the ontology engineering community. People interested in the glossary can find all the current definitions on the Wiki pages in Cicero³⁷. Furthermore, a quick start guide³⁸ explains how to use the tool and how to get access and contribute to it.

Using Cicero for getting feedback has several advantages:

- With Cicero, all discussions are centrally collected and accessible to everyone.
- Cicero applies a specialized methodology for structuring discussions. Cicero facilitates efficient discussions and accelerates convergence to a solution [Dellschaft et al., 2008].
- The discussions captured are (a) part of the design rationale and (b) part of the documentation of the glossary. Because Cicero is an extension of the well-known MediaWiki software, it is possible to have glossary definitions and discussions in the same system. The system then can be used for easily linking glossary definitions and holding relevant discussions with each other and for enhancing the documentation of the glossary.

During the feedback process, we can distinguish between feedback related to already existing definitions (e.g., if the distinction between two terms should be made clearer) and the proposal of new terms that should be included in the glossary.

After setting up the discussion project in Cicero, a call for collaboration in providing feedback of the NeOn Glossary was sent to several mailing lists (including public-owl-dev, protege-owl, web-semanitca-ayuda, semweb-spain, sti, super, and soa4all). Then, a story about this initiative was published on the NeOn website³⁹ and flyers with the call for participation were distributed at the “International Semantic Web Conference 2008 (ISWC 2008)”. For the near future, it is planned to collect the feedback from the discussion project and use it for publishing an updated version of the glossary.

³⁶ http://cicero.uni-koblenz.de/wiki/index.php/Prj:NeOn_Glossary_of_Processes_and_Activities

³⁷ http://cicero.uni-koblenz.de/wiki/index.php/Category:Glossary_Definition

³⁸ http://cicero.uni-koblenz.de/wiki/index.php/Help:Quick_Start

³⁹ http://www.neon-project.org/web-content/index.php?option=com_content&task=view&id=121&Itemid=1

5.3.3. Towards Glossary Standardization

Once the Ontology Engineering community provides us with feedback on the glossary, the next step will be to propose the standardization of the NeOn Glossary. Terminology standards help to avoid confusion by harmonizing terms; thus, we need that the processes and activities involved in the development of ontology networks be standardized.

From our understanding, any standardization agency such as ISO or W3C does not deal with the unification of Ontology Engineering terminology. The only technical committee dealing with ontologies in order to unify linguistic resources is ISO/TC37 (*Terminology and other language resources*). This committee has set up an ontology task force to clarify the terminology of the area; it is worth mentioning that we have been invited to participate in that task force with the goal of standardizing the NeOn Glossary of Processes and Activities.

Thus, the NeOn Glossary of Processes and Activities proposed in this thesis is intended to serve as a useful reference to those in the Ontology Engineering field and to those who come into contact with ontologies.

5.4. Required or If Applicable Processes and Activities

In order to facilitate the identification of which activities are important and necessary for developing a software product, IEEE standards [IEEE, 1997; IEEE, 2006] have identified the activities that are mandatory and those that are optional when developing a particular software product.

Activities of the software development process are categorized as either mandatory or “if applicable” [IEEE, 1997]. Mandatory activities must be always carried out, whereas “if applicable” activities contain an explanation of the cases to which they will apply. In Software Engineering, most activities of the Project Management activity group and some activities of the Support activity group are mandatory. On the other hand, and depending on the concrete development case and on the life cycle model chosen, activities in the Pre-Development, Development, and Post-Development activity groups may or may not be required, that is, they are “if applicable” [IEEE, 2006].

Based on the aforementioned Software Engineering classification and in order to speed up the ontology network development by software developers and ontology practitioners, we identify here which processes and activities are required and which ones are optional (or “if applicable”) for each process and activity included in the NeOn Glossary of Processes and Activities (Section 5.3).

- *Required* processes and activities are those processes and activities that should be carried out when developing networks of ontologies. The processes and activities identified as “required” can be considered as core for the ontology network development. Examples of these are ontology requirements specification, ontology conceptualization, ontology reuse, and ontology implementation.
- *If Applicable* or *Optional* processes and activities are those that can be carried out or not, depending on the case, when developing ontology networks. The other processes and activities (the “if applicable” ones, like ontology customization, ontology localization, ontology aligning, and ontology versioning) are optional in the ontology development, and they should be carried out depending on each specific case. For example, if the ontology is developed in English and has the requirement that the ontology should be available in other natural languages (such as French and Spanish), then the ontology localization activity should be performed.

To carry out the identification of which processes and activities are required and which are optional during the ontology network building process, we made an open call and invited ontology developers from the ontology engineering community participating in European projects (such as NeOn, Knowledge Web, X-Media, etc.) and working in universities and companies (iSOCO, OEG group, DERI, etc.) to take part in an on-line survey⁴⁰. This survey began on July 27th 2007 and the results were collected on August 21st 2007. Thirty five people answered the survey.

The analysed results of this survey, enhanced by our own experience in developing ontologies, are shown in Table 5.

5.5. Processes and Activities Classification based on IEEE Groups

Based on IEEE activity groups [IEEE, 2006] and on the classification proposed by METHONTOLOGY [Gómez-Pérez et al., 2003], we have grouped the 59 processes and activities identified in the NeOn Glossary of Processes and Activities (Section 5.3) into five main groups. The full set of processes and activities covers the entire ontology network life cycle. The groups provide an administrative classification that presents the processes and activities in a more coherent way.

The main groups, as presented in Figure 26, are the following:

1. **Management processes and activities**, which include the following three activities: scheduling, control and ontology quality assurance.
2. **Development-oriented processes and activities**. These are grouped into pre-development, development and post-development activities.

The **pre-development processes and activities**. These include the following activities: ontology environment study, ontology feasibility study, ontology reuse, ontology reengineering, non-ontological resource reuse, and non-ontological resource reengineering.

The **development processes and activities**. They include the following activities: ontology requirements specification, ontology conceptualization, ontology formalization, ontology implementation, ontology integration, ontology modularization, ontology restructuring, ontology merging, ontology aligning, ontology update, ontology modification, ontology localization, ontology translation, ontology annotation, and ontology customization.

The **post-development processes and activities**, which include the following activities: ontology upgrade, ontology versioning, and ontology evolution.

3. **Support processes and activities**. They include the following activities: knowledge acquisition, ontology evaluation, ontology documentation, ontology summarization, ontology assessment, and ontology configuration management.

⁴⁰ <http://droz.dia.fi.upm.es/survey/index.jsp>

	<i>Required</i>	<i>If Applicable</i>
Ontology Aligning		X
Ontology Annotation	X	
Ontology Assessment	X	
Ontology Comparison	X	
Ontology Conceptualization	X	
Ontology Configuration Management	X	
Control	X	
Ontology Customization		X
Ontology Design Patterns Reuse		X
Ontology Diagnosis	X	
Ontology Documentation	X	
Ontology Elicitation	X	
Ontology Enrichment		X
Ontology Environment Study	X	
Ontology Evaluation	X	
Ontology Evolution	X	
Ontology Extension		X
Ontology Feasibility Study	X	
Ontology Formalization	X	
Ontology Forward Engineering		X
Ontology Implementation	X	
Ontology Integration	X	
Knowledge Acquisition for Ontologies	X	
Ontology Learning		X
Ontology Localization		X
Ontology Mapping		X
Ontology Matching		X
Ontology Merging		X
Ontology Modification		X
Ontology Modularization		X
Ontology Module Extraction		X

	<i>Required</i>	<i>If Applicable</i>
Ontology Module Reuse		X
Ontology Partitioning		X
Ontology Population		X
Ontology Pruning		X
Ontology Quality Assurance	X	
Ontology Reengineering		X
Ontology Repair	X	
Ontology Requirements Specification	X	
Non-Ontological Resource Reengineering		X
Non-Ontological Resource Reverse Engineering		X
Non-Ontological Resource Transformation		X
Ontology Restructuring		X
Non-Ontological Resource Reuse		X
Ontological Resource Reuse		X
Ontology Reuse	X	
Ontology Reverse Engineering		X
Scheduling	X	
Ontology Search	X	
Ontology Selection	X	
Ontology Specialization		X
Ontology Statement Reuse		X
Ontology Summarization		X
Ontology Translation		X
Ontology Update		X
Ontology Upgrade	X	
Ontology Validation	X	
Ontology Verification	X	
Ontology Versioning	X	

Table 5. Required-If Applicable processes and activities

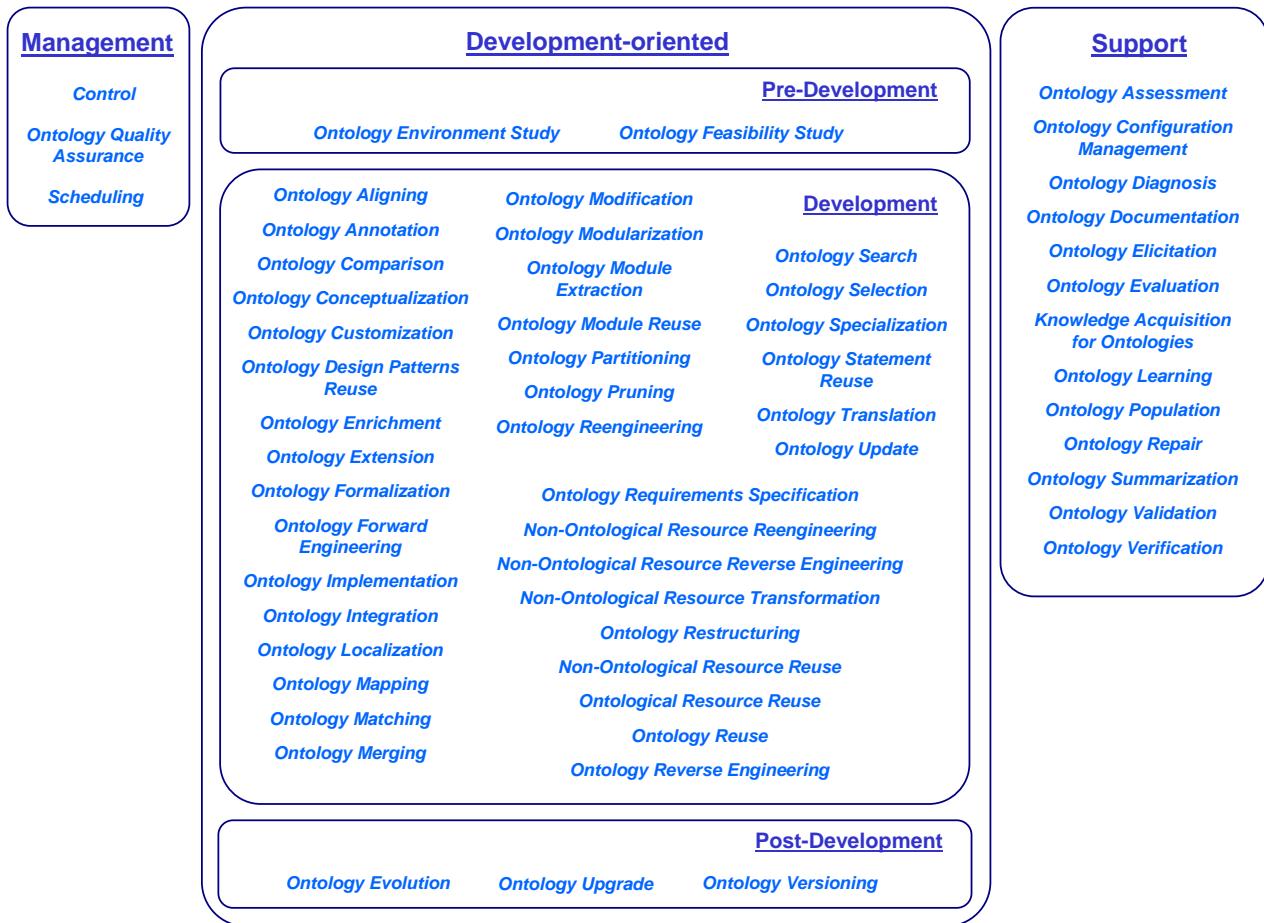


Figure 26. Processes and activities classification

5.6. Relation among Processes and Activities included in the NeOn Glossary and Ontology Networks

In this section, in Table 6 we present which processes and activities from those identified and defined in Section 5.3 are possibly involved when building the different types of ontologies identified in Section 3.1 (single ontologies, interconnected single ontologies, and ontology networks).

Process or Activity Name	Single Ontology	Interconnected Single Ontologies	Ontology Networks
Ontology Aligning			x (hasMapping)
Ontology Annotation	x	x	x
Ontology Assessment	x	x	x
Ontology Comparison	x	x	x
Ontology Conceptualization	x	x	x
Ontology Configuration Management	x	x	x
Control	x	x	x
Ontology Customization	x	x	x
Ontology Design Pattern Reuse		x	x (useImports)
Ontology Diagnosis	x	x	x
Ontology Documentation	x	x	x
Ontology Elicitation	x	x	x
Ontology Enrichment	x	x	x
Ontology Environment Study	x	x	x
Ontology Evaluation	x	x	x
Ontology Evolution			x (hasPriorVersion)
Ontology Extension			x (isExtension)
Ontology Feasibility Study	x	x	x
Ontology Formalization	x	x	x
Ontology Forward Engineering		x	x
Ontology Implementation	x	x	x
Ontology Integration		x	x
Knowledge Acquisition for Ontologies	x	x	x
Ontology Learning	x	x	x
Ontology Localization	x	x	x
Ontology Mapping			x (hasMapping)
Ontology Matching			x (hasMapping)
Ontology Merging			x (hasMapping)
Ontology Modification	x	x	x
Ontology Modularization		x	x (containsModule)
Ontology Module Extraction		x	x (containsModule)
Ontology Module Reuse		x	x (containsModule)
Ontology Partitioning		x	x

Process or Activity Name	Single Ontology	Interconnected Single Ontologies	Ontology Networks
			(containsModule)
Ontology Population	x	x	x
Ontology Pruning	x	x	x
Ontology Quality Assurance	x	x	x
Ontology Reengineering		x	x (useImports)
Ontology Repair	x	x	x
Ontology Requirements Specification	x	x	x
Non-Ontological Resource Reengineering		x	x
Non-Ontological Resource Reverse Engineering		x	x
Non-Ontological Resource Transformation		x	x
Ontology Restructuring	x	x	x
Non-Ontological Resource Reuse		x	x
Ontological Resource Reuse		x	x (useImports)
Ontology Reuse		x	x (useImports)
Ontology Reverse Engineering		x	x (useImports)
Scheduling	x	x	x
Ontology Search		x	x
Ontology Selection		x	x
Ontology Specialization		x	x (isSpecialization)
Ontology Statement Reuse		x	x (useImports)
Ontology Summarization	x	x	x
Ontology Translation	x	x	x
Ontology Update	x	x	x
Ontology Upgrade		x	x (hasPriorVersion)
Ontology Validation	x	x	x
Ontology Verification	x	x	x
Ontology Versioning	x	x	x

Table 6. Relation among processes and activities and ontology networks

5.7. Conclusions

As already mentioned, ontology experts have reached a certain implicit consensus on what processes and activities should be undertaken in ontology development. This implicit consensus can be made explicit by means of establishing a glossary of processes and activities in the Ontology Engineering field.

As we have shown along this chapter, the NeOn Glossary of Processes and Activities, which identifies and defines the processes and activities potentially involved in the ontology network construction, has been established by a consensus reaching process among ontology experts. This glossary is a first step for solving the lack of a standard glossary in the Ontology Engineering in contrast with the Software Engineering field that boasts the IEEE Standard Glossary of Software Engineering Terminology [IEEE, 1990].

Thus, this chapter contributes to the whole thesis by providing the identification and definition of the processes and activities potentially involved in the ontology network development. The NeOn Glossary of Processes and Activities presented in this chapter is published in the NeOn website⁴¹. For the time being, this glossary is addressed to the Ontology Engineering community. Comments, examples, and additional information are being provided to make the glossary more complete and user-friendly for software developers and users.

We are now in the process of obtaining feedback from the Ontology Engineering community on the current glossary. For this purpose, we use the argumentation tool Cicero, built on top of the wiki technology, so that the Ontology Engineering community can comment the definitions. Our long term goal is to have a more complete and commonly-agreed glossary, which could become the terminological reference in the Ontology Engineering field. The final goal is to propose the standardization of the NeOn Glossary of Processes and Activities in the ontology task force set up by the technical committee ISO/TC37.

Additionally, this chapter provides two classifications: (1) one related to required or optional processes and activities; and (2) other related to general groups based on IEEE [IEEE, 2006] for process and activities. In addition, it provides a table that shows the relation among processes and activities and ontology networks.

⁴¹ <http://www.neon-project.org/web-content/images/Publications/neonglossaryofactivities.pdf>

6. Scenarios for Building Ontology Networks

6.1. Introduction

As stated in Chapter 3, one of the goals of this PhD is to propose the NeOn Methodology for building collaboratively ontology networks. This methodology will include methods, techniques and tools for carrying out the activities identified and defined in the ontology network development process and will be focused on ontology requirements specification, scheduling and ontological resource reuse.

In this regard, we must solve first the following open research problem (identified in Chapter 2):

- None of the methodologies available cover complex scenarios in which reuse and reengineering of ontological and non-ontological resources are needed.

The current methodologies propose an ontology development from scratch through a simple scenario that ranges from specification to implementation though this scenario is very rigid. Our experience tells that there are alternative scenarios to develop ontologies. Thus, we should probe that

- The ontology network development entails combining different scenarios in which issues such as reuse of ontological and non-ontological resources, reengineering, merging, restructuration, and localization are involved.

In this regard, this chapter presents the identification of nine scenarios for building ontology networks, the relation between processes and activities and scenarios, and an analysis of how argumentation and collaboration issues are related to the scenarios identified.

6.2. Nine Scenarios for Building Ontology Networks

In the framework of the NeOn Methodology, we have identified a set of nine flexible scenarios for collaboratively building ontologies and ontology networks, placing special emphasis on reusing and reengineering knowledge resources (ontological and non-ontological).

Figure 27 presents the set of the 9 most plausible scenarios for building ontologies and ontology networks. Directed arrows with associated numbered circles represent the different scenarios. Each scenario is decomposed into different processes or activities. Processes and activities are represented with coloured circles or with rounded boxes and are defined in the NeOn Glossary of Processes and Activities presented in Section 5.3. Figure 27 also shows (as dotted boxes) the existing knowledge resources to be reused, and the possible outputs that result from the execution of some of the presented scenarios.

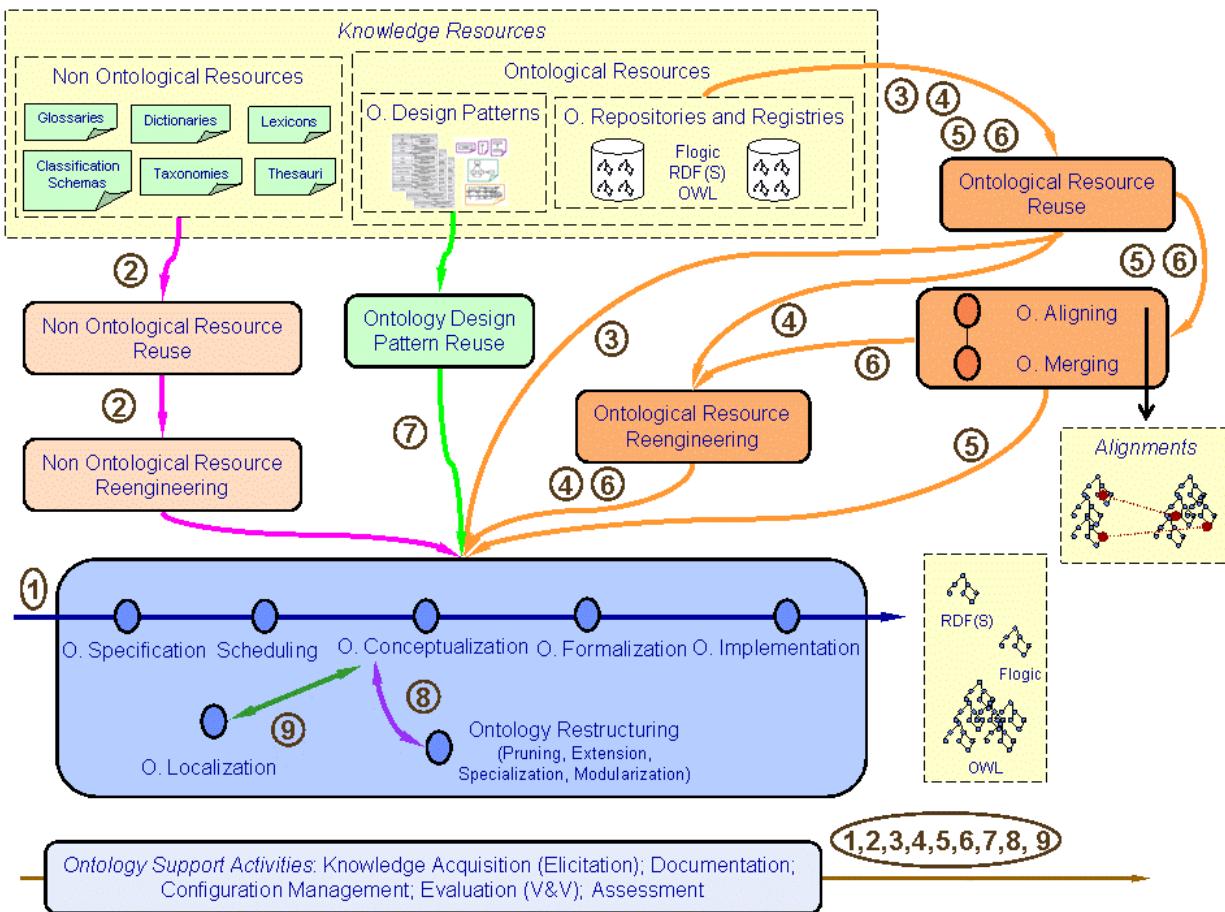


Figure 27. Scenarios for building ontologies and ontology networks

This chapter includes, as independent sections, the most common scenarios that may unfold during the ontology network development, though they can not be considered exhaustive:

- **Scenario 1: From specification to implementation.** The ontology network is developed from scratch, that is, without reusing knowledge resources available.
- **Scenario 2: Reusing and reengineering non-ontological resources.** Ontology developers should carry out the non-ontological resource reuse process for deciding, according to the requirements in the ORSD, which NORs can be reused to build the ontology network. Then, the selected NORs should be re-engineered into ontologies.
- **Scenario 3: Reusing ontological resources.** Ontology developers use ontological resources (ontologies as a whole, ontology modules, and/or ontology statements).
- **Scenario 4: Reusing and reengineering ontological resources.** Ontology developers reuse and re-engineer ontological resources.
- **Scenario 5: Reusing and merging ontological resources.** This scenario unfolds only in those cases where several ontological resources in the same domain are selected for reuse and when ontology developers wish to create a new ontological resource from two or more ontological resources.
- **Scenario 6: Reusing, merging and reengineering ontological resources.** Ontology developers reuse, merge, and reengineer ontological resources in the ontology network building. This scenario is similar to Scenario 5; however, here developers decide not to use the set of merged resources such as it is, but to reengineer it.

- Scenario 7: Reusing Ontology Design Patterns (ODPs).* Ontology developers access ODPs repositories to reuse them.
- Scenario 8: Restructuring ontological resources.* Ontology developers restructure (modularizing, pruning, extending, and/or specializing) ontological resources to be integrated in the ontology network being built.
- Scenario 9: Localizing ontological resources.* Ontology developers adapt an ontology to other languages and culture communities, obtaining a multilingual ontology.

Knowledge acquisition, documentation, configuration management, evaluation and assessment should be carried out all along the ontology development (during the whole ontology network development), that is, in any scenario used for developing the ontology network. The intensity of such support activities depends on the concrete moment of the development progress.

It is worth mentioning that these scenarios can be combined in different and flexible ways, and that any combination of scenarios should include Scenario 1 because this scenario is made up of the core activities that have to be performed in any ontology development. Indeed, as Figure 27 shows, the results of any other scenario should be integrated in the corresponding activity of Scenario 1.

Processes and activities included in this thesis cover partially Scenario 1 and Scenario 3.

The following sections present the different scenarios identified; each section is organized as follows:

- Motivation for the scenario.
- Assumptions for the scenario. For each of the scenarios presented, it is supposed that the scenario begins with the decision of whether a single ontology, an interconnected set of ontologies, or an ontology network is going to be developed. In this chapter, it is assumed that an ontology network will be built for all scenarios.
- Prerequisite resources.
- Sequence of processes, activities, and tasks to be carried out. The processes and activities included are taken from the NeOn Glossary of Processes and Activities (Section 5.3).
- Outcomes for the scenario.

6.2.1. Scenario 1: From Specification to Implementation

- Motivation:** This scenario unfolds in those cases in which ontology developers⁴² wish to build ontologies from scratch. The scenario is based on those methodological approaches that appeared in the 1990s and the first years of this new century, with the objective of transforming the ontology building in an engineering activity.
- Assumptions:** In this scenario it is supposed that ontology developers build the ontology network from scratch, that is, without reusing knowledge resources available.
- Prerequisite resources:** Knowledge about the domain (general or specific) of the ontology network to be developed should be available for the building of the ontology network.

⁴² In this thesis, ontology developers refer to software developers and ontology practitioners involved in the development of ontologies.

- **Sequences of activities**

When ontology developers think about the use of ontologies for solving a particular problem, a provisional⁴³ work team of ontology developers (if possible involving ontology engineers, software developers, domain experts and final users) should be established. Such a team will be concerned at least with the following activities: environment and feasibility study, knowledge acquisition, ontology requirements specification, and scheduling.

After the provisional team has been established, the team should carry out an *environment and feasibility study*. This allow them to decide whether ontologies should be developed or not for the specific problem. 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, whose differences are explained in Section 3.1.

Once such a decision has been taken, the provisional team should start with the *knowledge acquisition activity*, possibly including the *ontology elicitation activity*. This activity should be carried out during the whole development; however, ontology developers should acquire most of the knowledge at the beginning of the ontology development.

Simultaneously with the knowledge acquisition activities, ontology developers should specify the requirements that the ontology should fulfil, by means of the *ontology requirements specification activity*. The objective of this activity is to output the ontology requirements specification document (ORSD) that includes the purpose, the scope and the implementation language of the ontology network, the target group and the intended uses of the ontology network, as well as the set of requirements that the ontology network should fulfil, mainly in the form of competency questions (CQs). After the ontology requirements specification activity, it is advisory to carry out a quick search for knowledge resources, using as input terms included in the ORSD. Then, the *scheduling activity* must be carried out, using the ORSD and the results of such a quick search. During the scheduling activity, the team establishes the ontology network life cycle and the human resources needed for the ontology project. Such resources may include or not people from the provisional team, depending on organizational issues.

Then, the ontology developers assigned to the ontology project should carry out (1) the *ontology conceptualization activity*, in which knowledge is organized and structured into meaningful models at the knowledge level; (2) the *ontology formalization activity*, in which the conceptual model is transformed into a semi-computable model; and (3) the *ontology implementation activity*, in which a computable model (implemented in an ontology language) is generated.

- **Outcomes:** The principal output is a network of ontologies that represents the expected domain implemented in an ontology language (OWL⁴⁴, F-Logic [Kifer and Lausen, 1989], etc.). In addition, a broad range of documents, such as the ontology requirements specification document, the ontology description document and the ontology evaluation document, will be generated as output by the different activities.

⁴³ This first team can be considered as provisional because the definitive team is established during the scheduling activity.

⁴⁴ <http://www.w3.org/TR/owl-ref/>

6.2.2. Scenario 2: Reusing and Reengineering Non-Ontological Resources

- ❑ **Motivation:** Currently, ontology developers are realizing the benefits of “not reinventing the wheel” at each ontology development. They are starting to reuse as much as possible non-ontological resources, such as classification schemes, thesauri, lexicons and folksonomies, built by others that already have reached some degree of consensus, with the aim of speeding up the ontology development process. The reuse of such resources involves necessarily their reengineering into ontologies. Therefore, this scenario unfolds in those cases in which ontology developers wish to reuse the non-ontological resources at their disposal.
- ❑ **Assumptions:** In this scenario it is supposed that ontology developers develop the ontology network by means of reusing non-ontological resources (i.e., glossaries, dictionaries, lexicons, classification schemes and taxonomies, and thesauri).
- ❑ **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.

Additionally, non-ontological resources, related to the domain that the ontology network must cover, should be made available.

- ❑ **Sequence of processes and activities**

As Figure 27 shows (by arrows with the number 2), ontology developers should accomplish first the *non-ontological resource reuse process* and then choose the most suitable non-ontological resources (thesauri, glossaries, data bases, etc.) to be used for building the ontology network. Such non-ontological resources cover to some extend the domain of the ontology network being built. If ontology developers decide that one or more resources are useful for the ontology network development, then the *non-ontological resource reengineering process* should be carried out to transform the non-ontological resources selected into ontologies. After this process, ontology developers should use the resultant ontologies as input of some of the activities included in Scenario 1 (explained in Section 6.2.1), as shown in Figure 27.

The activities for carrying out the non-ontological resource reuse process are briefly explained below; details are available in [Suárez-Figueroa et al., 2008]:

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. The input for this activity is the ontology requirements specification document (ORSD).

Activity 2. Assess the set of candidate non-ontological resources. The goal of this activity is to assess the set of candidate non-ontological resources obtained in Activity 1. To carry out this activity, the following criteria should be used: coverage, precision, and consensus in the resource, which is a subjective criterion.

Activity 3. Select the most appropriate non-ontological resources. The goal of this activity is to select the most appropriate non-ontological resources from those candidates obtained in Activity 2.

As mentioned before, the goal of the non-ontological resource reengineering process is to transform a non-ontological resource into an ontology. This process can be divided into the following activities [García-Silva et al., 2008]:

Activity 1. Non-ontological resource reverse engineering. The goal of this activity is to analyse a non-ontological resource in order to identify its underlying components and create representations of the resource at the different levels of abstraction (design, requirements and conceptual).

Activity 2. Non-ontological resource transformation. The goal of this activity is to generate a conceptual model from the non-ontological resource. To perform such a transformation, the following characteristics have to be identified: (1) the non-ontological resource type; (2) the internal data model of the non-ontological resource; and (3) the semantics of the relations among the non-ontological resource entities. To guide this transformation process, in [García-Silva et al., 2008] the use of the so-called patterns for reengineering non-ontological resources (PR-NOR) is proposed.

Activity 3. Ontology forward engineering. The goal of this activity is to output a new implementation of the ontology on the basis of the new conceptual model identified in activity 2. To depict this activity, we use the ontology levels of abstraction because they are directly related to the ontology development process.

- **Outcomes:** The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Furthermore, a broad range of documents containing the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities.

Additionally, the non-ontological resources selected to be reused have been “ontologized” by means of the non-ontological resource reengineering activity.

It is important to mention that this scenario is treated in detail in a thesis now in progress [Villazón-Terrazas, in progress] about reuse and reengineering non-ontological resources.

6.2.3. Scenario 3: Reusing Ontological Resources

- **Motivation:** Lately, ontological resources have been collected in ontology library systems [Ding and Fensel, 2001], which offer various functions for managing, adapting and standardizing groups of ontologies, and in ontology repositories⁴⁵ that support knowledge access and reuse by humans and machines. As more ontological resources are available in ontology library systems, in ontology repositories and on the Internet, ontology developers are starting to reuse them not only with the idea of “not reinventing the wheel”, but also with the aim of taking advantage of them. Thus, this scenario unfolds in those cases in which ontology developers have at their disposal ontological resources useful for their problem and that can be reused in the ontology development.
- **Assumptions:** In this scenario it is supposed that ontology developers use ontological resources in the same or similar domain than that of the ontology network to be developed for building the ontology network. The ontological resources may have been developed by ontology developers or by others.
- **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.

Additionally, ontological resources related to the domain that the ontology network must cover should be made available.

⁴⁵ An ontology repository is a structured collection of ontologies, modules and additional meta-knowledge. The software that manages an ontology repository is known as ontology repository management system; this system stores, organizes, modifies and extracts knowledge from an ontology repository. Related to this, the ontology registries should be also mentioned since such registries store descriptions of ontologies and not ontologies themselves.

□ **Sequence of processes and activities**

As Figure 27 shows (by arrows with the number 3), ontology developers should perform the *ontological resource reuse process*, which is composed of the following activities:

Activity 1. Ontology search. Ontology developers search for the candidate ontological resources that satisfy the requirements in repositories and registries like Knowledge Zone⁴⁶, ONTHOLOGY.org⁴⁷, Oyster⁴⁸, Swoogle⁴⁹, and WATSON⁵⁰. These ontological resources could be implemented in different languages or could be available in different ontology tools.

Activity 2. Ontology assessment. Ontology developers must inspect the content and granularity of the ontological resources obtained in Activity 1 in order to find out if such resources satisfy the developers needs.

Activity 3. Ontology comparison. Ontology developers should compare the ontological resources assessed in Activity 2, taking into account a set of criteria identified by developers (e.g., existing ontology documentation) and the ontology network requirements identified in the ORSD.

Activity 4. Ontology selection. Ontology developers should select the set of ontological resources that are the most appropriate for their ontology network requirements, based on the comparisons obtained in Activity 3.

After selecting the most appropriate ontological resources, ontology developers should define the reuse mode; that is, ontology developers need to decide how they will reuse the selected ontological resources. There are three possible modes:

- The ontological resources selected will be reused as they are.
- The ontology reengineering activity should be carried out with the ontological resources selected.
- Some ontological resources will be merged to obtain a new ontological resource.

Before reusing the selected ontological resources by means of any reuse mode, it is also convenient to evaluate these resources through the *ontology evaluation activity*.

Activity 5. Ontology integration. Ontology developers should include, as they are, the ontological resources selected (the code) in Activity 4 into the ontology network being built following the activities of Scenario 1 (Section 6.2.1).

□ **Outcomes:** The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities.

⁴⁶ <http://smiprotege.stanford.edu:8080/KnowledgeZone/>

⁴⁷ <http://www.ontology.org/>

⁴⁸ <http://oyster.ontoware.org>

⁴⁹ <http://swoogle.umbc.edu/>

⁵⁰ <http://watson.kmi.open.ac.uk/WatsonWUI/>

6.2.4. Scenario 4: Building, Reusing and Reengineering Ontological Resources

- **Motivation:** This scenario unfolds in those cases in which ontology developers have at their disposal ontological resources useful for their problem and that can be reused in the ontology network development. However, such resources are not exactly useful as they are, so they should be modified (that is, reengineering) to serve to the intended purpose or problem.
- **Assumptions:** It is supposed that ontology developers use ontological resources by reengineering them before reusing them in the ontology network building.
- **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.
Additionally, ontological resources related to the domain to be covered by the ontology network to be developed should be available.
- **Sequence of processes and activities**

As Figure 27 shows (by arrows with the number 4), ontology developers should perform first the *ontological resource reuse process* to select the most suitable ontological resources to be used for building the ontology network. Then, they should carry out the *ontological resource reengineering process* to modify the selected ontological resources. Finally, they should use the resultant ontological resources as input of some of the activities included in Scenario 1 (explained in Section 6.2.1) as shown in Figure 27.

Specifically, ontology developers should carry out some activities as part of the ontological resource reuse process; such activities are the following: *ontology search*, *ontology assessment*, *ontology comparison*, and *ontology selection* as already explained in Scenario 3 (Section 6.2.3).

After the ontology selection activity, ontology developers should decide how they will reuse the ontological resources. They should also decide whether to perform the ontological resource reengineering process with the selected ontological resources, because these resources are not absolutely useful for the concrete use case as they are and they need to be transformed in some way.

The ontological resource reengineering process proposed here has been created taking as inspiration the software reengineering process [Byrne, 1992]. It is composed of the following activities: *ontological resource reverse engineering*, *ontological resource restructuring*, and *ontological resource forward engineering*.

Additionally, this process is related to the levels of abstraction shown in Figure 28, which are based on [Byrne, 1992] and are described below.

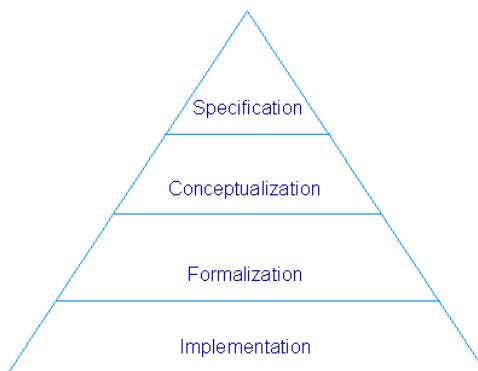


Figure 28. Levels of abstraction for the ontological resource reengineering process

- Specification is the highest level of abstraction. In this level, requirements, purpose and scope, among other components of the specification, are described.
- In the conceptualization level, ontology characteristics such as structure and components are described. The knowledge that the ontology represents is organized following a set of knowledge representation primitives (concepts, relations, etc.).
- In the formalization level, the formal or semi-computable model that was used to transform the conceptual model is described.
- The implementation level is the lowest abstraction level. Here, the ontology description focuses on implementation characteristics and is represented in an ontology language understandable by computers and usable by automatic reasoners.

Figure 29 presents the ontological resource reengineering model.

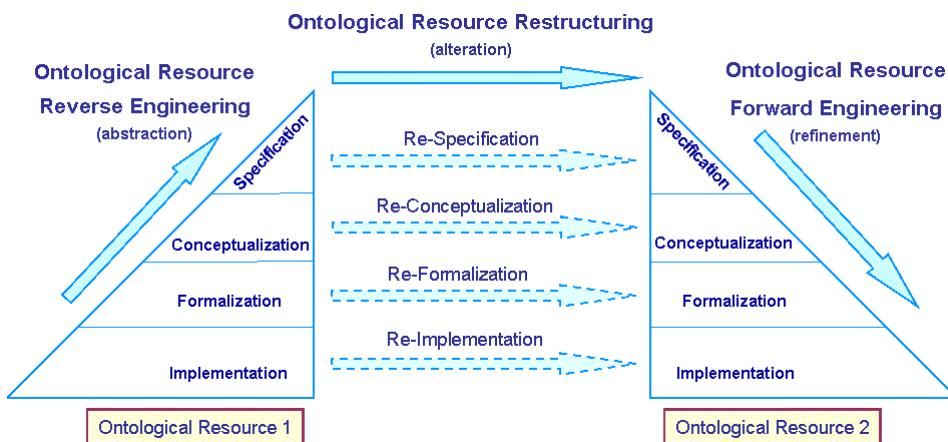


Figure 29. Ontological Resource Reengineering Model

This model suggests different paths to reengineer an ontological resource, taking into account the levels of abstraction presented in Figure 28 . Examples of these paths are

- At implementation level⁵¹: from ontological resource 1 code to ontological resource 2 code.
- At formalization level: reverse engineering (from code 1 to formalization 1), restructuring formalization 1 to obtain formalization 2, and forward engineering to obtain code of resource 2.
- At conceptualization level: reverse engineering (from code 1 to conceptualization 1), restructuring conceptualization 1 to obtain conceptualization 2, and forward engineering to obtain formalization or implementation 2.
- At conceptualization level: reverse engineering (from code 1 to conceptualization 1), restructuring conceptualization 1 to obtain conceptualization 2, and forward engineering to obtain code of the ontological resource 2.

⁵¹ This level could be considered no part of the proper reengineering process.

The choice of a concrete path depends on the ontological resource characteristics that have to be changed. Thus, in Figure 29, we can distinguish the following types of changes:

- Re-specification. If the ontology developer restructures the requirements specification, (s)he changes requirements, purpose and scope, among other elements of the requirements specification. For example, changes in requirements; addition or deletion of requirements, etc.
- Re-conceptualization. If (s)he restructures the conceptualization, changes might refer to modification of ontology structure, modification of granularity and richness of the knowledge, removal or addition of axioms, restructuration of ontology architecture (modularization), inclusion of new concepts, use of ontology design patterns, etc.
- Re-formalization. If (s)he restructures the formalization level, the changes refer to formalization characteristics (such as changing the ontology paradigm from Description Logic to Frames).
- Re-implementation. If (s)he restructures the implementation level, the changes are focused on implementation characteristics that are tightly related to the ontology implementation language (e.g., translation from RDF(S) to OWL). Other changes could be conforming to coding standards, improving code readability, renaming code items, etc.

Ontology developers should decide at which level they need to carry out the ontological resource reengineering process. Once ontology developers have decided the level, they should carry out the ontological resource reengineering process, and then they should integrate the result of such a process (code, formalization, conceptualization, or specification) into the corresponding activity of Scenario 1 (Section 6.2.1).

- **Outcomes:** The principal outcome is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including requirements specification, ontology documentation, ontology evaluation, etc. will be generated as output of different activities.

Furthermore, new ontological resources from those selected for their reuse are generated through the ontological resource reengineering process. Such new resources can be considered as new versions of the ontological resources after the reengineering process.

6.2.5. Scenario 5: Reusing and Merging Ontological Resources

- **Motivation:** This scenario unfolds in those cases where several ontological resources in the same domain can be selected for reuse and when the ontology developer wishes to create a new ontological resource from two or more, possibly overlapping, ontological resources. It could also occur that the ontology developer wishes only to establish alignments among the ontological resources selected in order to create the ontology network.
- **Assumptions:** It is supposed that ontology developers use ontological resources and merge them before reusing them in the development of the ontology network.
- **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.

Furthermore, ontological resources related to the domain to be covered by the ontology network to be developed should be available.

□ **Sequence of processes and activities**

As Figure 27 shows (by arrows with the number 5), ontology developers should perform first the *ontological resource reuse* process to select the most suitable ontological resources that will be used for building the ontology network. Concretely, ontology developers should carry out the activities presented in Scenario 3 (Section 6.2.3) as part of the ontological resource reuse process. After the ontology selection activity, ontology developers should decide how they will reuse the ontological resources selected. In this scenario, ontology developers decide to perform the following activities because the selected resources are valid as they are but not in a complete way for the concrete case if they are reuse separately. The activities to be performed are the following:

Activity 1. Ontology aligning. Ontology developers carry out this activity with the aim of obtaining a set of alignments among the selected ontological resources.

Activity 2. Ontology merging. Ontology developers can merge the selected ontological resources using the alignments (output of activity 1) to obtain a new ontological resource from the overlapping selected ones.

Ontology developers have here two different possibilities: (1) to establish the mappings among such selected resources; (2) to establish the mappings and to merge the selected resources.

After that, ontology developers should use the resultant merged ontological resource as input of some of the activities included in Scenario 1 (explained in Section 6.2.1) as shown in Figure 27.

- **Outcomes:** The principal outputs are (a) a set of alignments among the selected ontological resources, and (b) a set of new ontological resources to be integrated as they are in the ontology network.

6.2.6. Scenario 6: Reusing, Merging and Reengineering Ontological Resources

- **Motivation:** This scenario unfolds in those cases in which several ontological resources in the same domain can be selected to build the ontology network. Ontology developers decide to create a new ontological resource merging two or more, possibly overlapping, ontological resources. Such a merged ontological resource is not useful as it is, so it should be modified (that is, reengineering) to serve to the intended purpose to be covered by the ontology network being built.
- **Assumptions:** It is supposed that ontology developers use ontological resources by means of merging and reengineering them before they are reused in the ontology network building.
- **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.

Furthermore, ontological resources related to the domain to be covered by the ontology network to be developed should be available.

- **Sequence of processes and activities**

As Figure 27 shows (with arrows with the number 6), ontology developers should perform first the *ontological resource reuse process* to select the most suitable ontological resources for building the ontology network (as explained in Scenario 3 (Section 6.2.3)). Then, they should decide how they will reuse the selected ontological resources. It is in this scenario where ontology developers decide to perform the *ontology aligning and ontology merging* activities because the selected resources are valid but not in a complete way for the concrete case if they are considered separately, as explained in Scenario 5 (Section 6.2.5). After merging the selected resources, they should carry out the *ontological resource reengineering process* as described in Scenario 4 (Section 6.2.4). After that, they should use the resultant ontological resource as input of some of the activities included in Scenario 1 (explained in Section 6.2.1) as shown in Figure 27.

- **Outcomes:** The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities.

Furthermore, a merged ontological resource, taken from those selected for reuse, and a reengineered merged ontological resource are generated. Alignments between the ontological resources selected are also outputs of this scenario.

6.2.7. Scenario 7: Reusing Ontology Design Patterns

- **Motivation:** Recently, within the Ontology Engineering field, ontology design patterns (ODPs) have emerged as (1) a way of helping ontology developers to model OWL ontologies [Gangemi, 2005; Pan et al., 2007], and (2) a new mode of encoding best practices, based on experiences and knowledge of ‘good’ solutions. As any other type of patterns, ODPs are perceived as having three kinds of benefits [Blomqvist et al., 2009]: (1) reuse benefits, (2) guidance benefits, and (3) communication benefits. ODPs can be found in on-line libraries that include both the description and the OWL code associated to the patterns as, for example, “the Ontology Design Pattern Wiki”⁵², or they can be obtained from the work team “Semantic Web Best Practices and Deployment”⁵³. Thus, this scenario unfolds in those cases where best practices can be applied in the development of ontology networks.

- **Assumptions:** It is supposed that ontology developers reuse ontology design patterns (ODPs) for different purposes: to reduce modelling difficulties, to speed up the modelling process, to check the adequacy of modelling decisions, or to evaluate modelling solutions.

- **Prerequisite resources:** Knowledge about the domain of the ontology network to be developed should be available for the building of the ontology network. Furthermore, previous knowledge on the existence of design patterns and design patterns repositories should also be available. Ontology design patterns repositories or catalogues should be made available to users. These catalogues should contain templates that fully describe design patterns.

- **Sequence of processes and activities**

Ontology developers work on the development of an ontology network and very often encounter problems regarding the way in which certain knowledge should be modelled. This may happen during the ontology conceptualization activity, the ontology formalization activity, or during the ontology implementation activity. Since ontology developers are conscious of the existence of ontology design pattern repositories, they access them in order to find modelling solutions.

⁵² <http://ontologydesignpatterns.org/>

⁵³ <http://www.w3.org/2001/sw/BestPractices/>

As Figure 27 shows (with arrows with the number 7), ontology developers should perform the *ontology design pattern reuse process* to select the most suitable ODPs for building the ontology network. The activities to be carried out during the process are the following ones:

Activity 1. ODP search. Ontology developers search for possible ODPs to be used in a repository or catalogue. This implies to carry out a careful analysis of the different information sections included in the ODPs templates in order to find possible ODPs for solving the modelling problem.

Activity 2. ODP selection. Ontology developers select the ODPs that best match the modelling problem. This implies to contrast the ontology design pattern templates analysed against the real use case for finding out the overlap level.

Activity 3. ODP adaptation. Ontology developers adapt the ODP in order to match the modelling issue. Depending on the selected pattern, an instantiation or an extension of the ontology design pattern has to be performed.

Activity 4. ODP integration. Ontology developers integrate the ODPs into the corresponding model (conceptualization, formalization or implementation).

- **Outcomes:** The principal output of this reuse process is an ontology design pattern integrated into the ontology network being developed.

6.2.8. Scenario 8: Restructuring Ontological Resources

- **Motivation:** This scenario unfolds in those cases where the knowledge contained in the conceptual model of the ontology network should be corrected and reorganized to obtain the network that covers the ontology requirements.
- **Assumptions:** It is supposed that ontology developers restructure ontological resources.
- **Prerequisite resources:** Knowledge of the domain (general or specific) of the ontology network to be developed should be available for building the ontology network.
- **Sequence of activities**

Ontology developers should perform the *ontology restructuring activity* to modify the ontology network being built, after the ontology conceptualization activity. The *ontology restructuring activity* can be performed by executing any of the following subactivities, combining them in any manner and order:

- **Ontology modularization activity.** Ontology developers create different ontology modules in the ontology network, which facilitates the reuse of the knowledge included in the network.
- **Ontology pruning activity.** Ontology developers prune those branches of the taxonomies included in the ontology network that are considered not necessary to cover the ontology requirements.
- **Ontology enrichment activity.** This activity can be carried out by performing any of the two subactivities that follow:
 - **Ontology extension activity.** Ontology developers extend the ontology network, including (in width) new concepts and relations.
 - **Ontology specialization activity.** Ontology developers specialize those branches of the ontology network that require more granularity and include more specialized concepts and relations.

Note that this activity (ontology restructuring) can be performed (1) in an independent way as explained in this scenario, or (2) as part of the ontological resource reengineering process as described in Scenario 4 in Section 6.2.4.

- **Outcomes:** The principal output is a conceptual model of the ontology network that represents the expected domain.

6.2.9. Scenario 9: Localizing Ontological Resources

- **Motivation:** Although access to top-quality ontologies (e.g., Galen, CYC, or AKT) is, in many cases, free and unlimited for users all around the world, most of these ontologies are available only in English. Due to the language barrier, non-English users therefore often encounter problems when trying to access ontological knowledge in their own languages. Moreover, more and more ontology-based systems are being built for multilingual applications (e.g., multilingual machine translation, or multilingual information retrieval). For these reasons, the need for multilingual ontologies has increased. Thus, this scenario unfolds in those cases in which the ontology network to be developed should be written in different natural languages.
- **Assumptions:** It is supposed that ontology developers include multilingualism in an ontological resource.
- **Prerequisite resources:** Knowledge about the domain (general or specific) of the ontology network to be developed should be available for building the ontology network. Knowledge resources in different idioms (e.g., thesauri and dictionaries) should also be available.
- **Sequence of activities and tasks**

Ontology developers should perform the *ontology localization activity* once the ontology has been conceptualized and restructured. This activity requires the translation of all the ontology terms into another natural language (Spanish, French, German, etc.) different from the language used in the conceptualization, using multilingual thesauri and electronic dictionaries (e.g., EuroWordNet⁵⁴). This ontology localization activity is composed of the following tasks [Espinoza et al., 2009]:

Task 1. Selecting the most appropriate linguistic assets. The goal of this task is to select the most appropriate linguistic assets that help to reduce the cost, to improve the quality of the localization, and to increase the consistency of the localization activity.

Task 2. Selecting ontology label(s) to be localized. The goal of this task is to select the ontology label(s) to be localized.

Task 3. Obtaining ontology label translation(s). The goal of this task is to obtain the most appropriate translation in the target language for each ontology label.

Task 4. Evaluating label translation(s). The goal of this task is to evaluate the label translations in the target language.

Task 5. Updating the ontology. The goal of this task is to update the ontology with the label translations obtained for each localized label. The task output is an ontology enriched with labels in the target language associated to each localized term.

After this localization activity, the resulting conceptual model should be integrated in the conceptualization activity of Scenario 1 (Section 6.2.1).

- **Outcomes:** The principal outcome is a conceptual model of the ontology network in different natural languages (that is, a multilingual conceptual model) that represents the expected domain.

It is important to mention that this scenario is treated in detail in an ongoing thesis [Espinoza, in progress] about localizing ontologies.

⁵⁴ <http://www illc uva nl/EuroWordNet/>

6.3. Relation between Processes and Activities and Scenarios

In this section we present Table 7 that relates scenarios with processes and activities in the following way: (1) a process or an activity can be carried out in a particular scenario; or (2) a process or an activity can be carried out in a list of scenarios.

6.4. Collaboration and Argumentation in NeOn Scenarios

In general, collaboration can involve two main situations [Dellschafft et al., 2008b]: (i) situations of reuse-oriented collaboration (e. g. reusing ontologies or ontology design patterns, making an ontology functional to a given task through appropriate evaluation and selection, reengineering thesauri, folksonomies, and/or classification schemes) or (ii) situations of active interaction between agents for producing a knowledge resource (e. g. collaborative editing and annotation of ontology elements and changes, discussing and getting consensus on design solutions and their rationales in context).

A subset of the nine scenarios presented in Section 6.2 is related to the first of the situations aforementioned.

With respect to the second situation, the collaborative development of ontology networks involves teams composed by different agents with different roles, conscious of their participation in the same plan and who share a same goal. The different participants that have different roles in an ontology development project should carry out different activities in which interaction and collaboration is always needed. In this context, the participants should discuss their different viewpoints in an efficient manner. Thus, argumentative discussions are an important part of collaborative ontology engineering. 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.
- 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. 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 requirements specification, ontology conceptualization, ontology formalization and ontology implementation phases. The 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 the building of an ontology from scratch. It may also be needed during the reusing or reengineering of ontological and non-ontological resources, or during the alignment with other ontologies.

Process or Activity Name	Related Scenarios
Ontology Aligning	S5, S6
Ontology Annotation	S1...S9
Ontology Assessment	S1...S9
Ontology Comparison	S3 ... S6
Ontology Conceptualization	S1
Ontology Configuration Management	S1...S9
Control	S1...S9
Ontology Customization	S8
Ontology Design Pattern Reuse	S7
Ontology Diagnosis	S1...S9
Ontology Documentation	S1...S9
Ontology Elicitation	S1...S9
Ontology Enrichment	S4, S6, S8
Ontology Environment Study	S1
Ontology Evaluation	S1...S9
Ontology Evolution	S1
Ontology Extension	S4, S6, S8
Ontology Feasibility Study	S1
Ontology Formalization	S1
Ontology Forward Engineering	S2, S4, S6
Ontology Implementation	S1
Ontology Integration	S2...S6
Knowledge Acquisition for Ontologies	S1...S9
Ontology Learning	S1...S9
Ontology Localization	S9
Ontology Mapping	S5, S6
Ontology Matching	S5, S6
Ontology Merging	S5, S6
Ontology Modification	S1
Ontology Modularization	S4, S6, S8
Ontology Module Extraction	S4, S6, S8

Process or Activity Name	Related Scenarios
Ontology Module Reuse	S3..S6
Ontology Partitioning	S4, S6, S8
Ontology Population	S1...S9
Ontology Pruning	S4, S6, S8
Ontology Quality Assurance	S1...S9
Ontology Reengineering	S4, S6
Ontology Repair	S1...S9
Ontology Requirements Specification	S1
Non-Ontological Resource Reengineering	S2
Non-Ontological Resource Reverse Engineering	S2
Non-Ontological Resource Transformation	S2
Ontology Restructuring	S4, S6, S8
Non-Ontological Resource Reuse	S2
Ontological Resource Reuse	S3...S6
Ontology Reuse	S3...S6
Ontology Reverse Engineering	S4, S6
Scheduling	S1
Ontology Search	S3...S6
Ontology Selection	S3...S6
Ontology Specialization	S4, S6, S8
Ontology Statement Reuse	S3...S6
Ontology Summarization	S1...S9
Ontology Translation	S1
Ontology Update	S1
Ontology Upgrade	S1
Ontology Validation	S1...S9
Ontology Verification	S1...S9
Ontology Versioning	S1

Table 7. Correspondence between scenarios and processes and activities

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 to keep track of the design rationale all the way through the ontology engineering process, and to keep 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 leads 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.

6.5. Conclusions

As we have tried to show along this chapter, it is possible to develop ontology networks combining different scenarios that involve the reuse of ontological and non-ontological resources, reengineering, merging, restructuration, and localization.

In this chapter we have identified a set of 9 flexible scenarios for building ontologies and ontology networks. This set of scenarios is the basis for the NeOn Methodology framework presented in this PhD Thesis. The scenarios proposed are flexible because they can be combined among them in different ways and are perceived to be the opposite to the rigid scenario encountered when building ontologies from scratch and presented in METHONTOLOGY, On-To-Knowledge and DILIGENT. It is worth also mentioning that any combination of scenarios should include Scenario 1 because this scenario is made up of the core activities that have to be performed in any ontology development.

We have created a figure showing the 9 scenarios and their relations, and also have described all the scenarios following the same template. Such a description of scenarios includes the motivation, the assumptions, some prerequisite resources, the sequences of processes, activities and/or tasks to be carried out as part of the scenario, and the results or outcomes.

It should be noted that in the framework of the NeOn Methodology there are prescriptive methodological guidelines for carrying out processes and activities involved in Scenario 1 (ontology requirements specification and scheduling) and Scenario 3 presented in this thesis. In addition to the processes and activities presented in this thesis, there are also guidelines for Scenario 2 [Villazón-Terrazas, in progress], Scenario 5 [Suárez-Figueroa et al., 2010], Scenario 7 [Suárez-Figueroa et al., 2009], Scenario 8 (ontology modularization) [Suárez-Figueroa et al., 2009], and Scenario 9 [Espinoza, in progress]; and also for ontology evaluation [Suárez-Figueroa et al., 2009] and ontology evolution [Palma, 2009].

Chapter 11 presents how the scenarios identified have been followed in different use cases within European and educational projects.

7. Life Cycle Models for Ontology Networks

7.1. Introduction

In Software Engineering, every development project has a life cycle [Taylor, 2008], which is produced by instantiating a particular life cycle model. Life cycle models can be seen as abstractions of the phases or stages through which a product passes along its life. It is well known that there is no a unique life cycle model valid for all the developments [Taylor, 2008]. Examples of life cycle models are [Pfleeger, 2001; Sommerville, 2007]: waterfall, incremental, iterative, evolutionary prototyping, and rapid throwaway prototyping.

As stated in Chapter 1, the methodological support for developing ontologies should include the identification and definition of life cycle models and the life cycle. However, methodologies for building ontologies (METHONTOLOGY, On-To-Knowledge, and DILIGENT) have some limitations with respect to the aforementioned issues, as presented in Chapter 2, since they only propose a unique life cycle model.

Thus, as explained in Chapter 3, one of the goals of this PhD work is to provide the identification and definition of the life cycle models for networks of ontologies. In this regard, the following open research problem identified in Chapter 2 must be solved:

- There is only one ontology life cycle model (evolving prototypes), in contrast with the set of software life cycle models.

In this chapter we treat the definition of ontology network life cycle model and the identification of two different ontology life cycle models. Despite the diversity of models proposed in the literature for Software Engineering, we state that a collection of models consists of the waterfall model and the incremental-iterative model is enough to allow ontology developers to develop ontology networks.

Additionally, in this chapter we present the relation among the two models proposed and the set of nine flexible scenarios for collaboratively building ontologies and ontology networks, presented in Chapter 6. Models and scenarios have been created taking into account the importance of reusing and reengineering knowledge resources and merging resources, for this reason, the relation between them is established.

7.1.1. Ontology Network Life Cycle Model Definition

In Software Engineering, life cycle models [Taylor, 2008] can be seen as abstractions of the phases or stages through which a product passes along its life; in our case the product is an ontology network. Life cycle models are used to represent the entire life of a product from concept to disposal, and they could determine the order of the phases and establish the transition criteria between different phases. Each phase or stage should be described with a statement of purpose and outcomes.

Ontology network life cycle model is here defined as a model to describe how to develop (and maintain) an ontology network project; in other words, how to organize the processes and activities of the NeOn Glossary into phases or stages.

7.1.2. Preliminary Collection of Ontology Network Life Cycle Models

In the Software Engineering field it is acknowledged that there is no a unique standard life cycle model valid for all the software development projects, and that each life cycle model is appropriate for a concrete project, depending on several features [Taylor, 2008]. For example, sometimes it is better a simple model (like the waterfall one), whereas other times it is most suitable a spiral one (if the analysis of the risk is needed within the project).

In this PhD thesis, the claim, that is, ‘there is no a unique standard life cycle model valid for all the development projects, and that each life cycle model is appropriate for a concrete project, depending on several features’, is considered a premise for the Ontology Engineering field. Therefore, to propose a unique life cycle model for all the ontology network developments is not very realistic, according to different experiences in ontology developments in which different models were used. For this reason, we proposed in [Suárez-Figueroa et al., 2007; Suárez-Figueroa and Gómez-Pérez, 2008] a preliminary collection of ontology network life cycle models based on the life cycle models described and used in Software Engineering and taking into account the specific features of the ontology network development while assuming a rather centralized and controlled setting for ontology engineering in which approaches such as mining ontologies from tags were not regarded.

This preliminary collection of models included the following five ontology network life cycle models:

- *Waterfall model.* Its main characteristic is that it represents the stages of an ontology network as sequential phases. Thus, a concrete stage must be completed before the following stage begins.
- *Incremental model.* Its main feature is that it divides the requirements in different parts and then develops each part in a different cycle. The idea is to incrementally “produce and deliver” the network of ontologies (fully developed and functional), that is, the ontology network grows in layers centered in different topics (e.g., an ontology about project management in which first is represented the knowledge refers to publications, second, the knowledge about persons, and then, the knowledge about organizations and the project itself). Figure 30.a shows how an ontology network grows using this model (the stripped parts of the figure designate parts already developed).
- *Iterative model.* Its main characteristic is that it divides all the requirements into small parts and develops the ontology network, including requirements from all the parts. For example, in the first iteration we deal with concepts, and then in the next iteration we can improve the ontology with properties, etc. Figure 30.b shows how the ontology network is developed following this model (the stripped parts designate the parts developed).

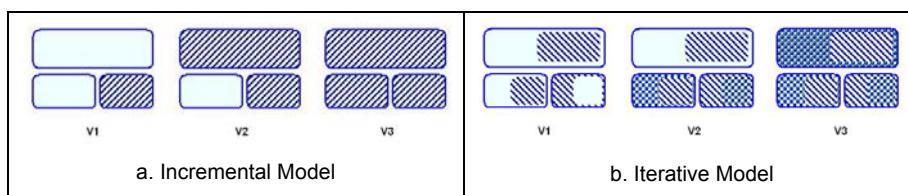


Figure 30. Schematic vision of an ontology network following (a) an incremental model and (b) an iterative model

- *Evolving prototyping model.* Its main feature is that it develops a partial product (in this case, a partial ontology network) that meets the requirements best understood. The preliminary versions of the ontology network being developed (that is, the prototypes) permit the user to give feedback of unknown or unclear requirements.

- *Spiral model.* Its main feature is that it proposes a set of repetitive cycles based on waterfall and prototype models. In this model, taking into account the special characteristics of ontology networks, the ontology engineering space is divided into three sections: planning, risk analysis, and engineering or development. This division is based on the need to evaluate and assess all the outputs of all the ontology network stages, and not only after the development phase as it happens in software projects.

In the preliminary evaluations of the uses of the models above proposed [Dzbor et al., 2009], it was obvious that ontology developers had some difficulties in distinguishing among incremental, iterative, evolving prototyping and spiral ontology network life cycle models. Some problems found during the selection among the aforementioned models refer to which model should be chosen when the requirements are not well understood or fully captured, and when the development should be carried out by parts.

Thus, taking into account (1) these preliminary evaluation results in which it was shown that the distinction among the different iterative models (incremental, iterative, evolving prototyping and spiral) is not easy for ontology developers; (2) the ideas presented by Larman [Larman, 2002], who proposed the existence of two different models: waterfall and iterative-incremental, and (3) the experience we had in ontology development in different research projects (Esperonto, Knowledge Web, SEEMP, etc.) in which the main models used were those based on waterfall ideas and on iterative ones, we decided to use these ideas that are valid in software engineering and apply them in ontology engineering, modifying the preliminary collection of ontology network life cycle models previously summarized by means of committing to only two different models:

- *Waterfall ontology network life cycle model.* A model representing the stages as a waterfall, where a concrete stage must be completed before the following stage begins and where backtracking is permitted from the maintenance phase to the phase after the requirements one.
- *Iterative-Incremental ontology network life cycle model.* Instead of distinguishing among incremental, iterative, evolving prototyping and spiral ontology network life cycle models, based on Larman's ideas [Larman, 2002], we propose here a unique iterative-incremental model. This approach is simpler, and thus, it will be easier to introduce in the ontology development by ontology developers.

In this chapter we present the collection of ontology network life cycle models proposed, which includes the enhanced waterfall model (Section 7.2) and the new iterative-incremental model (Section 7.3), in line with the rationale described in this section. Additionally, it is worth mentioning that these two models are intrinsically related to the set of nine flexible scenarios for collaboratively building ontologies and ontology networks, presented in Chapter 6. Such a relation is due to the creation of both models and scenarios, taking into account the importance of reusing and reengineering knowledge resources and merging resources.

7.2. Waterfall Ontology Network Life Cycle Models

The main characteristic of the waterfall life cycle model family proposed for the ontology network development is the representation of the stages of an ontology network as sequential phases. This model represents the stages as a waterfall. In this model a concrete stage must be completed before the following stage begins, and not backtracking is permitted except in the case of the maintenance phase.

The main assumption for using the waterfall ontology network life cycle model proposed is that the requirements are completely known, without ambiguities, and unchangeable at the beginning of the ontology network development.

This model could be used in the following situations:

- In ontology projects with a short duration (e.g., 2 months).
- In ontology projects in which the goal is to develop an existing ontology in a different formalism or language.
- In ontology projects in which the requirements are closed. E.g., to implement an ontology based on an ISO standard, or based on resources with previous consensus in the included knowledge.
- In ontology projects when ontologies should cover a small, well-understood domain.

Taking into account the characteristics of the ontology development, this model includes a set of support activities that should be performed in all of the phases. This set of support activities includes the acquisition of knowledge in the domain in which the ontology network is being developed, the evaluation (from a content-oriented perspective) and the assessment (from user and need perspectives) of the different phase outputs, project and configuration management and documentation.

Because of the importance of reusing and reengineering knowledge resources and merging ontological resources, we define and propose the following five significantly different versions of the waterfall ontology network life cycle model. These versions have been created incrementally (that is, the 4-phase is the basis for the 5-phase, the 5-phase is the basis for the 6-phase, etc.), as Figure 31 shows.

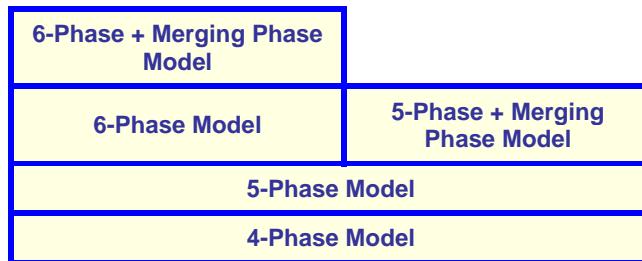


Figure 31. Pyramid of the versions of the Waterfall Ontology Network Life Cycle Model

Before detailing the different versions, we can summarize them in the following way:

- The 4-Phase Waterfall Model. It represents the stages of an ontology network, starting with the initiation phase and going through the design phase and the implementation phase to the maintenance phase.
- The 5-Phase Waterfall Model. It extends the 4-phase model with the reuse of ontological resources as they are.
- The 5-Phase + Merging Phase Waterfall Model. It is a special case of the 5-phase model. It includes the Merging Phase to obtain a new ontological resource from two or more ontological resources previously selected in the reuse phase.
- The 6-Phase Waterfall Model. It extends the 5-phase model with Reengineering Phase. It allows the reengineering of knowledge resources (ontological and non-ontological). It could happen that several knowledge resources are transformed into ontologies in the reengineering phase.
- The 6-Phase + Merging Phase Waterfall Model. It extends the 6-phase model by including the Merging Phase after the reuse phase.

□ **The 4-phase waterfall ontology network life cycle model.**

This model represents the stages of an ontology network, starting with the initiation phase and going through the design phase, the implementation phase to the maintenance phase.

The model proposed is shown in Figure 32, and the main purposes and outcomes for each phase in the model are the following:

- *Initiation Phase.* In this phase it is necessary to produce an ontology requirement specification document (ORSD) (explained in Chapter 8), including the requirements that the ontology network should satisfy and taking into account knowledge about the concrete domain. Also in this phase the approval or rejection of the ontology network development should be obtained. This phase has also as requisite to identify the development team and to establish the resources, responsibilities and timing (that is, the scheduling for the ontology project).
- *Design Phase.* The output of this phase should be both an informal model and a formal one that satisfy the requirements obtained in the previous phase. The formal model can not be used by computers, but it can be reused in other ontology networks.
- *Implementation Phase.* In this phase, the formal model is implemented in an ontology language. The output of this phase is an ontology implemented in RDF(S), OWL, or other language that can be used by semantic applications or by other ontology networks.

It is worth mentioning that the last two phases (design and implementation ones) are normally performed in parallel when ontology development tools (such as NeOn Toolkit, Protégé, etc.) are used.

- *Maintenance Phase.* If, during the use of the ontology network, errors or missing knowledge are detected, then the ontology development team should go back to the design phase. Additionally, in this phase the generation of new versions for the ontology network should be also carried out.

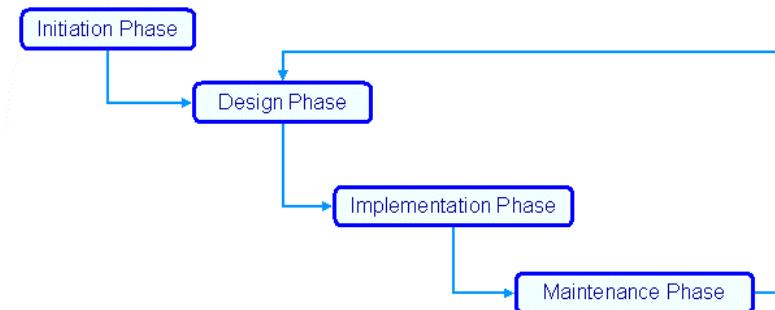


Figure 32. The 4-Phase Waterfall Ontology Network Life Cycle Model

□ **The 5-phase waterfall ontology network life cycle model.**

This model, shown in Figure 33, extends the 4-phase model with a new phase in which the reuse of already implemented ontological resources is considered. The main purpose in the *Reuse Phase* is to obtain one or more ontological resources to be reused in the ontology network being developed. The output of this reuse phase could be either an informal model or a formal one to be used in the modelling phase; or an implemented model (in an ontology language) to be used in the implementation phase.

For the other phases, the purposes and outcomes are the same as those presented in the 4-phase model.

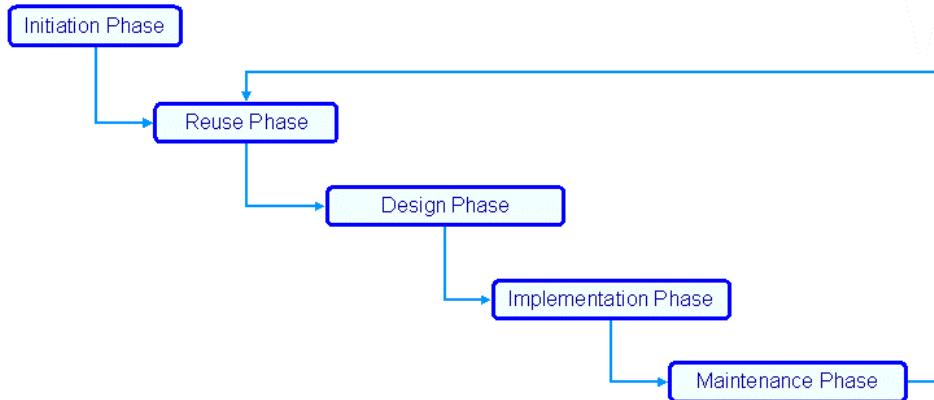


Figure 33. The 5-Phase Waterfall Ontology Network Life Cycle Model

□ **The 5-phase + merging phase waterfall ontology network life cycle model.**

Figure 34 shows the model proposed, which is a special case of the 5-phase model. Now, a new phase (the *Merging Phase*) is added after the reuse one. This merging phase has as a main purpose to obtain a new ontological resource from two or more ontological resources selected in the reuse phase.

For the other phases, the purposes and outcomes are the same as those presented in the 5-phase model.

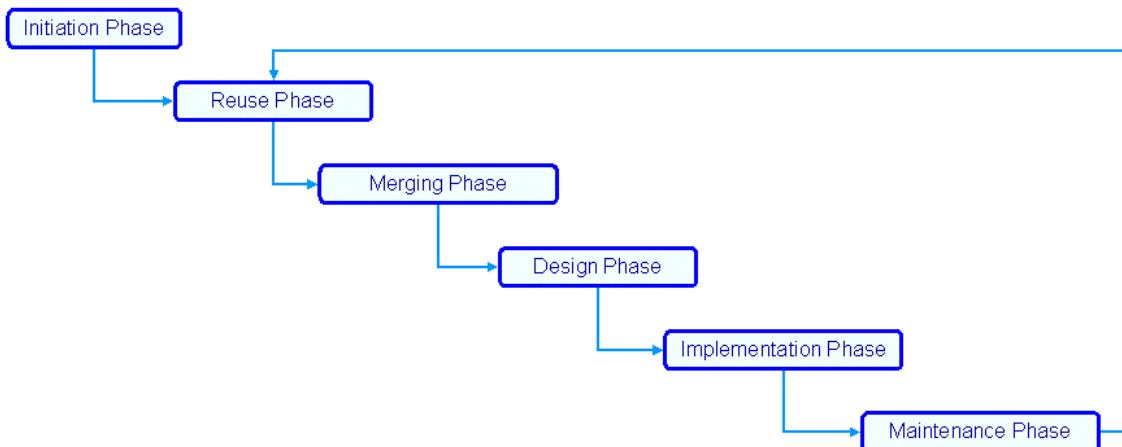


Figure 34. The 5-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

□ **The 6-phase waterfall ontology network life cycle model.**

In this model, shown in Figure 35, the 5-phase model is taken as general basis and a new phase (*Reengineering Phase*) is included after the reuse one. This model allows the reuse of knowledge resources (ontological and non-ontological) and their later reengineering. In this model the reuse phase has as output one or more knowledge resources to be reused in the ontology network that is being developed. After this phase, the non-ontological resources are transformed into ontologies in the reengineering phase; the ontological resources, on the other hand, can or cannot be reengineered, a decision that should be taken by the ontology development team.

For the other phases, the purposes and outcomes are the same as those presented in the 6-phase model.

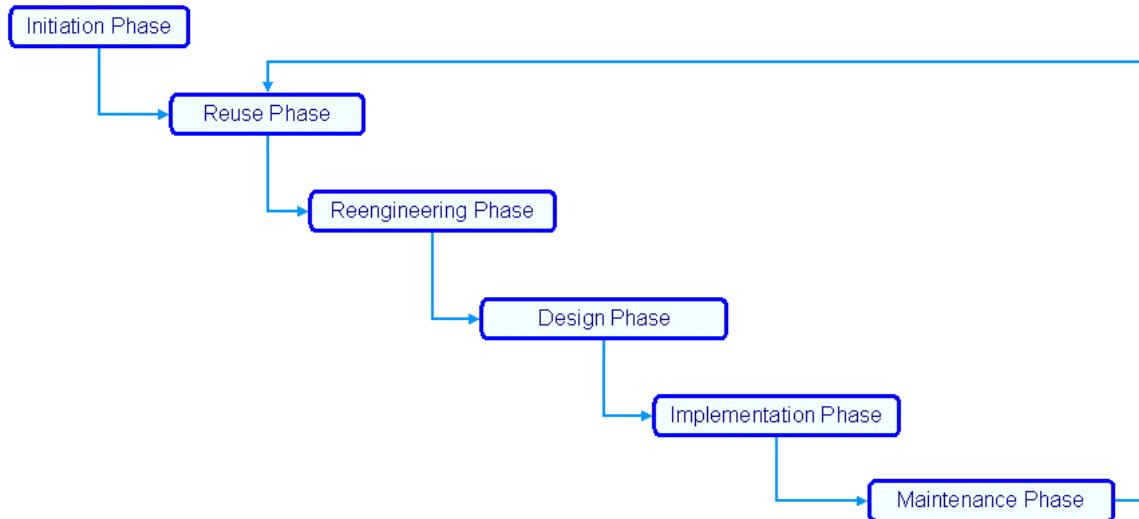


Figure 35. The 6-Phase Waterfall Ontology Network Life Cycle Model

□ **The 6-phase + merging phase waterfall ontology network life cycle model.**

This model, extended from the 6-phase model and shown in Figure 36, includes the *Merging Phase* after the reuse phase. For the other phases, the purposes and outcomes are the same as those presented in the 6-phase model.

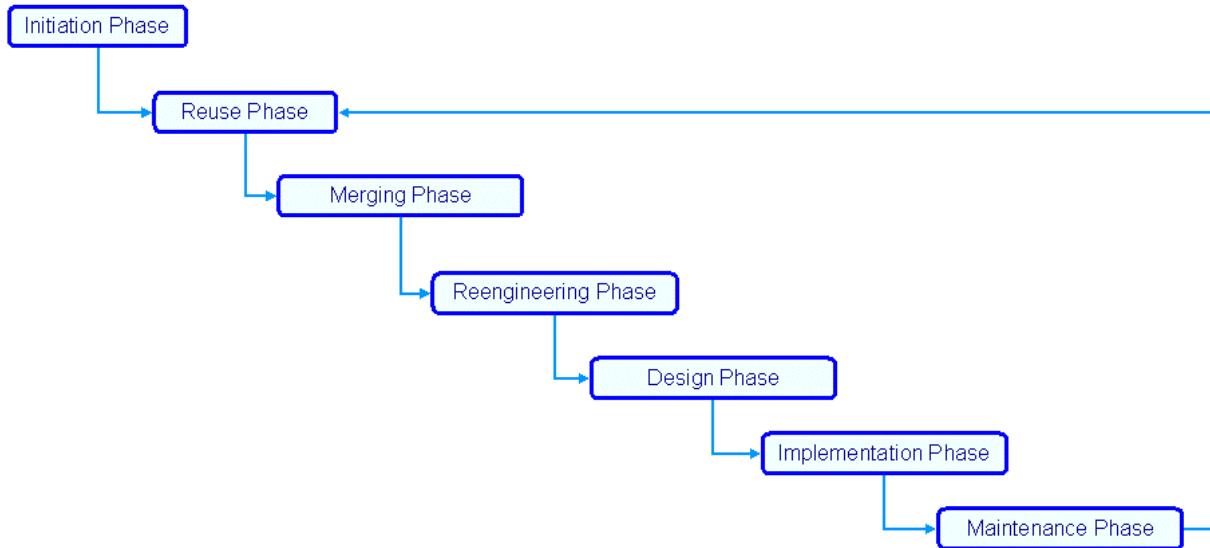


Figure 36. The 6-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

7.3. Iterative-Incremental Ontology Network Life Cycle Model

The main feature of this model family is the development of ontology networks organized in a set of iterations (or short mini-projects with a fixed duration). Each individual iteration is similar to an ontology network project that uses any type of waterfall model from those presented in Section 7.2, as shown schematically in Figure 37.

This model could be used in the following situations:

- In ontology projects with large groups of developers having different profiles and roles.
- In ontology project in which the development involves several different domains that are not well understood.
- In ontology project in which requirements are not completely known or can change during the ontology development.

Ontology requirements specified in the ORSD can be divided in different subsets. The result of any iteration is a functional and partial ontology network that meets a subset of the ontology network requirements. Such a partial ontology network can be used, evaluated and integrated in any other ontology network.

This model is based on the continuous improvement and extension of the ontology network resulted from performing multiple iterations with cyclic feedback and adaptation. In this way, the ontology network grows incrementally along the development. Generally, in each iteration new requirements are taken into account, but, occasionally, in a particular iteration the partial ontology network could be only enhanced.

This model focuses on a set of basic requirements; from these requirements, a subset is chosen and considered in the development of the ontology network. The partial result is reviewed, the risk of continuing with the next iteration is analysed, and the initial set of requirements is increased and/or modified in the next iteration until the complete ontology network is developed.

The main benefit of this model is to identify and alleviate the possible risks as soon as possible. Other benefits are:

- The development team is more motivated by having rapidly an ontology that can be used.
- Some priorities can be established in the set of requirements.
- The development can be possibly adapted to changes in the requirements.
- The scheduling of each iteration can be adapted based on the experience of previous iterations.

It is worth mentioning that at the beginning of the ontology network project, the number of iterations during the ontology project is influenced by

- The decision of performing a more complete and detailed ontology requirements specification. In this case the number of iterations will be lower.
- The decision of carrying out a simpler and more incomplete requirements specification; in which case more number of iterations and more revisions will be needed.

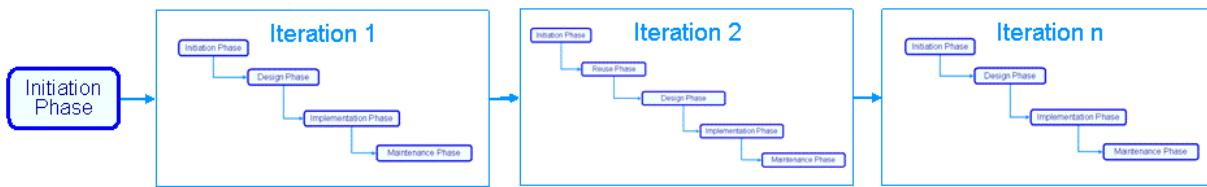


Figure 37. Schematic vision of the Iterative-Incremental Model

Figure 37 shows the schematic vision of the iterative-incremental model. The first initiation phase shown in the figure has as main outcomes the ontology network requirements and the general and global plan for the whole ontology network development. Regarding the different iterations, as mentioned before, each iteration in the iterative-incremental model can follow a different version of the waterfall model from those presented in Section 7.2. However, any version of the waterfall model to be used in the iterative-incremental model should be modified in the following way:

- No backtracking is allowed between phases in a particular iteration, because the refinement should be performed in the next iterations.
- Revising the ontology network requirements and the global plan should be carried out in the initiation phase of each iteration. Additionally, a detailed plan for the particular iteration should be performed.

7.4. Relation between Scenarios and Life Cycle Models

The set of nine flexible scenarios for building ontologies and ontology networks presented in Chapter 6, and the two proposed ontology network life cycle models presented in this chapter, are intrinsically related because both scenarios and life cycle models have been created (1) taking into account the importance of reusing and reengineering knowledge resources (ontological and non-ontological) and merging ontological resources, and (2) assuming a controlled setting for ontology engineering in which approaches such as mining ontologies from tags are not considered.

Table 8 summarizes the relationships between scenarios for building ontology networks and ontology network life cycle models. These relationships have been established based on the following:

- Scenario 1 (as stated in Section 6.2.1) is for building ontology networks from scratch. The scenario mainly includes core activities such as specification, conceptualization and implementation. This way of building ontologies fits together the stages represented in the 4-phase waterfall model (initiation phase, design phase, implementation phase, and maintenance phase).
- Scenario 2 (as stated in Section 6.2.2) is for building ontology networks by reusing and reengineering non-ontological resources, which is represented in the 6-phase waterfall model.
- Scenario 3 (as stated in Section 6.2.3) is for building ontology networks by reusing ontological resources. This way of building ontologies is represented by the 5-phase waterfall model.
- Scenario 4 (as stated in Section 6.2.4) refers to the development of ontology networks by reusing and reengineering ontological resources. This way of building ontologies is represented by the 6-phase waterfall model.
- Scenario 5 (as stated in Section 6.2.5) is for building ontology networks by reusing and merging ontological resources, which is represented by the 5-phase + merging phase waterfall model.

- Scenario 6 (as stated in Section 6.2.6) refers to the development of ontology networks by reusing, merging and reengineering ontological resources. This way of building ontology networks is represented by the 6-phase + merging phase waterfall model.
- Scenario 7 (as stated in Section 6.2.7) is for building ontology networks by reusing ontology design patterns, which is represented by the 5-phase waterfall model.
- Scenario 8 (as stated in Section 6.2.8) is for building ontology networks by restructuring ontological resources. This is mainly related to the core activities already mentioned in Scenario 1. Thus, this Scenario 8 is also represented by the 4-phase waterfall model.
- Scenario 9 (as stated in Section 6.2.9) refers to the development of ontology networks by localizing ontologies. This way of building ontologies is mainly related to Scenario 1, and thus represented by the 4-phase waterfall model.

As explained in Section 7.3, the iterative-incremental model is basically formed by a set of iterations that can follow any version of waterfall ontology network life cycle model. Thus, the relation between scenarios and the iterative-incremental model depends on the different versions of waterfall model used in the iterative-incremental one, and for this reason, the relations presented in Table 8 are also valid for this model.

	4-Phase Model	5-Phase Model	5-Phase + Merging Phase Model	6-Phase Model	6-Phase + Merging Phase Model
Scenario 1	X				
Scenario 2				X	
Scenario 3		X			
Scenario 4				X	
Scenario 5			X		
Scenario 6					X
Scenario 7		X			
Scenario 8	X				
Scenario 9	X				

Table 8. Relation between scenarios and life cycle models

7.5. Coverage of the Set of two Life Cycle Models

In this section we explain how the two models presented in Sections 7.2 and 7.3 are enough to cover the key features of the 5 preliminary models summarized in Section 7.1.2.

- The enhanced *waterfall model* is an improvement of the previous waterfall model.
- The *iterative-incremental model* covers the *incremental model* because requirements can be divided in different subsets, and each subset can be met in a different iteration of the iterative-incremental model.
- The *iterative-incremental model* covers the *iterative model* because different sets of the ontology network requirements can be met in successive iterations of the iterative-incremental model.
- The *iterative-incremental model* covers the *evolving prototyping model* because the requirements best understood can be met in the initial iterations of the iterative-incremental model. The preliminary versions of the ontology network being developed (that is, the

prototypes) permit the user to give feedback of unknown or unclear requirements, and thus to include new requirements.

- The *iterative-incremental model* covers the *spiral model* because requirements best understood can be met in repetitive iterations based on waterfall model; and a risk analysis can be performed during the initiation phases.

7.6. Conclusions

In this PhD thesis we have considered as a premise the claim made in the Software Engineering field that says that there is no a unique life cycle model valid for all the software development projects and that each life cycle model is appropriate for a concrete project, depending on several features [Taylor, 2008]. Therefore, we consider that is unrealistic to propose a unique life cycle model for all the ontology network developments in the Ontology Engineering field.

For this reason, we proposed [Suárez-Figueroa et al., 2007; Suárez-Figueroa and Gómez-Pérez, 2008] first a preliminary collection of ontology network life cycle models, based on the life cycle models described and used in Software Engineering, taking into account the specific features of the ontology network development. Such a collection of models include the following ontology network life cycle models: waterfall model, incremental model, iterative model, evolving prototyping model and spiral model.

However, in this chapter we state that such a preliminary collection could be reduced to a set of two different ontology network life cycle models (the enhanced waterfall model (Section 7.2) and the new iterative-incremental model (Section 7.3)). This set of two models is enough to allow ontology developers to develop ontology networks and covers the previous models in the preliminary collection, thus,

- The enhanced waterfall model covers the previous waterfall model. The 4-phase version is the basis for rest of versions of the waterfall model defined and described in Section 7.2.
- The iterative-incremental model covers the incremental, iterative, evolving prototyping, and spiral models.

Additionally, we have established the relationships among the scenarios described in Chapter 6 and these two ontology network life cycle models.

Finally, Chapter 11 presents the life cycle models in two different use cases within the NeOn project.

8. Ontology Requirements Specification

8.1. Introduction

One of the key processes in software development is software specification [Sommerville, 2007], whose aim is to understand and define what functionalities are required from the software product. It has been proved that a detailed software requirements document provides several benefits [IEEE, 1993], such as (a) the establishment of the basis for agreement between customers and suppliers on what the software product is supposed to do, (b) the reduction of the development effort, (c) the provision of a basis for estimating costs and schedules, and (d) the offer of a baseline for validation and verification.

When a software application based on ontologies is being developed, ontology requirements should be identified in addition to the application requirements. Our experience in building ontology-based applications, in domains as different as satellite data processing⁵⁵, finding funding programs⁵⁶, fishery stocks⁵⁷ and e-employment⁵⁸, has shown that more critical than capturing software requirements was the efficient and precise identification of the knowledge that the ontology should contain. Up to now, application developers already have precise methodologies [Sommerville, 2007; IEEE, 1993; Wiegert, 2003] that help them to define application requirements. However, the guidelines included in current methodologies for building ontologies are not enough for defining ontology requirements. Furthermore, so far there has not been a substantial contribution to the Grüninger and Fox's proposal [Grüninger and Fox, 1995] on the specification of ontology requirements based on CQs.

As stated in Chapter 3, one of the goals of this PhD thesis is to propose the NeOn Methodology for building collaboratively ontology networks. This methodology includes methods, techniques and tools for carrying out the processes and activities identified and defined in the ontology network development process, and it focused on ontology requirements specification, scheduling and ontological resource reuse.

In this regard, the following open research problem identified in Chapter 2 must be solved:

- No detailed prescriptive guidelines exist to carry out the ontology requirements specification activity.

In this PhD thesis, we propose efficient, precise prescriptive and detailed methodological guidelines for specifying ontology requirements. Such methodological guidelines are based on the use of the so-called CQs and are inspired by methodologies for building ontologies and by available practices and previous experiences in different national and European funded projects. These methodological guidelines help to capture knowledge from users and to produce the ontology requirements specification document (ORSD) that will be used by ontology developers to develop an ontology that will fulfil the requirements identified. Such a procedure for specifying ontology requirements should be clear, simple, useful, and applicable by ontology developers.

⁵⁵ <http://www.ontogrid.net>

⁵⁶ <http://esperonto.net/fundfinder>

⁵⁷ http://www.neon-project.org/nw/Ontology-driven_fish_stock_depletion_assessment_system

⁵⁸ <http://www.seemp.org>

8.2. Methodological Guidelines for Ontology Requirements Specification

The goal of the ontology requirements specification is to state why the ontology is being built, which its intended uses are, who the end-users are, and what are the requirements that the ontology should fulfil. For specifying the ontology requirements we will use the competency questions technique proposed in [Grüninger and Fox, 1995]. Before identifying the set of competency questions, we will identify the purpose and scope of the ontology, its level of formality, and its intended uses and end-users.

In the framework of the NeOn Methodology for building ontology networks, we propose the **Filling Card**, presented in Table 9, for the ontology requirements specification activity. The card includes the definition, goal, inputs and outputs, who carries out the activity and when the activity should be performed. Table 9 is an instantiation of the filling card template shown in Table 3.

Ontology Requirements Specification	
<i>Definition</i>	
<p><i>Ontology Requirements Specification</i> refers to the activity of collecting the requirements that the ontology should fulfil (for example, reasons to build the ontology, identification of target groups and intended uses). Such requirements may be reached through a consensus process.</p>	
<i>Goal</i>	
<p>The 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>
A set of ontological needs.	Ontology Requirements Specification Document (ORSD).
<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 at the beginning of the ontology project and in parallel with the knowledge acquisition activity.</p>	

Table 9. Ontology Requirements Specification Filling Card

The tasks for carrying out the ontology requirements specification activity can be seen in Figure 38. The result of this activity is the Ontology Requirements Specification Document (ORSD) that should be written following the ORSD template shown in Table 10.

Ontology Requirements Specification Document Template	
1 Purpose	<i>The general goal of the ontology. In other words, the main function or role that the ontology should have.</i>
2 Scope	<i>The general coverage and the degree of detail that the ontology should have.</i>
3 Implementation Language	<i>The formal language that the ontology should have.</i>
4 Intended End-Users	<i>The intended end-users expected for the ontology.</i>
5 Intended Uses	<i>The intended uses expected for the ontology.</i>
6 Ontology Requirements	a. Non-Functional Requirements <i>The general requirements or aspects that the ontology should fulfil, including optionally priorities for each requirement.</i> b. Functional Requirements: Groups of Competency Questions <i>The content specific requirements that the ontology should fulfil, in the form of groups of competency questions and their answers, including optionally priorities for each group and for each competency question.</i>
7 Pre-Glossary of Terms	a. Terms from Competency Questions <i>The list of terms included in the competency questions and their frequencies.</i> b. Terms from Answers <i>The list of terms included in the answers and their frequencies.</i> c. Objects <i>The list of objects included in the competency questions and in their answers.</i>

Table 10. Template for the OSRD

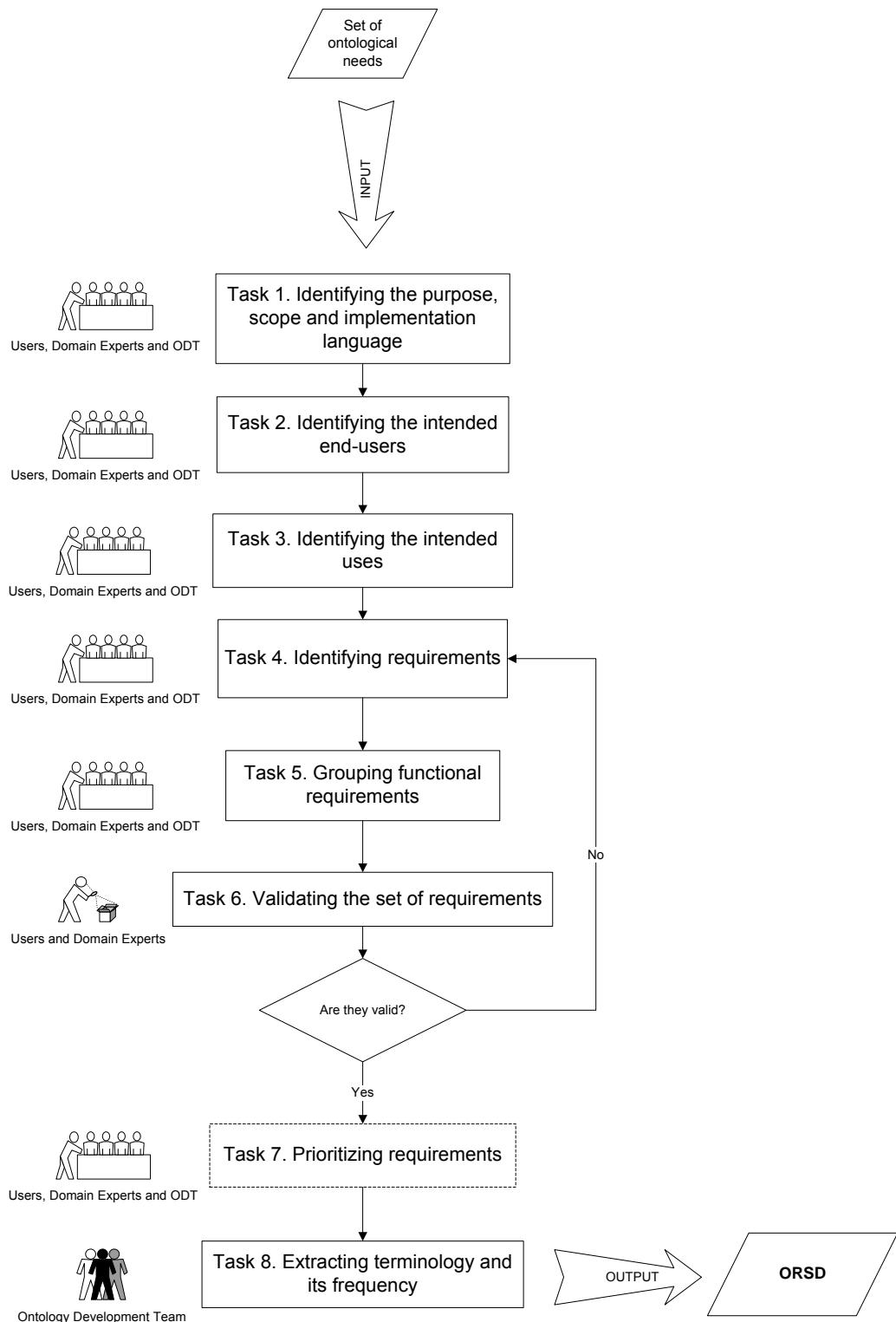


Figure 38. Tasks for Ontology Requirements Specification

The tasks for carrying out the ontology requirements specification activity are explained in detail next.

Task 1. Identifying the purpose, scope and implementation language.

The objective is to determine the main goal of the ontology, its coverage and foreseeable granularity, and its implementation language (e.g., OWL, RDFS⁵⁹, and WSML⁶⁰). The ontology development team holds a set of interviews with users and domain experts in order to carry out this task, taking as input a set of ontological needs. The users and domain experts are crucial to identify the purpose and scope of the ontology; on the other hand, the ontology developers should decide the formal language to be used for implementing the ontology.

The task output is included in slots 1-3 of the template shown in Table 10.

Task 2. Identifying the intended end-users.

The goal of this task is to establish who the intended main end-users of the ontology will be. The ontology development team holds a set of interviews with the users and domain experts to carry out this task, taking as input a set of ontological needs.

The task output is a list containing the intended end-users of the ontology to be built; the list is included in slot 4 of the template shown in Table 10.

Task 3. Identifying the intended uses.

The development of an ontology is mainly motivated by scenarios related to the application that will use the ontology. The goal of this task is to obtain the intended uses and use scenarios of the ontology. The ontology development team holds a set of interviews with the users and domain experts in order to carry out this task, taking as input a set of ontological needs; the purpose here is to obtain the uses of the ontology within the application, and to have a general idea of the application requirements, in terms of knowledge to be represented.

The task output is a list of intended uses in the form of scenarios, which is included in slot 5 of the template shown in Table 10. Such scenarios describe a set of general ontology requirements that the ontology should satisfy after being formally implemented. The scenarios should be described in natural language; they can be expressed in UML as use cases.

Task 4. Identifying requirements.

The goal of this task is to acquire the set of requirements that the ontology should satisfy. Taking as inspiration the Software Engineering field, in which requirements are divided into functional⁶¹ and non-functional⁶² requirements [Sommerville, 2007], we also divide ontology requirements into the following two types, whose definition is different from those in Software Engineering:

- Non-functional ontology requirements* refer to the characteristics, qualities, or general aspects not related to the ontology content that the ontology should satisfy. Examples of non-functional requirements are (a) whether the terminology to be used in the ontology must be taken from standards, (b) whether the ontology must be multilingual, or (c) whether the ontology should be written following a specific naming convention.

⁵⁹ <http://www.w3.org/TR/rdf-schema/>

⁶⁰ <http://www.wsmo.org/wsml/wsml-syntax>

⁶¹ Functional requirements refer to the required behaviour of the system, that is, the functionalities that the software system should have.

⁶² Non-functional requirements refer to implicit expectations about how well the software system should work. That is, these requirements can be seen as aspects about the system or as ‘non-behaviour’ requirements.

- Functional ontology requirements*, which can be seen as content specific requirements, refer to the particular knowledge to be represented by the ontology and the particular terminology to be included in the ontology. In the SEEMP case for example, the knowledge and the terminology about curriculum vitae with candidate skills, education level, expertise, previous work experience, or about job offers with information on job location, salary, etc.

The ontology development team should interview the users and domain experts, taking as input a set of ontological needs, and they should obtain as result the initial set of ontology requirements (non-functional and functional) of the ontology to be built. To identify functional requirements, they use as main technique the writing of the requirements in natural language in the form of the so-called CQs. They can use Mind map tools [Buzan, 1974] and Excel for gathering the requirements. If people are geographically distributed, they can employ wiki tools, such as Cicero.

Some strategies for identifying CQs are

- Top-Down: The team starts with complex questions that are decomposed in simpler ones.
- Bottom-Up: The team starts with simple questions that are composed to create complex ones.
- Middle out: The team starts just writing down important questions that are composed and decomposed later on to form abstract and simple questions respectively.

The output of this task is (1) a list of non-functional ontology requirements written in natural language, which is included in slot 6a of the template shown in Table 10, and (2) a list functional ontology requirements in the form of CQs and their associated answers, which is the input of task 5. This list of functional requirements will be grouped in Task 5 and then included in slot 6b of the template shown in Table 10.

Task 5. Grouping functional requirements.

The goal of this task is to group into several categories the list of functional ontology requirements in the form of CQs and their associated answers obtained in Task 4. The users, the domain experts, and the ontology development team should classify the list of CQs written in natural language with a hybrid approach that not only combines pre-established categories such as time and date, units of measure, currencies, location, languages, etc., but it also creates categories for those terms that appear with the highest frequencies in the list of CQs.

Techniques such as card sorting can be used when the grouping is done manually. In addition, mind map tools can help to display graphically and in groups the CQs or Cicero if the grouping is done collaboratively.

The task output is the set of groups of functional requirements in the form of CQs and their associated answers, which is included in slot 6b of ORSD template shown in Table 10.

Usually this task is carried out in parallel with Task 4.

Task 6. Validating the set of requirements. It includes both non-functional and functional requirements.

The aim here is to identify possible conflicts between ontology requirements, missing ontology requirements, and contradictions between them. Users and domain experts must carry out this task taking as input the set of requirements identified in Task 4 to decide if each element of the set is valid or not.

The task output is the confirmation of the validity of the set of non-functional and functional ontology requirements.

The criteria that can be used in this validation task and that are mainly inspired by [IEEE, 1993; Davis, 1993] are the following:

- Correctness.** A set of requirements is *correct* if, and only if, each requirement refers to some features of the ontology to be developed.
- Completeness.** Inspired by [Wieringa, 1996], a set of requirements can be considered *complete* if, and only if, users and domain experts review the requirements and confirm that they are not aware of additional requirements.
- Consistent.** A set of requirements can be considered internally *consistent* if, and only if, no conflicts exist between them.
- Verifiable.** A set of requirements is *verifiable* if, and only if, there is a finite process with a reasonable cost that tests whether the final ontology satisfies each requirement.
- Understandable.** Each requirement must be *understandable* to end-users and domain experts.
- No Ambiguity.** An ontology requirement is *unambiguous* if, and only if, it has only one meaning; that is, if it does not admit any doubt or misunderstanding.
- Conciseness.** A set of requirements is *concise* if, and only if, each and every requirement is relevant and no duplicated or irrelevant requirements exist.
- Realism.** A set of requirements is *realist* if, and only if, each and every requirement meaning makes sense in the domain.
- Modifiable.** A set of requirements is *modifiable* if, and only if, its structure and style allow changing issues in an easy, complete and consistent way.
- Traceable.** An ontology requirement is *traceable* if, and only if, its origin is known and it can be referred to in other documents during the ontology development.

Task 7. Prioritizing requirements.

The goal of this task is to give different levels of priority to the non-functional and functional ontology requirements identified. In the case of functional requirements, priorities should be given to the different groups of CQs, and, within each group, to the different CQs; additionally, priorities could be given to each CQs independently of the groups. Users, domain experts and the ontology development team should carry out this task, taking as input the requirements identified in Task 4 and the groups of CQs written in natural language obtained in Task 5. The task output is a set of priorities attached to each requirement, to each group of CQs, and to each CQ in a group. The output is included in the slots 6a and 6b of the template shown in Table 10.

Priorities will be used by the ontology development team for planning and scheduling the ontology development and for deciding which parts of the ontology are going to be developed first. This task is optional, but recommended. In fact, if no priorities are given to the groups of CQs, ontology developers will start modelling the ontology without any guidance regarding the functional requirements that should be implemented first; in this case the waterfall ontology life cycle model should be selected during the scheduling of the ontology project. On the contrary, if different priorities have been assigned to functional ontology requirements, the iterative-incremental ontology life cycle model should be selected in the scheduling activity.

Task 8. Extracting terminology and its frequency.

The goal of this task is to extract a pre-glossary of terms with their frequencies from the list of CQs and their answers identified in Task 4. The ontology development team carries out this task using terminology extraction techniques and tools supporting such techniques.

This pre-glossary of terms is divided in three different parts: terms from the CQs, terms from the CQs' answers, and terms identified as named entities.

- From the requirements in the form of CQs, we extract terminology (names, adjectives and verbs) that will be formally represented in the ontology by means of concepts, attributes, relations or instances (in the case of named entities).
- From the answers to the CQs, we extract terminology that could be represented in the ontology as concepts or as instances.
- From both CQs and corresponding answers, we extract named entities such as countries or currencies, which are objects in the universe of discourse.

The output is included in the slots 7a, 7b and 7c of the template shown in Table 10, respectively.

The set of terms with higher appearance frequencies will be used later on for searching knowledge resources that could be potentially reused in the ontology development. As heuristic, we can mention that normally the set of more frequent terms is that requires more effort during the ontology development; for this reason, frequencies are important to know which knowledge resources allow to save more effort.

8.3. Ontology Requirements Specification: Examples

In this thesis we provide three different examples of how to use the guidelines proposed for the ontology requirements specification activity and what results are expected from any of the tasks detailed in the guidelines. All the examples show an excerpt of the ORSD obtained after performing the ontology requirements specification activity following the methodological guidelines proposed in Section 8.2. Such examples are included in Annex I.

The first example refers to the requirements specification of the SEEMP Reference Ontology. It is important to mention that the work done within the SEEMP project in the ontology requirements specification activity has been one of the inputs to get preliminary guidelines for this activity. Such preliminary guidelines have been extended and improved in this thesis. Using the proposed guidelines, we described the requirements specification activity with the SEEMP reference ontology. This requirements specification is not intended to be exhaustive; it just describes the most important points. A detailed and complete requirements specification is described in [SEEMP Consortium, 2006].

The remaining two examples instantiate the methodological guidelines for the ontology requirements specification in the invoice use case and in the nomenclature use case within the NeOn project⁶³.

⁶³ http://www.neon-project.org/nw/Supporting_collaboration_in_pharmaceutical_industry

8.4. Conclusions

One of the critical activities when developing ontologies is to identify their functional and non-functional requirements. In this chapter we have systematized the ontology requirements specification activity by proposing detailed and prescriptive methodological guidelines for specifying ontology requirements, based on CQs, and by providing a template for writing the ontology requirement specification document.

Thus, we show along this chapter that while so far there have not been substantial contributions to the technique proposed by Grüninger and Fox [Grüninger and Fox, 1995] on the specification of ontology requirements based on CQs, a precise method, based on this approach, is established in this thesis. This method for specifying ontology network requirements can be applicable by ontology developers, and it allows identifying the main terms to be included later on in the ontology.

The ORSD will play a key role during the ontology development process because it facilitates different activities. In that sense, we will show in later chapters that the ontology requirements specification document (1) is a crucial input for the scheduling of ontology development projects and (2) facilitates, among other activities, the search and reuse of non-ontological resources for reengineering them into ontologies (such as, lexicons, glossaries, and dictionaries); the search and reuse of ontologies, ontology modules, ontology statements (e.g., using Watson or Swoogle), and ontology design patterns; and the verification of the ontology during the whole ontology development.

Finally, we carried out a set of experiments whose aim was to test the methodological guidelines proposed for the ontology requirements specification activity. The main results and conclusions of such experiments are presented in Chapter 11.

The samples presented in Annex I and the experiments carried out and presented in Chapter 11 show that as in other disciplines, the requirements specification (a) establishes the basis for agreement between the users and ontology developers, (b) establishes clearly what the ontology development team should represent in the ontology, what possibly reduces the development effort, (c) provides a basis for estimating costs and schedules, and (d) offers a baseline for verification.

9. Planning and Scheduling: Obtaining the Ontology Network Life Cycle

9.1. Introduction

Planning and scheduling are related activities that are applied in different contexts such as civil engineering, software development, etc. While planning⁶⁴ is the act of drawing up plans, that is, a series of steps to be carried out to achieve an objective, scheduling⁶⁵ is defined as the activity to set order and time to planned events. Scheduling should be performed after planning; and both are crucial in any project.

In Software Engineering, every development project has a life cycle [Taylor, 2008], which is produced by instantiating a particular life cycle model. Life cycle models can be seen as abstractions of the phases or stages through which a product passes along its life. Examples of life cycle models are waterfall, incremental, iterative, evolutionary prototyping, and rapid throwaway prototyping, as described in Section 2.3.2.

To properly manage software development projects, it is crucial to have knowledge of the entire software development life cycle [Stellman and Greene, 2005]. In this regard, software engineers always plan and schedule every development project before starting it. The project plan defines the tasks to be done and establishes the human resources to perform the project work. To estimate the effort required to perform each task, techniques such as [Stellman and Greene, 2005] Wideband Delphi, PROBE and COCOMO II can be used.

The project schedule is a calendar that links the tasks to be done with the resources to support their performance. The most common form of representing schedules is to use a Gantt chart [Stellman and Greene, 2005]; and the most popular tool for creating a project schedule is Microsoft Project [Stellman and Greene, 2005]. Such a tool allows managing and scheduling different types of projects. To create a project schedule, Microsoft Project provides three different ways: (a) from scratch by adding tasks, phases, etc.; (b) from a project template (engineering template, commercial construction template, annual report preparation template, etc.) selected by the user using a template library; and (c) from an existing project.

However, unlike what happens in Software Engineering, in the Ontology Engineering field planning and scheduling ontology developments are still in their early stages.

The Ontology Engineering field lacks methods to guide ontology developers when planning and scheduling their ontology development projects. Furthermore, there are no project templates oriented to ontology development projects in the template library of Microsoft Project and nor is there an ad-hoc support tool for providing ontology developers with ontology project schedules in the form of a Gantt chart.

In this regard, in this chapter we propose solutions to the following open research problems:

- There are no detailed prescriptive guidelines to select a specific life cycle model and thus to create a particular ontology life cycle, as part of the scheduling activity.
- There is no ad-hoc support tool for scheduling ontology development projects.

⁶⁴ <http://www.wordnet-online.com/planning.shtml>

⁶⁵ <http://www.wordnet-online.com/scheduling.shtml>

Despite the fact that until now nobody has made any proposal on how to carry out the planning and scheduling activities, the set of processes and activities identified in Chapter 5 and the life cycle models proposed in Chapter 7 are the key to bridging this gap. Thus, we should probe that it is possible to establish a clear, simple, and useful procedure to select and instantiate an ontology network life cycle model and obtain the ontology network life cycle, which is the basis for ontology development projects schedules. This procedure should be applicable by ontology developers.

Therefore, the contributions of this chapter to the thesis are (1) the groundings for scheduling ontology network development projects, (2) the creation of the gOntt plug-in, a tool that supports the scheduling of ontology network development projects and helps to execute such projects, and (3) the definition of prescriptive methodological guidelines for scheduling ontology development projects using gOntt.

9.2. Scheduling Ontology Development Projects

Scheduling, as defined in the NeOn Glossary of Processes and Activities included in Section 5.3, refers to the activity of identifying the different processes and activities to be performed during the ontology development, their arrangement, and the time and resources needed for their completion. Thus, this activity includes as an important task the establishment of the **ontology network life cycle** that is the specific ordered sequence of processes and activities that ontology developers carry out during the life of the ontology network.

The goal of scheduling is to organize the different processes and activities in time, that is, to state a concrete programming or scheduling that guides the ontology network development, including processes and activities, their order and time, as well as human resources restrictions.

To establish the concrete schedule for the ontology network development, four important questions have to be answered:

- 1) Which ontology network life cycle model is the most appropriate for the ontology network development?
- 2) Which particular processes and activities should be carried out in the ontology network development?
- 3) Which order and dependencies exist among processes and activities?
- 4) How many resources (human and time) are needed for the development of the ontology network?

The first three questions are related to the establishment of the ontology network life cycle, and their responses would result in a general plan for the ontology network development. The fourth question, which is out of the scope of this thesis, is related to the inclusion of time and human resources restrictions for each process and activity included in the plan, and its response would result in the concrete schedule for the ontology network development. The information about how many people should be involved in the ontology network development can be obtained using the ONTOCOM model [Simperl et al., 2009]. This is a cost estimation model, whose goal is to predict the costs (expressed in person per month) arising in typical ontology engineering processes. However, this model does not provide the duration neither the resources needed for each process and activity planned.

As in the case of software engineering projects, an ontology engineer can not start the ontology project scheduling without having identified first the ontology requirements. Before scheduling the ontology development, it is also advisory to carry out a quick search for available knowledge resources (ontologies, thesauri, lexicons, etc.) in order to possibly reuse them, avoiding the construction of ontologies from scratch. Such resources could be taken from the internet, standardization bodies (e.g., ISO), the intranet of the organization, ontology gateways or registries, etc. This approach is already explained in Section 6.2.1.

The **Filling Card** for the scheduling activity, presented in Table 11, includes the definition, goal, inputs and outputs, performer of the activity and time of the activity. This concrete filling card follows the template presented in Section 4.1 (Table 3).

Scheduling	
<i>Definition</i>	
<p><i>Scheduling</i> refers to the activity of identifying the different activities and processes to be performed during the ontology development, their arrangement, and the time and resources needed for their completion.</p>	
<i>Goal</i>	
<p>The scheduling activity states a concrete programming or scheduling to guide the ontology network development, including processes and activities and the order in which they appear, and time and human resources restrictions and assignments.</p>	
<i>Input</i>	<i>Output</i>
Ontology Requirements Specification Document (ORSD) and types of potential knowledge resources to be reused.	Schedule for the ontology network development.
<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 after the ontology requirements specification activity and after performing a quick search for existing and available knowledge resources.</p>	

Table 11. Scheduling Filling Card

9.3. Groundwork for Scheduling Ontology Development Projects

We propose to carry out the scheduling of ontology development project based mainly on the scenarios identified in Chapter 6. For this reason and to be able to answer the first three questions presented in Section 9.2, we did some research that yielded the following results:

- A set of simple questions that help to select (a) the most appropriate version of the waterfall ontology network life cycle model and (b) the set of scenarios to be followed in the development.
- The correspondences between the scenarios and the processes and activities to be carried out in the ontology development.
- The correspondences between the processes and activities and the phases of the ontology life cycle model.
- The order and dependencies between processes and activities.

These results are the basis for the creation of the guidelines and the tool for scheduling ontology development projects.

9.3.1. Scenarios and Life Cycle Models

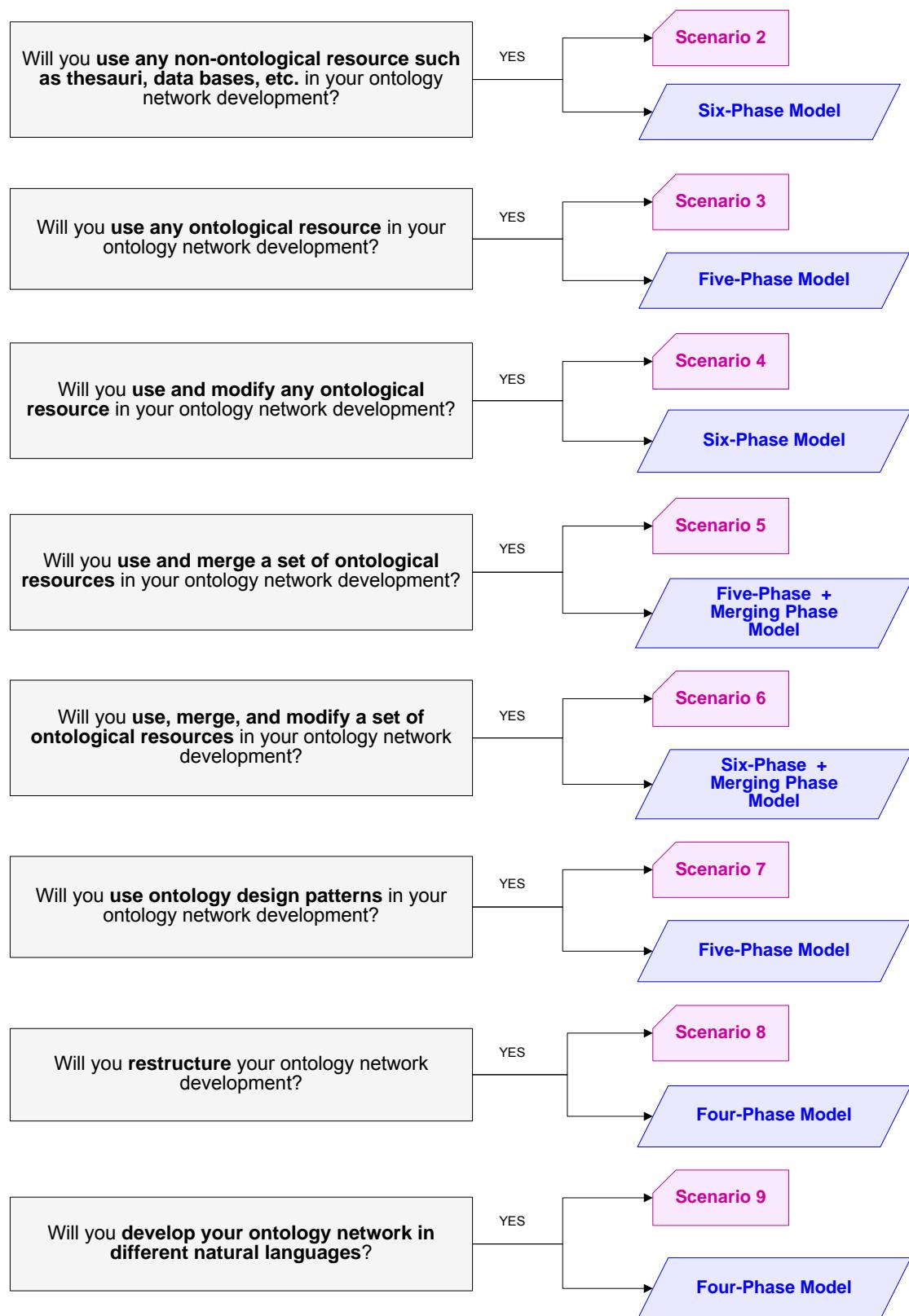
As presented in Chapter 7, we propose two life cycle models: the waterfall and the iterative-incremental. The waterfall model has five different versions (as presented in Section 7.2). In the iterative-incremental model, each iteration can follow any of the different versions of waterfall model. Note that in Section 7.4, we present a table (Table 8) that shows the relationships between scenarios for building ontology networks and some ontology network life cycle models.

To select a particular waterfall model version, we propose the set of natural language questions displayed on the left hand-side of Figure 39. These questions are related to the different scenarios identified in the NeOn Methodology presented in Chapter 6. If one or more questions of those proposed in Figure 39 are answered affirmatively, then several candidate models could be used. In this case, the model version selected should be the most specific one, based on the pyramid shown in Figure 31. Otherwise, if all answers are negative, then the 4-phase waterfall model is selected by default.

These questions are also useful for selecting the set of scenarios that are to follow during the ontology network development. Additionally, we identified the following set of restrictions among scenarios:

- If Scenario 4 should be executed, then Scenario 3 should be also executed.
- If Scenario 5 should be executed, then Scenario 3 should be also executed.
- If Scenario 6 should be executed, then Scenario 5 should be also executed, and transitively Scenario 3 should be also executed.

Finally, it is worth mentioning that Scenario 1 should be always included in the set of scenarios obtained for the development because, as explained in Chapter 6, Scenario 1 is mandatory. Additionally, we should mention that the scenarios can be combined.

**Figure 39. Decision tree for selecting model version and scenarios**

9.3.2. Scenarios and Processes and Activities

To obtain the set of processes and activities that will be used during the ontology network development, we propose a table of correspondences between scenarios and processes and activities. The table is presented in Section 6.3 (Table 7).

9.3.3. Processes, Activities, and Life Cycle Models

Processes and activities should be carried out in a particular phase of the selected ontology network life cycle model to fulfil the purpose and outcome of that phase. To obtain the mapping between processes and activities and phases in the ontology network life cycle model, we propose Table 12, which matches the process and activity outputs against the purpose and outcome of each phase of the ontology network life cycle model. To do this, we took into account the following phases in our repository of models: design phase, implementation phase, initiation phase, maintenance phase, merging phase, reengineering phase, and reuse phase. Table 12 relates processes and activities to model phases in the following way: (1) a process or an activity should be carried out in a particular model phase; or (2) a process or an activity should be carried out in all the phases of the model.

9.3.4. Order of Processes and Activities

The preliminary order in which processes and activities should be performed is determined by the following major factors:

- a) The selected ontology network life cycle model dictates an initial ordering of processes and activities, based on the order in which model phases should be performed. The order of the model phases is presented in figures that correspond to each ontology life cycle model described in Chapter 7.
- b) The availability of output information from one process or activity could affect the start of another process or activity. The second process or activity might require, as inputs, one or more of the outputs of the first one. For example, the ontology requirements specification activity should be performed before the scheduling one, as already mentioned in Section 6.2.1. Another example could be the non-ontological resource reuse that should be carried out before the non-ontological resource reengineering.
- c) Processes and activities might be executed in parallel. For example, the ontological resource reuse could be performed in parallel with the non-ontological resource reuse rather than for serial execution if there are enough human resources.

These restrictions have been represented in scheduling templates in the form of Gantt charts. We identified and represented 112 templates that can be used as preliminary schedules. Figure 40 shows the template for the 4-phase waterfall model with Scenarios 1 and 9; Figure 41 shows the template for the 6-phase waterfall model with Scenarios 2 and 3; and Figure 42 shows the template for the 6-phase waterfall model with Scenarios 3, 4, 7, and 8.

<u>Process or Activity</u>	<u>Corresponding Phase</u>
<i>Ontology Aligning</i>	Merging Phase
<i>Ontology Annotation</i>	All Phases
<i>Ontology Assessment</i>	All Phases
<i>Ontology Comparison</i>	Reuse Phase
<i>Ontology Conceptualization</i>	Design Phase
<i>Ontology Configuration Management</i>	All Phases
<i>Control</i>	All Phases
<i>Ontology Customization</i>	Reengineering Phase
<i>Ontology Design Pattern Reuse</i>	Reuse Phase
<i>Ontology Diagnosis</i>	All Phases
<i>Ontology Documentation</i>	All Phases
<i>Ontology Elicitation</i>	All Phases
<i>Ontology Enrichment</i>	Reengineering Phase
<i>Ontology Environment Study</i>	Initiation Phase
<i>Ontology Evaluation</i>	All Phases
<i>Ontology Evolution</i>	Design Phase
<i>Ontology Extension</i>	Reengineering Phase
<i>Ontology Feasibility Study</i>	Initiation Phase
<i>Ontology Formalization</i>	Design Phase
<i>Ontology Forward Engineering</i>	Reengineering Phase
<i>Ontology Implementation</i>	Implementation Phase
<i>Ontology Integration</i>	Design Phase
<i>Knowledge Acquisition for Ontologies</i>	All Phases
<i>Ontology Learning</i>	All Phases
<i>Ontology Localization</i>	Design Phase
<i>Ontology Mapping</i>	Merging Phase
<i>Ontology Matching</i>	Merging Phase
<i>Ontology Merging</i>	Merging Phase
<i>Ontology Modification</i>	Design Phase
<i>Ontology Modularization</i>	Reengineering Phase
<i>Ontology Module Extraction</i>	Reengineering Phase

<u>Process or Activity</u>	<u>Corresponding Phase</u>
<i>Ontology Module Reuse</i>	Reuse Phase
<i>Ontology Partitioning</i>	Reengineering Phase
<i>Ontology Population</i>	All Phases
<i>Ontology Pruning</i>	Reengineering Phase
<i>Ontology Quality Assurance</i>	All Phases
<i>Ontology Reengineering</i>	Reengineering Phase
<i>Ontology Repair</i>	All Phases
<i>Ontology Requirements Specification</i>	Initiation Phase
<i>Non-Ontological Resource Reengineering</i>	Reengineering Phase
<i>Non-Ontological Resource Reverse Engineering</i>	Reengineering Phase
<i>Non-Ontological Resource Transformation</i>	Reengineering Phase
<i>Ontology Restructuring</i>	Reengineering Phase
<i>Non-Ontological Resource Reuse</i>	Reuse Phase
<i>Ontological Resource Reuse</i>	Reuse Phase
<i>Ontology Reuse</i>	Reuse Phase
<i>Ontology Reverse Engineering</i>	Reengineering Phase
<i>Scheduling</i>	Initiation Phase
<i>Ontology Search</i>	Reuse Phase
<i>Ontology Selection</i>	Reuse Phase
<i>Ontology Specialization</i>	Reengineering Phase
<i>Ontology Statement Reuse</i>	Reuse Phase
<i>Ontology Summarization</i>	All Phases
<i>Ontology Translation</i>	Implementation Phase
<i>Ontology Update</i>	Design Phase
<i>Ontology Upgrade</i>	Maintenance Phase
<i>Ontology Validation</i>	All Phases
<i>Ontology Verification</i>	All Phases
<i>Ontology Versioning</i>	Maintenance Phase

Table 12. Correspondence between processes and activities and ontology network life cycle model phases

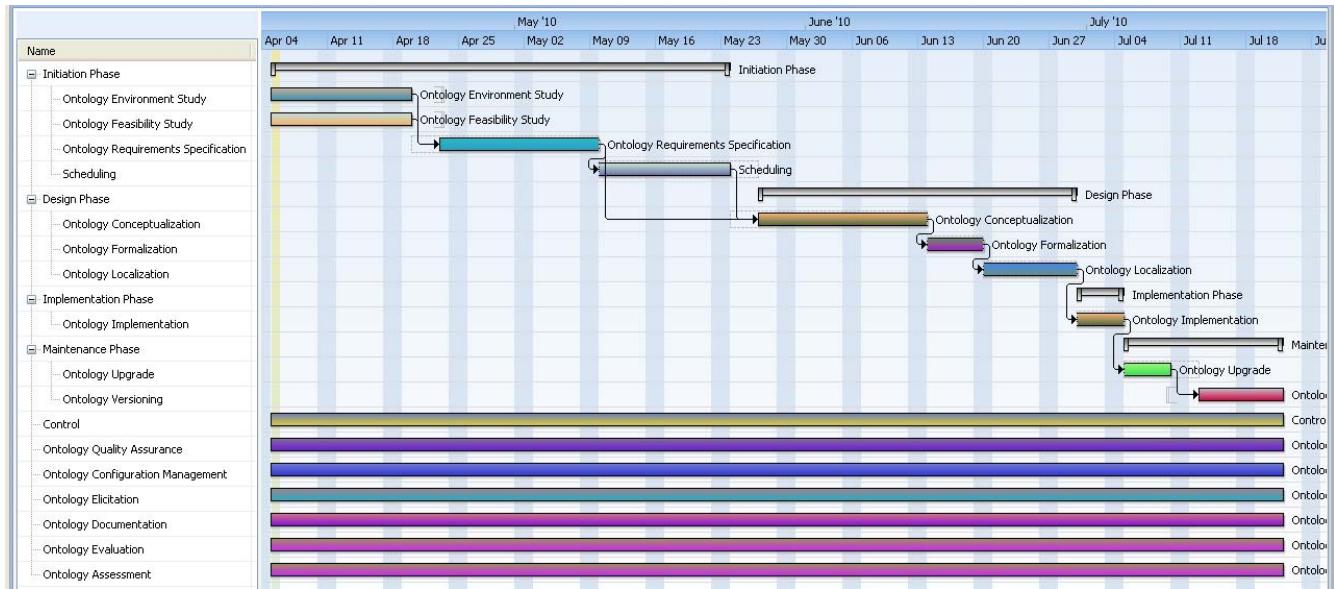


Figure 40. Example of a scheduling template for the 4-Phase Waterfall Model with Scenarios 1 and 9

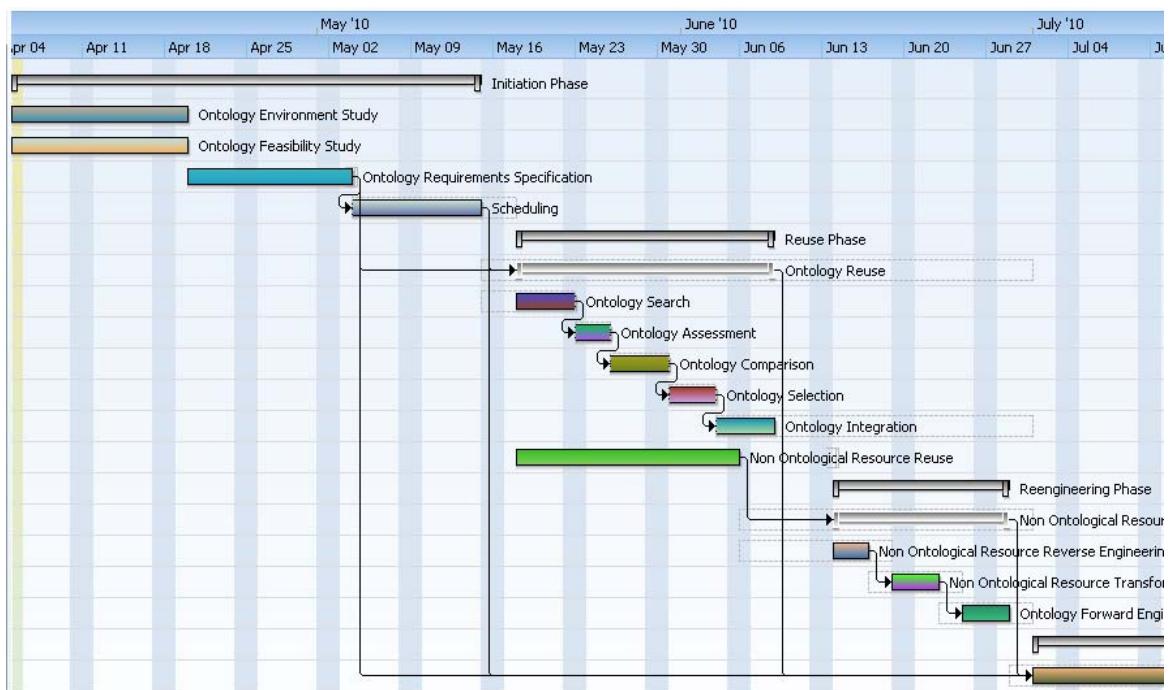


Figure 41. Example of a scheduling template for the 6-Phase Waterfall Model with Scenarios 2 and 3

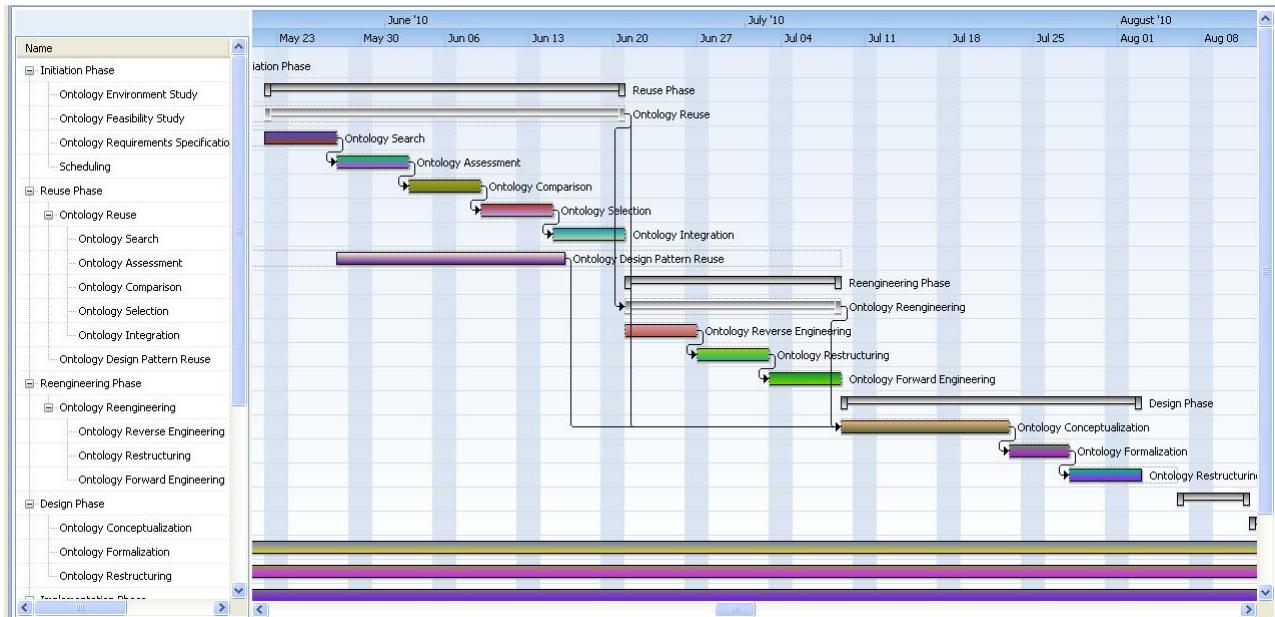


Figure 42. Example of a scheduling template for the 6-Phase Waterfall Model with Scenarios 3, 4, 7, and 8

9.4. gOntt: NeOn Toolkit plug-in for the Scheduling Activity

To support the scheduling of ontology development projects, following the methodological foundations presented in Section 9.3, we have created **gOntt** as a NeOn Toolkit plug-in. The gOntt plug-in has the following main objectives:

1. To support ontology developers in deciding which ontology network life cycle model is the most appropriate for building their ontologies.
2. To help ontology developers to decide which concrete process and activities should be carried out in the ontology network development and in which order.
3. To instantiate the life cycle model selected and to create a particular life cycle for the ontology development with the processes and activities needed, including time restrictions between processes and activities.
4. To inform ontology developers about how to carry out a particular process or activity through the NeOn methodological guidelines and a reference to the concrete NeOn plug-ins to be used. That is, to help ontology developers in the ontology project execution.

According to the aforementioned objectives, gOntt functionalities can be divided in two main groups: *functionalities for scheduling ontology development projects* and *functionalities for helping in the execution of ontology development projects*.

The functionalities for *scheduling an ontology network development* are

- To create particular schedules from scratch, by allowing the ontology developer the inclusion of processes, activities, phases and relationships and restrictions between them. Such processes and activities could either come from the NeOn Glossary of Processes and Activities or be new ones proposed by the developer.

- To create particular schedules in a guided way. gOntt creates preliminary plans for the ontology development with a simple two-step wizard. The ontology developer uses the wizard to answer a set of simple and intuitive questions that implicitly allow him to select the ontology life cycle model and the processes and activities to be carried out. This way of working is different from that of the Microsoft Project, in which the user should directly select a general project template (e.g., engineering template, commercial construction template, annual report preparation template, etc.) from a template library.

gOntt internally uses the methodological foundations explained in Section 9.3, including the scheduling templates presented in Section 9.3.4 to automatically generate the initial plan. Such scheduling templates show ontology project default plans based on the different and possible combinations among life cycle models, scenarios, and processes and activities.

gOntt provides the user with an initial plan in the form of a Gantt chart that the user can modify in the following fashion: (a) by including or deleting processes and activities, (b) by changing order and dependencies among processes and activities, and (c) by including resource assignments and restrictions to the planned processes and activities (this possibility is out of the scope of this thesis).

- To create, modify, and delete gOntt projects.
- To save and open gOntt projects in an extension (.got) based on an xml standard.
- To provide graphical and textual visualizations of gOntt projects.
- To delete processes, activities and phases from a gOntt project.
- To modify the names of the processes, activities and phases and their order in a gOntt project.
- To create, modify and delete connections between activities, between processes, and between activities and processes. If a connection exists between two elements, the second one cannot start until the first connection finishes. These connections can have two different meanings: logical dependencies and temporal dependencies.
 - Logical dependencies: when it is required that one activity is carried out before another because of the nature of the activities (e.g., diagnosis before repair in ontology validation).
 - Temporal dependencies: when an activity should be performed after another because of project needs (e.g., ontology reuse and non-ontological reuse can be carried out in parallel because they have no restrictions between them but, in some cases, there are no enough human resources to perform the activities in parallel and so they should be planned to be performed in sequence).
- To include and modify the duration and the starting date of the processes, activities and phases.

The functionalities for *helping in the execution of ontology development projects* are

- ❑ To provide the developer with some methodological guidelines for the processes and activities identified in the NeOn Methodology, and thus
 - To display a **filling card**, which includes the process or activity definition, its goal, inputs and outputs, performer of the process or activity, and time of the performance. Figure 43 shows an example of the filling card for the ontology localization activity.

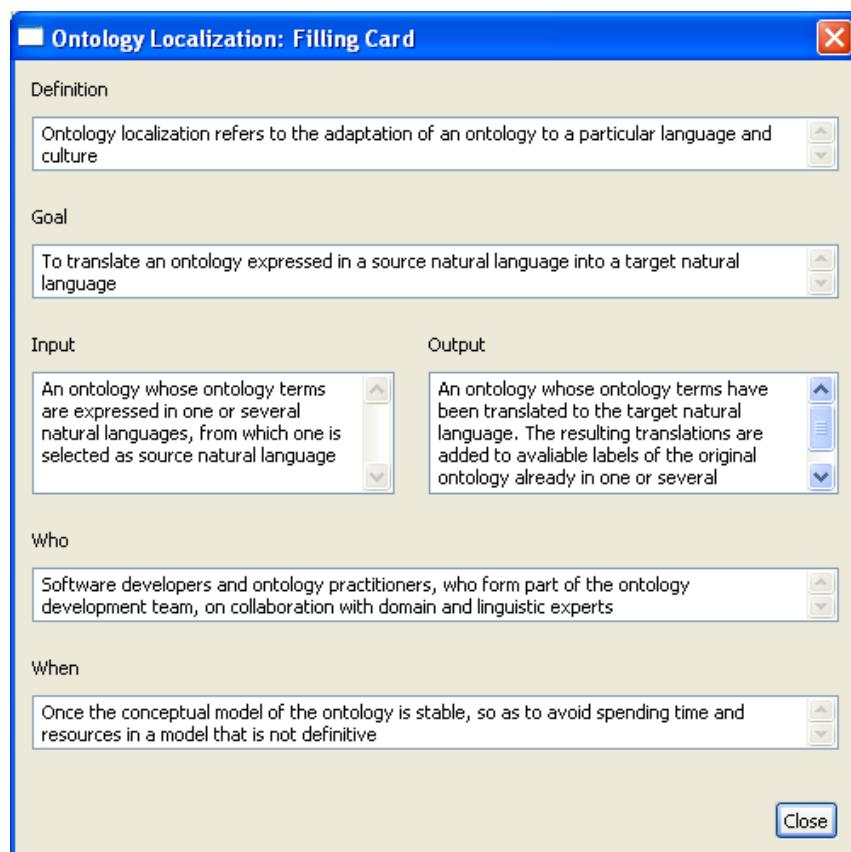


Figure 43. Example of the Ontology Localization Filling Card in gOntt

- To display a **workflow** and some **methodological guidelines** explaining how the process or the activity should be carried out, including its inputs, outputs and actors involved. Figure 44 shows an example of the methodological guidelines for the ontology localization activity. Workflows are implemented with Eclipse Cheat Sheets⁶⁶.

⁶⁶ <http://www.ibm.com/developerworksopensource/library/os-ecl-cheatsheets>

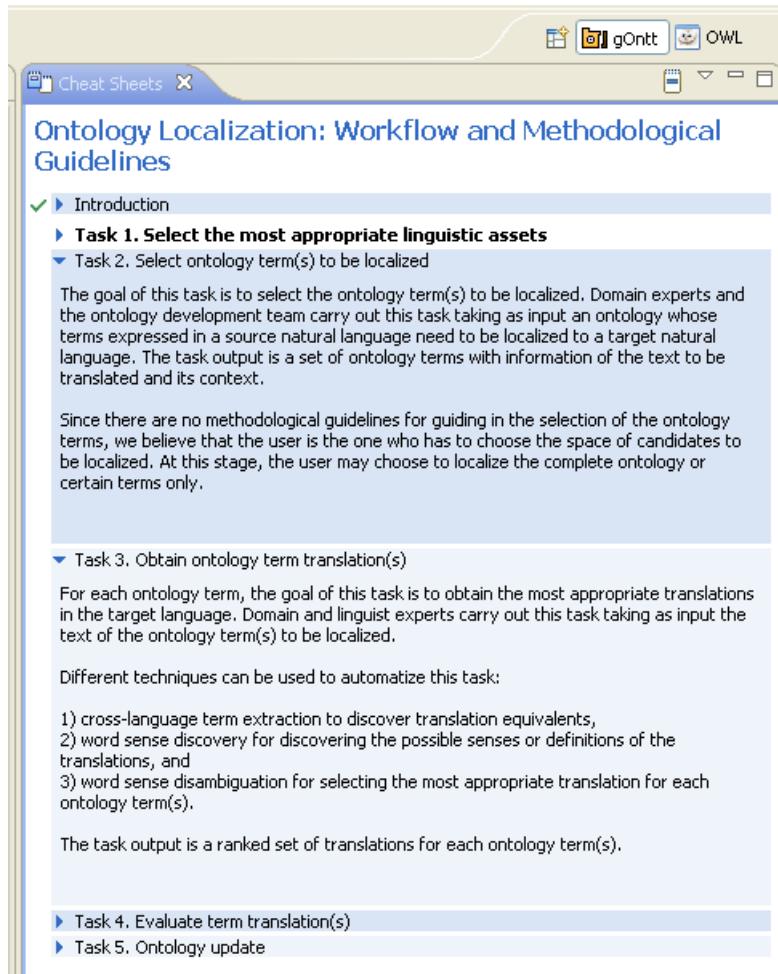


Figure 44. Example of the workflow and of some methodological guidelines for the Ontology Localization activity in gOntt

- To provide a direct access to the NeOn plug-ins associated to each process and activity planned. This means that gOntt triggers the different NeOn Toolkit plug-ins associated to each process or activity included in the plan.

Figure 45 shows the general appearance of the gOntt plug-in, whereas Figure 46 shows a specific plan in which it is worth mentioning that in the reuse phase, ontological and non-ontological resource reuses can be replicated for each different ontological and non-ontological resource used in the ontology development; the same can be said in the reengineering phase for the non-ontological resource reengineering process.

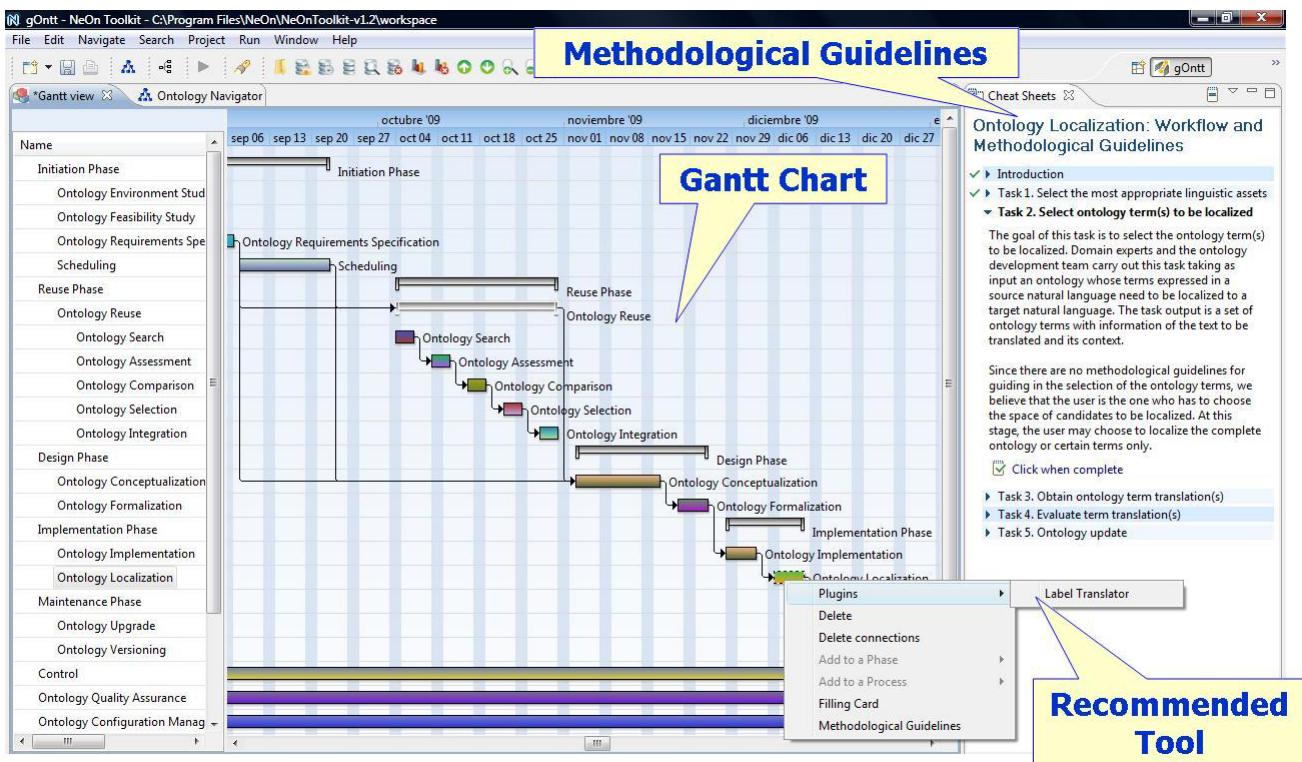


Figure 45. Screenshot of the gOntt plug-in

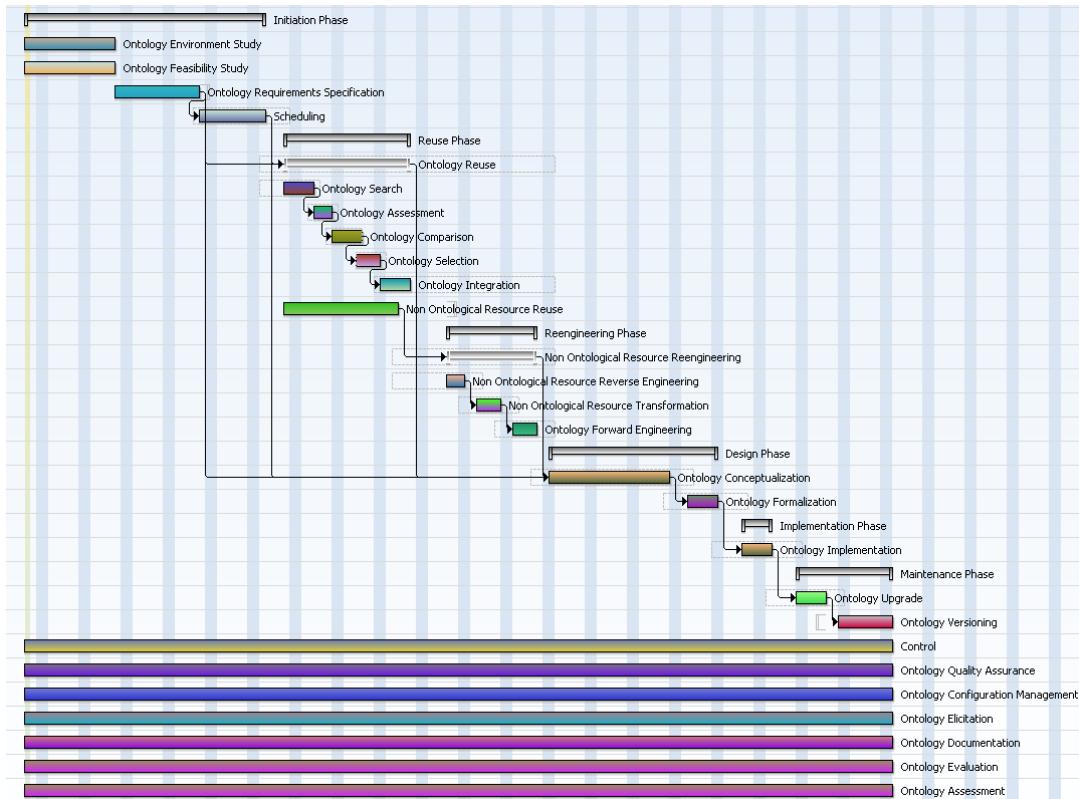


Figure 46. A specific plan generated by gOntt

9.5. Guidelines for Scheduling with gOntt

In this section, we propose a **workflow** for carrying out the scheduling of ontology development projects using gOntt. We include prescriptive methodological guidelines for each of the tasks in the workflow proposed. The tasks for carrying out the scheduling activity can be seen in Figure 47, and are explained in detail below.

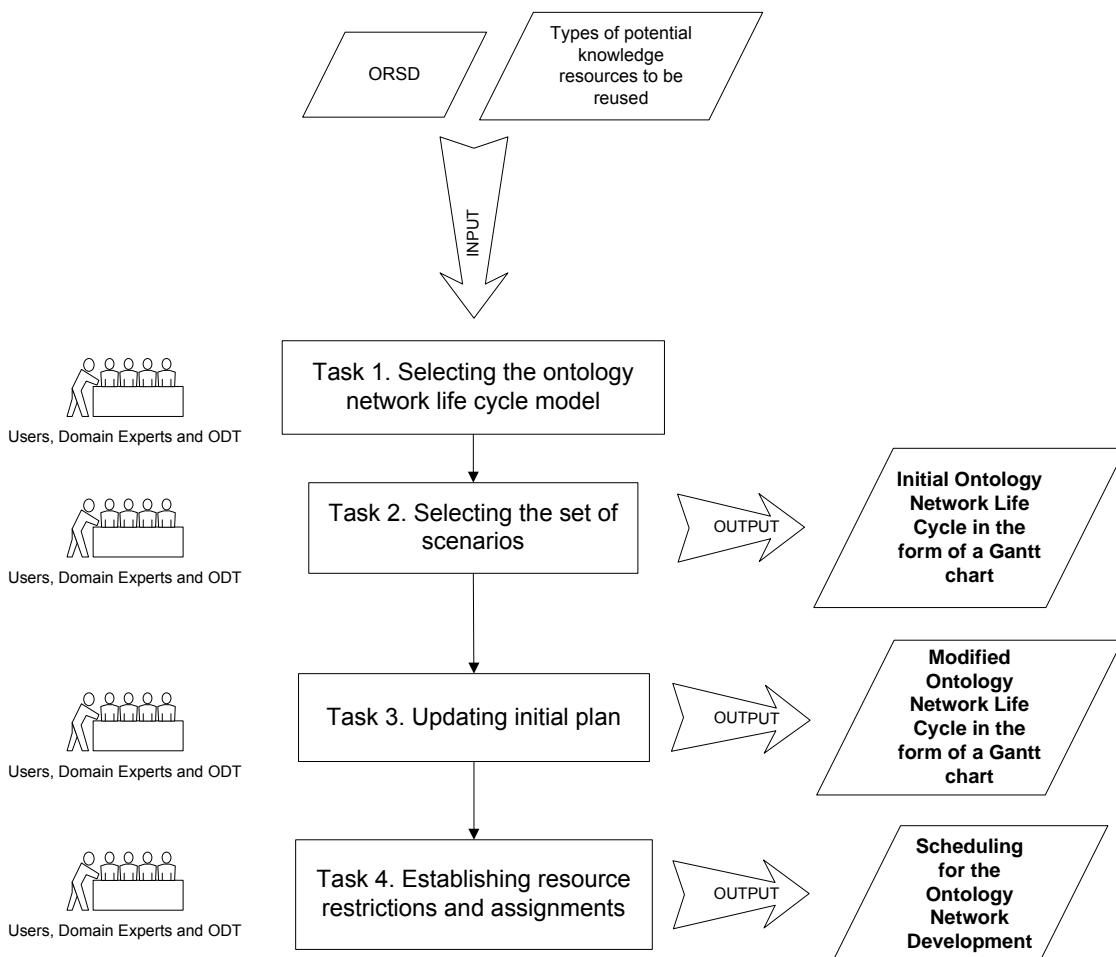


Figure 47. Tasks for scheduling ontology development projects with gOntt

Task 1. Selecting the ontology network life cycle model.

The goal of this task is to obtain the most appropriate ontology network life cycle model for the ontology network to be developed. Users, domain experts and the ontology development team carry out this task taking as input both the ontology requirement specification document (ORSD) and the types of potential knowledge resources to be reused during the development. To help ontology developers to decide which is the most appropriate life cycle model among those presented in Section 7, gOntt presents a simple natural language question, displayed in Figure 48. Based on the response given to the question, the waterfall model or the iterative-incremental is selected. In the case of the iterative-incremental model, ontology developers should also provide the expected number of iterations in the ontology development.

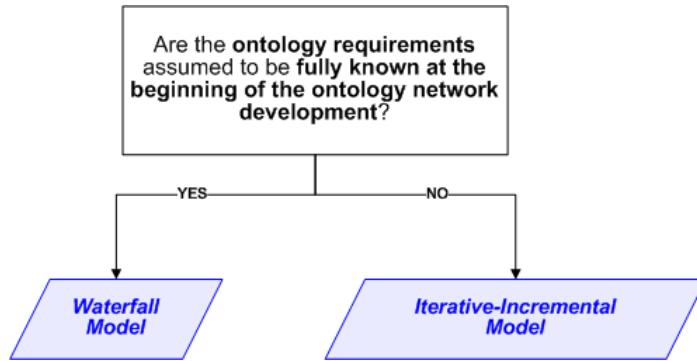


Figure 48. Decision tree for selecting the most appropriate model

Task 2. Selecting the set of scenarios.

The goal of this task is to select the set of scenarios to be followed during the ontology network development. Users, domain experts and the ontology development team carry out this task taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

To help ontology developers in this task, gOntt presents the set of natural language questions displayed on the left hand-side of Figure 39. If the model selected in Task 1 is the waterfall one, then questions should be answered once; on the other hand, if the model selected is the iterative-incremental one, then the set of questions should be answered once per each iteration expected.

With the responses to these questions and the methodological foundations presented in Section 9.3, gOntt is able to obtain the initial ontology network life cycle, that is, an initial plan for the ontology network development. The task output can be represented as a Gantt chart, which is *de facto standard* in software project management. The main idea here was not to create new representations for schedules in ontology engineering but to adapt the existing ones from Software Engineering to this field.

Task 3. Updating initial plan.

The goal of this task is to modify (if necessary) the initial plan presented by gOntt. Users, domain experts and the ontology development team carry out this task taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

Ontology developers can modify the initial plan in the following ways: (a) by including or deleting processes, activities, and model phases, and (b) by changing order and dependencies among processes and activities.

Task 4. Establishing resource restrictions and assignments.

The goal of this task is to include information about temporal scheduling and human resource assignments in the life cycle obtained in Task 3. Users, domain experts and the ontology development team carry out this task taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

Task 4 is out of the scope of this thesis.

9.6. Conclusions

To sum up, we could say that the Ontology Engineering field lacks methods and tools that could guide ontology practitioners when planning and scheduling their ontology development projects. The main contributions of this chapter are

- The definition of methodological basis for scheduling ontology development projects, which are based on the set of scenarios identified in the NeOn Methodology (Chapter 6), on the set of life cycle models (Chapter 7), and on the NeOn Glossary of Processes and Activities (Chapter 5).
- The gOntt plug-in, a tool that supports the scheduling of ontology networks developments.
- The definition of prescriptive methodological guidelines for scheduling ontology development projects using gOntt.

In order to manage properly ontology development projects in complex settings and to apply correctly the NeOn Methodology, it is crucial to have knowledge of the entire ontology development life cycle before starting the developments. The ontology project plan defines the tasks to be done, the time when the tasks will be executed and the dependencies between tasks. The project plan is the only way, as can be shown in other disciplines, to commit people to the project and to show how the work will be performed. It also helps the ontology engineer to monitor its execution and assess the impact of a particular delay in the planned tasks.

In relation to the work presented in this chapter, we plan to integrate the guidelines proposed and gOntt with the the ONTOCOM model [Simperl et al., 2009] for predicting the total costs of the ontology development project. Additionally, we want to extend such works by providing details of the cost associated to carrying out a particular task in an ontology development project.

Finally, we carried out a set of experiments whose aim was to test the preliminary methodological guidelines proposed for establishing the ontology network life cycle. Additionally, we performed an experiment whose goal was to test the use of gOntt both in the scheduling activity and in the development of ontology networks. The main results and conclusions of such experiments are presented in Chapter 11.

10. Ontological Resource Reuse

10.1. Introduction

Ontologies play an important role for many knowledge-intensive applications. The process of building ontologies from scratch as proposed in METHONTOLOGY [Gómez-Pérez et al., 2003] and in On-To-Knowledge [Staab et al., 2001] is time-and-cost consuming. One way of reducing the time and costs associated to the ontology development is by reusing available ontological resources. Ontologies developed by reuse are also expected to spread good practices (from well-developed ontologies) and increase the overall quality of ontological models.

The state of the art presented in Chapter 2 shows that most of the methods studied for carrying out the ontological resource reuse consist of high level steps but do not provide detailed guidelines explaining how to perform each step. Additionally, such methods do not take into account different levels of granularity during the reuse. Thus, one of the open research problem to be solved in this PhD thesis is the following:

- There are no detailed prescriptive guidelines to reuse ontological resources at different levels of granularity.

In this chapter we try to give solution to the abovementioned open research problem by reusing domain and general or common ontologies as a whole and/or by reusing ontology statements. That is, we establish a clear, simple, and useful procedure to reuse ontological resources. As stated in Chapter 3 as a restriction, it is *out of the scope of this thesis to provide guidelines for reusing ontology modules and for reusing and reengineering knowledge resources that are not ontologies*.

The rest of the chapter is structured as follows: Section 10.2 presents an overview of the ontological resource reuse; Section 10.3 provides the methodological guidelines for reusing general or common ontologies; Section 10.4 describes the methodological guidelines for reusing domain ontologies as a whole; Section 10.5 presents the methodological guidelines for reusing ontology statements; and finally Section 10.6 concludes the work presented in the chapter.

10.2. Overview of Ontological Resource Reuse

As mentioned in Chapter 6, the NeOn Methodology presents 9 scenarios for building networks of ontologies. One of these scenarios is *Building Ontology Networks by Reusing Ontological Resources*. In this scenario, ontology developers should analyse whether ontological resources can be reused to build an ontology network or not. The underlying principle is that reusing ontological resources reduces the time and costs associated to the ontology development.

The reuse of ontological resources is encouraged by a recent increase in the number of on-line available ontologies, ontology libraries and repositories⁶⁷.

⁶⁷ See for example a list of novel ontology search engines described at: <http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/SemanticWebSearchEngines>.

The ontological resource reuse process is often influenced by the type of ontology to be reused as presented in Figure 49. There are several types:

- General [van Heijst et al., 1997] or common ontologies [Mizoguchi et al., 1995] provide conceptualizations of generic topics such as time, space, and mereology, and represent knowledge reusable in different domains. They are usually based on well studied theories⁶⁸: mereology, which formalizes parthood relation; topology, which formalizes connection relations; time theories, which formalize terms like *time interval*, *time point*, etc. Given the generality of the topic described, it is common to have several ontologies on the same topic, each of them taking a different standpoint in the conceptual model of the topic. For example, in the case of the topic “*time*” one ontology can model a particular temporal point, and other ontology can model a temporal interval. When reusing one of these ontologies, the ontology engineer needs to be aware of the different views and assumptions the ontology relies on. Guidelines for reusing general or common ontologies are provided in Section 10.3.
- Domain ontologies provide knowledge of a specific domain such as medicine, pharmacy, fisheries, etc. These ontologies can be helpful in cases when a domain ontology is being built in the same domain.

The reuse of large ontologies such as WordNet⁶⁹, the NCI ontology [Golbeck et al., 2003] is rather difficult because they contain a large amount of knowledge not really needed for a particular ontology development. Sometimes, the reuse requires to retrieve bits of knowledge (e.g., modules, statements) and integrate them later on in the new ontology being built rather than to retrieve entire ontologies [d’Aquin et al., 2007b]. For this reason, we distinguish different levels of granularity in the reuse of ontologies, as shown in Figure 49.

- Ontologies can be reused as a *whole* if they closely meet the expectations and the needs of the ontology engineer. Methodological guidelines for reusing domain ontologies as a whole are provided in Section 10.4.
- In certain cases, only one part or *module*⁷⁰ of an ontology is relevant for reuse. For example, when building an ontology about lung cancer, it is not necessary to reuse an entire ontology about the human body, it suffices to reuse a module describing concepts related to the lung. Guidelines for reusing ontology modules are out of the scope of the thesis.
- In other cases, only some knowledge components from the ontology (the description of a particular entity, the branch in the taxonomic hierarchy in which an entity appears, or entity neighborhoods in the ontology) are relevant for the development needs. In these cases, the reuse of ontological knowledge is performed at the *statement* level, providing the ontology developer with better control of the material being reused. Methodological guidelines for reusing ontology statements⁷¹ are provided in Section 10.5.

⁶⁸ A theory is considered here as a system of definitions, axioms and theorems that can be formal, semi-formal or informally represented.

⁶⁹ <http://wordnet.princeton.edu/>

⁷⁰ We consider a module [d’Aquin et al., 2007c] as a part of the ontology that defines the relevant set of terms for a particular purpose.

⁷¹ An ontology statement (or triple) contains the following three components: *subject*, *predicate*, and *object*.

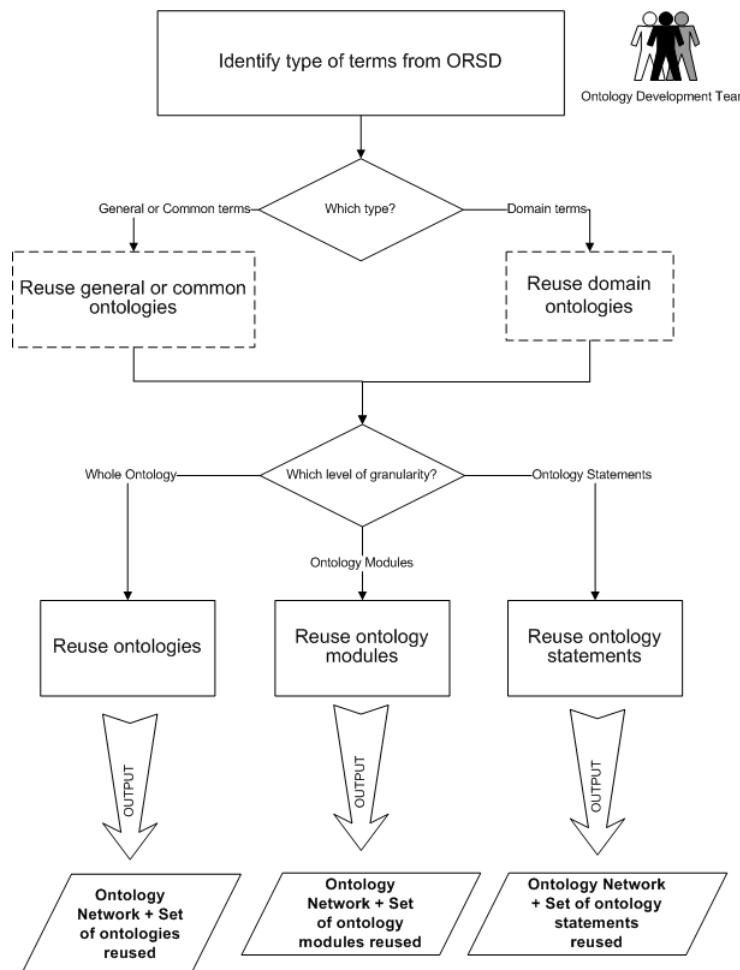


Figure 49. Different types of ontological resource reuse

The rest of the chapter includes some methodological guidelines for reusing ontological resources. It also provides some guidelines for reusing general or common ontologies, domain ontologies and ontology statements.

10.3. Reusing General or Common Ontologies

Common or general ontologies⁷² [Van Heist et al., 1997] are normally based on philosophical theories difficult to understand by engineers. Examples of theories formalizing knowledge are the mereology theory [Varzi, 2007], the topology theory [Varzi, 2007], and the time theory [Hayes, 1995]. Part of this knowledge has even been implemented in languages such as KIF [Genesereth and Fikes, 1992], OWL [Patel-Schneider et al., 2004], and CML [Schreiber et al., 1994b].

One of the first works that describe how to build general ontologies is due to Borst [Borst, 1997], who built and used a mereology ontology in an engineering application for modelling, simulating and designing physical systems. Borst implemented the mereology ontology from scratch in Ontolingua, an ontology based on the work presented in [Simons, 1987].

⁷² Henceforth, we refer to general or common ontologies, just as general ontologies.

General ontologies formalize notions that go beyond particular domains. For example, the *part of* relation links objects in the mechanical domain (the spark plug is part of the motor), and also in the domain of cultural activities (the interpretation of Radetzsky March is part of the New Year Concert). Thus, it seems logical to affirm that such general ontologies have a great potential for reuse.

However, nowadays most ontology developers consider this type of ontologies difficult to understand and, therefore, difficult to use, which implies that they avoid using these formal notions embedded in the general ontologies. As a confirmation of this assertion, we can show the results of a simple experiment we performed. Using Watson on-line we searched for semantic documents that contain the term “part”. Our search yielded a high number of results (4032)⁷³. We then examined a subset of such ontologies (approximately 500) and realized that relations such as hasPart, partOf, etc. are simply defined as object properties with no background knowledge about this kind of relationships. We found in our analysis only one ontology (*bio-zen.owl*⁷⁴) that reuses the general ontology Dolce-Lite. A similar situation occurred when we searched for ontologies containing the term “time”. In this case, we obtained 8803 semantic documents and found that only one of the ontologies analysed (*bio-zen.owl*) reuses the notion of time from a general ontology.

Let's suppose now that we have to develop an ontology about pharmaceutical products in which we directly define an object property as ‘isPartOf’. In this case, to answer the CQ ‘which medicine contains iron?’, a Java program similar to those presented in Figure 50 would be necessary. However, if we are aware of the formal properties (e.g., transitivity, antisymmetry, etc.) of the relationship ‘*part of*’ and we include such formal notions in the ontology (e.g., transitivity), then the aforementioned CQ could be solved with the SPARQL⁷⁵ query shown in Figure 51.

```
// Java program JP1

public static List<Individual> SearchForFeature(OntModel m, Individual initial,
ObjectProperty property, OntClass concept)
{
    List<Individual> openL = new ArrayList();
    List<Individual> wholesL = new ArrayList();
    List<Individual> lIndividuals = new ArrayList();

    Iterator itc = concept.listInstances();

    while(itc.hasNext()) lIndividuals.add((Individual) itc.next());

    openL.add(initial);

    while (!openL.isEmpty())
    {
        Individual q = (Individual) openL.get(0);
        openL.remove(0);
        if (lIndividuals.contains(q)) wholesL.add(q);
        Iterator it = q.listPropertyValues( property );
        while(it.hasNext())
            openL.add(((OntResource) it.next()).asIndividual());
    }
    return wholesL;
}
```

Figure 50. Example of a Java program

⁷³ Access carried out the 27 of April of 2010.

⁷⁴ <http://neuroscientific.net/bio-zen.owl>

⁷⁵ The W3C Recommendation for SPARQL is available in (Prud'hommeaux and Seaborne, 2008).

```

# SPARQL query SQ1

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.NewOnto1.org/killer201004271447#>

SELECT ?X
WHERE
{
    ?X rdf:type ub:drug .
    ub:iron ub:isPartOf ?X .
}

```

Figure 51. Example of a SPARQL query

Thus, we can gain time and effort if we explicitly define ‘what things are’ in the ontology by reusing general ontologies, instead of implementing a Java program that refers to ‘how things are done’. That is, reusing general ontologies in the ontology development (a) prevents from repeating the definition of general notions (e.g., *part of*) and (b) allows taking advantage of the formal characteristics of such notions (properties and definitions).

The main aim of this section is to bridge the gap between the people who work in formal specifications with a solid theoretical grounding and those who work in more technical issues.

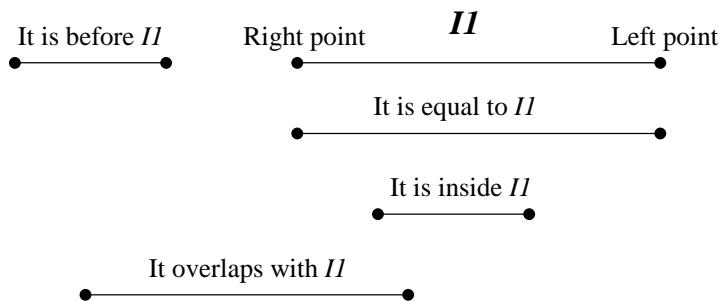
This section is organized as follows: first, we explain with examples two of the theories most relevant in philosophical ontology, in particular, the time theory (Section 10.3.1) and the mereology theory (Section 10.3.2). Then, we propose prescriptive methodological guidelines to reuse general ontologies (Section 10.3.3). Finally, we provide in Annex II an example of how to use the guidelines here proposed for the general ontology reuse process.

10.3.1. Time Modelling

This section presents some notions of time theories in an intuitive manner, which will facilitate the selection of time ontologies to non-experts. These notions are based on [Fernández-López and Gómez-Pérez, 2004].

- *Time points*. As a first intuitive approximation, we can see time points as points in the line time. A study of different ways to define the concept of time point can be found in [Hayes, 1995].
- *Time intervals*. Also as a first intuitive approximation, we can see a time interval as the time between two time points. The abovementioned work also studies different ways to define the concept of time interval.
- *Absolute and relative time*. On the one hand, we can say that time is represented in an absolute way when it is related to a fact⁷⁶. For example, we can associate 1789 with the beginning of the French Revolution. On the other hand, we say that the valid time of a fact is represented in a relative way when it is related to the valid time of another fact (e.g., the Russian Revolution was after the French Revolution).
- *Relations between time intervals*. The most well-known relations between intervals are the ones identified by Allen [Allen, 1984]. Some of them are shown in Figure 52.

⁷⁶ <http://www.cs.auc.dk/~csj/Glossary/>

**Figure 52. Relations between intervals**

- **Convex and non convex intervals.** These notions allow identifying periodic intervals with gaps between them (e.g., “every Wednesday”), which are called non convex intervals. Conversely, convex intervals are those that are not composed of “separate pieces” (e.g., from 1st April 2004 to 30th April).
- **Open and closed intervals.** Sometimes, the interval end points might be or not included in the interval. For example, [1985, 1986) is an interval left closed and right open.
- **Proper intervals.** An interval whose extremes are different is called proper (see DAML ontology documentation^{77 78}). Thus, for example, [1985, 1986] is a proper interval, however, [1985, 1985] is not.
- **Temporal granularities.** For example, the duration of an event in days can be transformed into minutes. This operation is usually easy to carry out. However, there are cases where the transformation can be more difficult. Let’s think, for instance, in the transformation of an interval between two different natural days into working days [Bettini et al., 1996].
- **Total ordering.** Total ordering means that, for every pair of temporal points t_1 and t_2 , necessarily $t_1 < t_2$ or $t_2 < t_1$. It is useful to make this assumption in many applications. However, and in distributed systems mainly, this assumption may not be so appropriate. For example the time point t_1 when the process P1 is started in the machine M1 and the point t_2 when the process P2 is started in the machine M2 may not be linked through the relations $t_1 < t_2$ nor through $t_2 < t_1$.
- **Infinity.** An infinite interval is that which is not limited in the past or in the future. There are two common ways of allowing infinitely long intervals (see footnote 77). In the first approach (see Figure 53), an infinite interval in the past and in the future is modelled through a point in the negative infinite and another one in the positive infinite. In the second approach, an infinite interval in the past and in the future is modelled through one interval with no beginning and one interval with no ending. Given that the second approach does not allow considering points in the infinite, both approaches are incompatible with the same ontology. This is the reason why some ontologies leave this part of the modelling open.
- **Density,** which is used to represent that between any two distinct points there is a third distinct point. If we assume that between the second s and the second $s + 1$ there is no another second, and the time is viewed as an ordered set of seconds, then density cannot be assumed. The same happens if time is represented as an ordered set of minutes, or hours, etc. However, we have the option of assuming that a second establishes the interval where a time point is. According to this approach, if we say that point P is in second s , we are specifying the interval of time where P is. In this case, density can be assumed.

⁷⁷ <http://www.cs.rochester.edu/~ferguson/daml/>⁷⁸ <http://vacuumcleaner.cs.kun.nl/c-corn/documentation/CoRN.ftc.MoreIntervals.html>

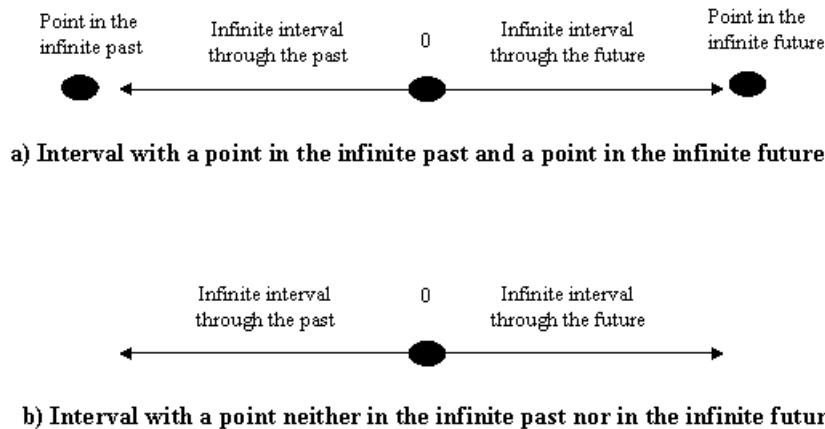


Figure 53. The two usual representations of infinite temporal intervals (see footnote 77)

- *Isomorphism to the real numbers.* The set of real numbers is very often the model of the time theory. We can observe that if we take this model in an exact way, we assume the properties of density, convexity, total ordering; we also assume that there are no points in the infinite.

In summary, we provide in Table 13 all the notions aforementioned.

Notions	
<i>Time points</i>	<i>Proper intervals</i>
<i>Time intervals</i>	<i>Temporal granularities</i>
<i>Absolute and relative time</i>	<i>Total ordering</i>
<i>Relations between time intervals</i>	<i>Infinity</i>
<i>Convex and non convex intervals</i>	<i>Density</i>
<i>Open and closed intervals</i>	<i>Isomorphism to the real numbers</i>

Table 13. Summary of notions in the time modelling

10.3.2. Mereology Modelling

A *mereology* is a formal theory of parts and associated concepts [Borst, 1997; Schneider, 2004]. 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 [Varzi, 2007].

Theory *M*

Most of the authors agree on the following core of axioms (named with A) and definitions (named with D) [Varzi, 2007]. Along these paragraphs we use the examples of territories to clarify the meaning of axioms and definitions. The mentions to administrative units really refer to their physical territories.

- 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 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:

- 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 ⁷⁹. 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 physically located in Africa (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 the same 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 the embodiment of the common core of any mereological theory. A.1-A.3 should be extended to build a mereology. Some of the most relevant extensions are the supplementation principles, the sum and the product principles, and the atomistic assumption. The combination of extensions leads to the following mereologies: minimal, extensional, closure, closure extensional, general, general extensional and atomistic.

Minimal Mereology (MM)

A way to extend M is assuming the following decomposition principle [Varzi, 2007]:

- 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, fulfils 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 A.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.

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 . The domain of territories also fulfils the strong supplementation principle. 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. The reason why this principle is named *strong* is that A.5 implies A.4.

This theory is called ‘extensional’ because a theorem (T) 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.

⁷⁹ <http://hcs.science.uva.nl/projects/NewKACTUS/library/lib/mereology.html>

Closure Mereology (CM)

Another way of extending M is by composition [Varzi, 2003]:

- 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. Although Spain contains both Madrid and Barcelona, it is not the sum, since it is not the smallest object that fulfils such a property.
- 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.

The assumption of A.6 and A.7 is controversial. That is, not every researcher accepts that the inclusion of these axioms leads to a practical or theoretical benefit. In fact, it is not obvious that the overlap of Spain and Africa makes an entity.

Closure Extensional Mereology (CEM)

The action of adding the axioms A.6 and A.7 to MM or to EM yields as a result Minimal Closure Mereology (CMM) or Extensional Closure Mereologies (CEM), respectively. In the presence of A.4, A.7 implies A.5. Consequently, CMM and CEM are the same theory [Varzi, 2003].

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 of all xs 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 [Varzi, 2003], is the theory of General Extensional Mereology, or GEM , since A.8 implies A.7 and A.7 + A.4 imply A.5 [Simons, 1987]. 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 divided in its turn.

Figure 54 shows a diagram with all the theories [Varzi, 2003] presented in this section until now.

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

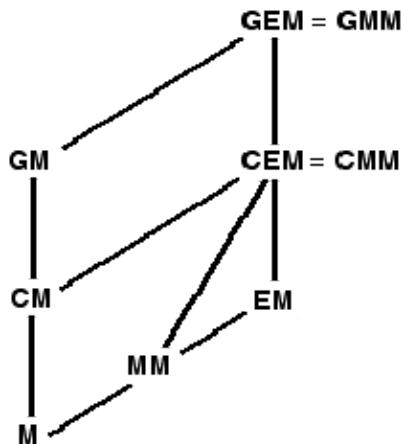


Figure 54. Hasse diagram of mereological theories (from weaker to stronger, going uphill) [Varzi, 2003]

In summary, we provide in Table 14 all the axioms and definitions aforementioned.

Axioms	Definitions
A.1) Reflexivity	D.1) Proper part
A.2) Antisymmetry	D.2) Direct part
A.3) Transitivity	D.3) Overlap
A.4) Weak supplementation principle	D.4) Underlap
A.5) Strong supplementation	D.5) Disjoint
A.6) Sum principle	D.6) Binary sum
A.7) Product principle	D.7) Binary product
A.8) Unrestricted fusion principle	D.8) General sum
A.9) Atomicity	D.9) Atom

Table 14. Summary of the axioms and definitions in the mereology modelling

10.3.3. Methodological Guidelines for Reusing General Ontologies

The goal of reusing general ontologies is to find and select general or common ontologies and integrate them in the ontology network being developed.

In the framework of the NeOn Methodology for building ontology networks, we propose the **Filling Card** presented in Table 15 for the reuse of general ontologies. The card includes the definition, the goal, the inputs and outputs, the performer of the process and the time scheduled for the process.

General Ontology Reuse	
<i>Definition</i>	
<p><i>General Ontology Reuse</i> refers to the process of using general ontologies in the solution of different problems.</p>	
<i>Goal</i>	
<p>The goal of this process is to find and select general ontologies and integrate them in the ontology network being developed.</p>	
<i>Input</i>	<i>Output</i>
<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 that compare with the same criteria the candidate ontologies to be reused.</p>	<p>A general ontology integrated in the ontology network being developed.</p>
<i>Who</i>	
<p>Software developers and ontology practitioners involved in the ontology development. It may be required the help of an ontology practitioner familiarized in formal ontologies or theories.</p>	
<i>When</i>	
<p>The general ontology reuse process should be carried out after the ontology specification activity.</p>	

Table 15. General Ontology Reuse Filling Card

Figure 55 shows the **workflow** and the activities for carrying out the general ontology reuse process.

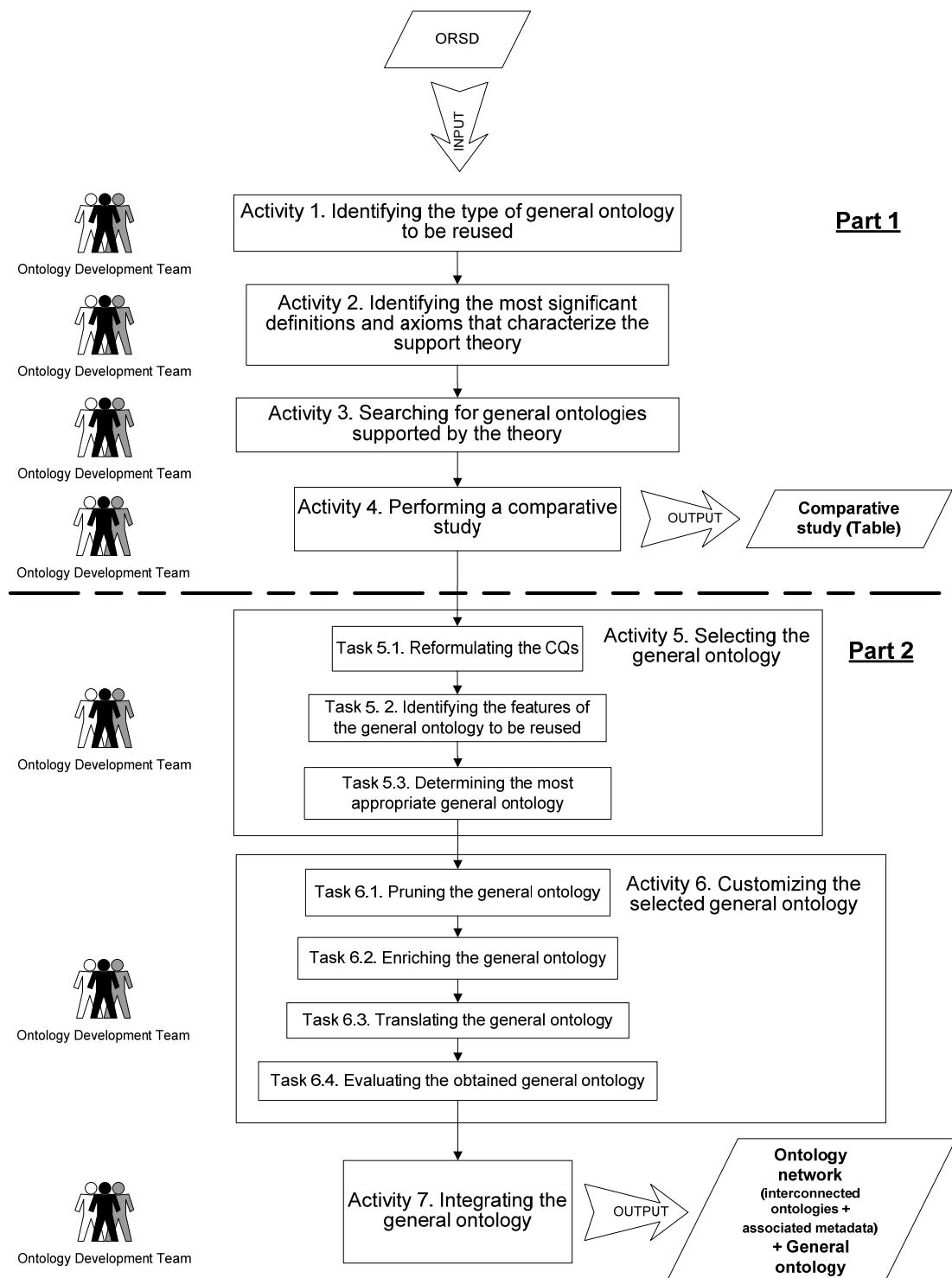


Figure 55. Activities for Reusing General Ontologies

The workflow in Figure 55 is divided in two main parts:

- *Part 1.* Activities 1 to 5 have to be carried out to obtain a comparative study of the theory that supports the type of general ontology to be reused (e.g., a study on time modelling or a study on part-whole modelling). The purpose of this part is to compare the general ontologies that describe a particular theory (e.g., time, mereology) with the features of such a theory. Our proposal is to carry out such a comparative study in the form of a table and then to use this table to select the general ontology that best fits with the requirements of the ontology to be developed.
- *Part 2.* Activities 6 to 8 have to be carried out to select, customize, and integrate in the ontology to be developed the most appropriate general ontology.

It is worth mentioning that if there is a previous comparative study of the theory that supports the type of general ontology to be reused, then activities 1 to 5 do not have to be carried out, and thus the reuse process should start directly in Part 2.

The activities shown in Figure 55 are explained in detail next. Thus, we include the NeOn methodological guidelines for reusing general ontologies. For the sake of clarity, we explain the different activities involved in the whole process and consider the reuse of just one general ontology. To reuse more than one general ontology, the process described should be performed iteratively.

Activity 1. Identifying the type of general ontology to be reused.

The goal of this activity is to determine whether general ontologies should be reused, and if the answer is affirmative, what type of general ontology should be reused. Table 16 shows some heuristics to help to decide the type of general ontology to be reused. Such heuristics are based on the analysis of competency questions (CQs) of the ontology being built. Table 17 shows some example of these heuristics.

Condition on the CQs of the ontology	Typical cases	Type of general ontology to reuse
A reference to time appears.	<p>The word <i>when</i> appears.</p> <p>Some of the following adverbs appear: <i>after</i>, <i>before</i>, <i>at the same time</i>, <i>simultaneously</i>, etc.</p> <p>Some of the following expressions appear: <i>how many times</i>, <i>how much time</i>, <i>how often</i>, etc.</p> <p>Some of the following nouns appear: <i>date</i>, <i>hour</i>, <i>minute</i>, <i>second</i>, etc.</p>	<i>Time</i>
<p>The conjunction of the following conditions is satisfied:</p> <p>CM₁) the CQ refers to a relation R that establishes an order; and</p> <p>CM₂) R fulfils the weak supplementation principle.</p>	<p>Spatial relations</p> <p>Relations between activities</p> <p>Relations between mechanical devices and their pieces</p>	<i>Mereology</i>
CM ₃) The CQ refers to a relation S that is subrelation of an R that satisfies conditions CM ₁ and CM ₂ .	Some relations in the aforementioned domains (e.g., <i>is capital of</i> , <i>is the conductor of</i> , <i>is the main component of</i> , etc.).	

Table 16. Heuristics to determine what type of general ontologies is needed

Examples of the condition	Typical cases and conditions	Type of general ontology to reuse
When is the deadline of activity a?	The word <i>when</i> appears.	Time
Is activity a_1 before activity a_2 ?	The adverb <i>before</i> appears.	
How often must driver d rest?	The expression <i>how often</i> appears.	
How many hours does activity a last?	The expression <i>how many</i> and the noun <i>date</i> appear.	
Which places can be visited in region r?	<p><i>Satisfaction of CM₁:</i> the place p_1 can belong to the place p_2, p_2 to p_3, etc., in such a way that the order $p_1 < p_2 < p_3$ could be established, where '$<$' means <i>belongs to</i>.</p> <p><i>Satisfaction of CM₂:</i> if $x \neq y$ and x belongs to y, then y necessarily has another place z such that z does not overlap x. For example, given that Madrid belongs to Spain and Madrid is not equal to Spain, then Spain has other places.</p>	
Which were the interpretations in concert c?	<p><i>Satisfaction of CM₁:</i> the interpretation of a music piece p can belong to the interpretation of a music work w, w can <i>belong</i> to concert c, etc, in such a way that the order $p < w < c$ could be established, where '$<$' means <i>belongs to</i>.</p> <p><i>Satisfaction of CM₂:</i> for example, given that the interpretation of Chopin's Sonata opus 35 belongs to Pollini's concert in Paris the 13th of October of 2009, and the interpretation of the Sonata opus 35 is not equal to such a concert, then it had other interpretations.</p>	Mereology
Which are the pieces of the car c?	<p><i>Satisfaction of CM₁:</i> a spark plug p can belong to an engine e, e to car c, etc, in such a way that the order $p < e < c$ could be established, where '$<$' means <i>belongs to</i>.</p> <p><i>Satisfaction of CM₂:</i> for example, given that the 602 cc H2 air cooled engine belongs to Citroën 2CV, and such an engine is not equal to the 2CV, then the 2CV has other components.</p>	
Which is the capital of country c?	<i>Satisfaction of CM₃:</i> <i>is capital of</i> is subrelation of <i>is part of</i> .	

Table 17. Examples of the heuristics to determine what type of general ontologies is needed

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

The ontology development team should study the theory (and its variants) that supports the type of general ontology to be reused. The goal of this activity is to identify the definitions and axioms that characterize the support theory.

In this thesis we provide some guidelines for the time and the mereology theories (Sections 10.3.1 and 10.3.2, respectively). For other general ontologies (e.g., topomereology), the literature on the topic should be analysed to obtain a description similar to those provided in Sections 10.3.1 and 10.3.2. In this last case, the activity may require the help of someone familiarized with general theories.

It is important to mention that this activity can be time and resource consuming, but the result can be exploited in further projects.

Activity 3. Searching for general ontologies supported by the theory.

The ontology development team should search for ontologies that support partially or completely the theory in ontology libraries, repositories and registries. Those ontologies should be implemented in a formal language.

The activity output is a set of candidate general ontologies that could be implemented in different languages, for example, KIF [Genesereth and Fikes, 1992], OWL [Patel-Schneider et al., 2004], RDF(S) [Brickley and Guha, 2004], CommonKADS Modelling Language (CML) [Schreiber et al., 1994b], etc.

As an example, we present here a representative set of time ontologies (not necessarily an exhaustive set) [Fernández-López and Gómez-Pérez, 2004]:

- ❑ The time ontology in Upper Cyc Ontology⁸⁰, which is included in the Cyc knowledge base [Lenat and Guha, 1990], and is implemented in KIF and XML.
- ❑ The Unrestricted Time ontology⁸¹

- The AKT Time ontology, developed within the AKT initiative and implemented in OCML [Motta, 1999].

Activity 4. Performing a comparative study.

The goal here is to compare the candidate general ontologies obtained in Activity 3 according to the features of the support theory identified in Activity 2. We propose to represent this comparative study in the form of a table to facilitate its use. In the table, 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 studies what features each ontology has. If we analyse a particular feature in a particular ontology, the results can be the following:

- Yes: if the feature is represented in the ontology.
- No: if the feature is not represented in the ontology.
- *Inferred*: if the feature is not directly represented in the ontology, but can be inferred.

As an example, we present here a comparative table of ontologies implementing time theories (Table 18). The table is based on the work described in [Fernández-López and Gómez-Pérez, 2004].

Activity 5. Selecting the general ontology to be reused.

The goal of this activity is to select the most appropriate general ontology to be reused in the ontology network being developed. This activity takes as input the comparative table obtained in Activity 4.

This activity is divided in the following tasks:

Task 5.1. Reformulating the CQs.

The main goal of this task is to reformulate the CQs included in the ORSD of the ontology network that is being developed with the vocabulary of the support theory included in the general ontology. Additionally, another goal of this task is to identify axioms that link CQ terms to general ontologies terms.

Table 19 proposes a series of heuristics (a) to perform the transformation of the CQs and (b) to identify the linking axioms. Additionally, Table 20 shows some examples of transformations using the series of heuristics presented in Table 19.

⁸⁵ <http://www.cs.rochester.edu/~ferguson/daml/>

Features	<i>Time in Upper Cyc</i>	<i>Unrestricted Time</i>	<i>Simple Time</i>	<i>Reusable Time</i>	<i>Kestrel Time Ontology</i>	<i>SRI Time Ontology</i>	<i>Time in SUMO</i>	<i>DAML time ontology</i>	<i>AKT time ontology</i>
Time points	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Time intervals	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Ordering relations: overlaps, etc.	Yes	Yes	Yes	Yes	Yes	No (<i>before only</i>)	Yes	Yes	Yes
Modelling of years, months, days, hours and minutes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Time zones	No	No	No	No	No	No	Yes	Yes	No
Granularities	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Linkage of other types of concept with concepts of time	Yes	Yes	No	No	No	No	Yes	Yes	No
Axioms	No	Yes	Yes	Yes	No	No	Yes	Yes	No

Table 18. Features of the ontologies that implement time theories (based on the work described in [Fernández-López and Gómez-Pérez, 2004])

Condition on the CQs of the ontology	Case	Action to be carried out
TIME		
It can be expressed as: <i>in which moment does X happen?</i>	CT ₁) Time points	To introduce a linking axiom to time points
It can be expressed as: <i>in which interval of time does X happen?</i>	CT ₂) Time intervals	To introduce a linking axiom to time intervals
MEROLOGY		
CM ₁ and CM ₂ are fulfilled	CsM ₁) We are interested in knowing if two objects have common parts	To reformulate the CQ to mention the term <i>overlap</i>
CM ₁ and CM ₂ are fulfilled	CsM ₂) We are interested in knowing some feature of the largest common part of two objects	To reformulate the CQ to mention the term <i>binary product</i>
CM ₁ and CM ₂ are fulfilled	CsM ₃) We are interested in knowing if two objects have common wholes	To reformulate the CQ to mention the term <i>underlap</i>
CM ₁ and CM ₂ are fulfilled	CsM ₄) We are interested in knowing some feature of the smallest common whole of two objects	To reformulate the CQ to mention the term <i>binary sum</i>
CM ₁ and CM ₂ are fulfilled	CsM ₅) We are interested in knowing if the collection of objects that fulfil a property makes up a common whole with certain features	To reformulate the CQ to mention the term <i>general sum</i>
CM ₁ and CM ₂ are fulfilled	CsM ₆) We are interested in knowing the parts of an object excluding the object itself	To reformulate the CQ to mention the term <i>proper part</i>
CM ₁ and CM ₂ are fulfilled	CsM ₇) We are interested in knowing the direct parts of an object	To reformulate the CQ to mention the term <i>direct part</i>
CM ₁ and CM ₂ are fulfilled	CsM ₈) We are interested in knowing which parts of object o ₁ are not in object o ₂	To reformulate the CQ to mention the term <i>disjoint</i>
CM3 is fulfilled	CsM ₉) Ever recommended	To introduce a linking axiom establishing that S is sub-relation of <i>is part of</i>

Table 19. Heuristics to transform CQs

<i>Original CQ</i>	<i>Case</i>	<i>Action to be carried out</i>
TIME		
<i>When does concert c begin?</i>	$CT_1)$	To introduce the following axiom: “The range of <i>begins</i> in the concept <i>concert</i> is <i>time point</i> ”
<i>When is concert series s celebrated?</i>	$CT_2)$	To introduce the following axiom: “The range of <i>is celebrated</i> in the concept <i>concert series</i> is <i>time interval</i> ”
MEROLOGY		
<i>Do the Nordic Countries and EU have parts in common?</i>	$CsM_1)$	To rewrite the CQ in this way: “ <i>Do Nordic Countries and the EU overlap?</i> ”
<i>Do the Nordic Countries that make up the European Union make up a common organization?</i>	$CsM_2)$	To rewrite the CQ in this way: “ <i>Is the binary product of Nordic Countries and the European Union a common organization?</i> ”
<i>Do Belgium and Luxemburg belong to common organizations?</i>	$CsM_3)$	To rewrite the CQ in this way: “ <i>Do Belgium and Luxemburg underlap?</i> ”
<i>Do Belgium and Luxemburg make up a common political organization?</i>	$CsM_4)$	To rewrite the CQ in this way: “ <i>Is the binary sum of Belgium and Luxemburg a political organization?</i> ”
<i>Do countries with a per capita income greater than 30000 \$ form an economical organization?</i>	$CsM_5)$	To rewrite the CQ in this way: “ <i>Is the general sum of countries with a per capita income greater than 30000 \$ an economical organization?</i> ”
<i>Which are the countries, provinces, cities, towns, etc. of Benelux?</i>	$CsM_6)$	Given that we are not interested in the answer <i>Benelux</i> , we should rewrite the CQ in this way: “ <i>Which are the proper parts of Benelux?</i> ”
<i>Which are the countries of Benelux?</i>	$CsM_7)$	Given that we are not interested in provinces, cities, towns, etc., we should rewrite the CQ in this way: “ <i>Which are the direct parts of Benelux?</i> ”
<i>Does Avelina Vidal perform in Madrid?</i>	$CsM_8)$	To rewrite the CQ in this way: “ <i>Does Avelina Vidal perform in some part of Madrid?</i> ” For example, if Avelina Vidal performs in the National Auditorium, then she performs in Madrid. Let's note that it is possible that the system only has the information ‘Avelina Vidal performs in Madrid’, but not the information of the particular place. In such a case, Madrid as a part of itself has to be considered to answer the CQ. Let's also note that the system applies the composition of the relations <i>performs</i> and <i>is part of</i> when it answers the CQ.
<i>Which is the capital of country C?</i>	$CsM_9)$	To introduce the following linking axiom: “ <i>Is capital of is sub-relation of is part of</i> ”

Table 20. Example of CQs transformations

Task 5.2. Identifying the features of the general ontology to be reused.

The goal here is to identify which typical definitions, axioms and principles of the support theory are needed in the ontology network to be developed.

To facilitate the execution of the next task, that is, Task 5.3, when the features required have been identified, the ontology developer should mark them in the table containing the results of the comparative study of Task 4.

Table 21 and Table 23 offer a series of heuristics to take a decision in this task. Table 22 and Table 24 offer examples of how to use the abovementioned heuristics.

Features	When they are useful
F_1) Modelling of time points	When questions that can be expressed as, <i>In which moment does X happen?</i> are to be answered.
F_2) Modelling of time intervals	When questions that can be expressed as, <i>In which interval of time does X happen?</i> are to be answered.
F_3) Modelling of absolute time (by means of time units)	When questions that can be expressed as, <i>When does X happen?</i> are to be answered.
F_4) Modelling of relative time	When questions that can be expressed as, <i>Does X happen before Y? Does X happen after Y? Does X happen at the same time as Y?</i> , etc. are to be answered.
F_5) Modelling of relations between time entities	When we want to know about time intervals without "separate" pieces.
F_6) Modelling of convex intervals	
F_7) Modelling of non convex intervals	When we want to reason with time intervals with "separate" pieces (e.g., every Wednesday, every 8 hours, every laboral day, etc.).
F_8) Modelling of open intervals	When we want to know about time intervals expressed (explicit or implicitly) in the form <i>from X to Y (X and/or Y not included)</i> .
F_9) Modelling of closed intervals	When we want to know about time intervals expressed (explicit or implicitly) in the form <i>from X to Y (both included)</i> .
F_{10}) Explicit modelling of proper intervals	Not useful at present, although it may be applied in the future.
F_{11}) Modelling of different temporal granularities	When we are interested in dealing with different time measure units.
F_{12}) Modelling of different time zones	When we want to know about different time zones of the world.
F_{13}) Modelling of total ordering	When we are not interested in different future possibilities. Until now, all our ontologies assume total ordering.
F_{14}) Modelling of infinity	When the time interval considered has at least one of its ends with no limit. Useful in formal models of Physics, Biology, Economics, etc.
F_{15}) Density	When continuous time is assumed. Useful in Physics, Biology, Economics, etc.
F_{16}) Isomorphism to the real numbers	When total ordering, infinity and density are assumed.

Table 21. Heuristics to determine which time features are useful

Features	Examples of use
$F_1)$	<p>FACT: Fact 1) Avelina Vidal's concert at Madrid National Auditorium begins the 8th of May of 2010 at 19:30.</p> <p>COMPETENCY QUESTION: CQ) When does Avelina Vidal's concert at Madrid National Auditorium begin?</p> <p>HOW TO ANSWER IT: Directly from fact 1.</p>
$F_2)$	<p>FACT: Fact 1) The concert series Ciclo Excelentia is celebrated in the time interval with left point equal to 14th of April of 2010 and right point equal to 29th of December of 2010.</p> <p>COMPETENCY QUESTION: CQ) When is Ciclo Excelentia celebrated?</p> <p>HOW TO ANSWER IT: Directly from fact 2.</p>
$F_3)$	<p>See example of modelling of time points (F_1).</p>
$F_4, F_5)$	<p>FACT: Facts of <i>time point</i> and <i>time interval</i> examples.</p> <p>COMPETENCY QUESTION: CQ) Is Avelina Vidal's concert at Madrid National Auditorium celebrated before Ciclo Excelentia?</p> <p>HOW TO ANSWER IT: Given that Avelina Vidal's concert begins in a date after the left point of the time interval of the celebration of Ciclo Excelentia, the answer is <i>no</i>.</p>
$F_6)$	<p>See example of modelling of time intervals (F_2).</p>
$F_7)$	<p>FACT: Fact 1) The Prado Museum is closed every Monday.</p> <p>COMPETENCY QUESTION: CQ) <i>Will the Prado Museum be closed the 5th of May of 2010?</i></p> <p>HOW TO ANSWER IT: If the system reasons with non convex intervals, it can calculate the intersection of the interval <i>every Monday</i> with the 5th of May of 2010. Given that such an intersection is empty (the 5th of May of 2010 is Wednesday), the answer is <i>no</i>.</p>
$F_8)$	<p>FACTS:</p> <p>Fact 1) The opera "Salome" is scheduled in the Royal Theatre of Madrid the following days of April of 2010: 11th, 13th, 14th, 16th, 17th, 19th, 20th, 22nd, 23rd, 25th, 26th, 28th.</p> <p>Fact 2) The opera "Il Puritani" is scheduled for the 29th of April of 2010 in the Royal Theatre of Madrid.</p> <p>Fact 3) The only opera performances scheduled for April of 2010 in the Royal Theatre of Madrid are the ones shown in facts 1 and 2.</p> <p>COMPETENCY QUESTION: CQ) Are there opera performances scheduled on April of 2010 before the 11th in the Royal Theater of Madrid?</p> <p>HOW TO ANSWER IT: The system can calculate the union of the scheduling of both operas ("Salome" and "Il Puritani"). Then, it calculates the intersection of the result of the former step with the interval [the 1st of April of 2010, the 11th of April of 2010), which is open by the right point.</p> <p>Let's note that, given that there are gaps between the different performances, the interval of the scheduling of "Salome" is non convex.</p>

Table 22. Examples of the heuristics to determine which time features are useful

Features	Examples of use
$F_9)$	See example of modelling of time intervals.
$F_{11})$	<p>COMPETENCY QUESTION: CQ) How many working days will the Prado Museum be open during the month of April of 2010?</p> <p>HOW TO ANSWER IT: Let's suppose that our system manages different temporal granularities and that it can transform months into natural days, and natural days into working days. First, it infers the natural days of the month of April (except Mondays, when the Prado Museum is closed). Then it transforms the interval of natural days into the non convex interval of working days.</p>
$F_{12})$	<p>FACT: Fact 1) The New Year Concert was celebrated the 1st of January of 2010 at 11:15 AM (GMT+1).</p> <p>COMPETENCY QUESTION: CQ) When was the New Year Concert celebrated? (we want the answer in GMT)</p> <p>HOW TO ANSWER IT: The question can be answered if the system can transform time zones.</p>
$F_{14})$	<p>ASSUMPTION: We have represented a calendar.</p> <p>COMPETENCY QUESTION: CQ) What day of the week will be the 27th of April of 2040?</p> <p>REMARK: Let's note we should not have limits to pose this question.</p>
$F_{15})$	<p>FACT: Fact 1) A body has covered a distance of 20 meters at a speed of 13 meters per second.</p> <p>COMPETENCY QUESTION: How much time has the body spent to cover the 20 meters?</p> <p>HOW TO ANSWER IT: Dividing 20 between 13, the result is 1,538461.</p> <p>In this example, we are assuming density.</p>
$F_{16})$	See examples for ordering (F_{13}), infinity (F_{14}) and density (F_{15}).

Table 22. Examples of the heuristics to determine which time features are useful

Axioms and definitions	When they are useful
Theory M	
A.1) Reflexivity	Recommended if its implementation is possible, to ensure the right meaning of <i>part of</i> .
A.2) Antisymmetry	Recommended if its implementation is possible, for consistency verification.
A.3) Transitivity	When X has parts X_1, X_2, \dots, X_n . In its turn, there is some X_i with parts $X_{i1}, X_{i2}, \dots, X_{im}$. That is, X has several levels of parts. Besides, we are interested in all the levels when we ask, <i>Which are the parts of X?</i>
D.1) Proper part	When we are interested in knowing the parts of an object excluding the object itself.
D.2) Direct part	When we are interested in knowing the direct parts of an object.
D.3) Overlap	When we are interested in knowing if two objects have common parts.
D.4) Underlap	When we are interested in knowing if two objects have common wholes.
D.5) Disjoint	When we are interested in knowing which parts of object o1 are not in object o2.
Minimal Mereology (MM)	
A.4) Weak supplementation	Recommended if its implementation is possible, for consistency verification.
Extensional Mereology (EM)	
A.5) Strong supplementation	When we are interested in inferring if two objects are identical. Note that (A.5) is assumed in both (D.6) and (D.7); that is, (A5) is implemented if (D.6) or (D.7) are implemented.
Closure Mereology (CM)	
A.6) Sum principle	Note that (A.6) is assumed in the binary sum definition (D.6), although its explicit implementation is not necessary.
A.7) Product principle	Note that (A.7) is assumed in the binary product definition (D.7), although its explicit implementation is not necessary.

Table 23. Heuristics to determine which mereology features are useful

Axioms and definitions	When they are useful
<i>Closure Extensional Mereology (CM)</i>	
D.6) Binary sum	When we are interested in knowing some feature of the smallest common whole of two objects.
D.7) Binary product.	When we are interested in knowing some feature of the largest common part of two objects.
<i>General (classical) Mereology (GM)</i>	
A.8) Unrestricted fusion principle	Note that (A.8) is assumed in the general sum definition (D.8), although its explicit implementation is not necessary.
D.8) General sum	When we are interested in knowing if the collection of objects that fulfils a property forms a common whole with certain features.
<i>Atomistic Mereology</i>	
D.9) Atom	Note that the weak supplementation principle should be represented in all the cases (if possible).
A.9) Atomicity	When the following conditions are fulfilled: a) Each object has simple parts that are not divided and b) We are interesting in knowing lacks of information.

Table 23. Heuristics to determine which mereology features are useful

Axioms and definitions	Examples of use
Theory M	
A.1)	<p><i>RULE:</i> Rule 1) If P_x is connected to P_y through a flight, and P_x is part of X, and P_y is part of Y, then X is connected through a flight to Y.</p> <p><i>FACTS:</i></p> <p>Fact 1) Madrid is connected to Amsterdam by plane.</p> <p>Fact 2) Amsterdam is part of Holland.</p> <p><i>COMPETENCY QUESTION:</i> CQ) Is Madrid connected to Holland by plane?</p> <p><i>HOW TO ANSWER IT:</i> Applying the reflexivity of <i>is part of</i>, we can deduce that Madrid is part of Madrid (thereafter fact 3.A.1). Facts 1, 3.A.1 and 2 allow triggering rule 1 and answering yes to CQ.</p>
A.2)	<p><i>RIGHT FACTS:</i></p> <p>Fact 1) Holland is part of the Netherlands.</p> <p>Fact 2) Holland is different from the Netherlands.</p> <p><i>WRONG FACT:</i> Fact 3) The Netherlands is part of Holland.</p> <p><i>HOW TO DETECT THE MISTAKE:</i> The conjunction of facts 1 and 3 fulfils the antecedent of antisymmetry axiom. Therefore, the system infers that the Netherlands are identical to Holland, which is a contradiction with fact 2.</p>
A.3)	<p><i>FACTS:</i></p> <p>Fact 1) Amsterdam is part of North Holland.</p> <p>Fact 2) North Holland is part of the Netherlands.</p> <p>Fact 3) The Netherlands is a sovereign state.</p> <p><i>COMPETENCY QUESTION:</i> CQ) Which sovereign state is Amsterdam part of?</p> <p><i>HOW TO ANSWER IT:</i> The conjunction of facts 1 and 2 fulfils the antecedent of the transitivity axiom. Therefore, Amsterdam is part of the Netherlands. Given that fact 3 establishes that the Netherlands is a sovereign state, the answer to CQ is the Netherlands.</p>
D.1)	<p><i>COMPETENCY QUESTION:</i> CQ) Which are the proper parts of Benelux?</p> <p><i>HOW TO ANSWER IT:</i> The answer to the reformulation of the CQ will exclude the Benelux itself.</p>
D.2)	<p><i>COMPETENCY QUESTION:</i> CQ) Which are the direct parts of Benelux?</p> <p><i>HOW TO ANSWER IT:</i> The answer to the reformulation of the CQ will only include the countries of the Benelux and exclude provinces, towns, etc.</p>

Table 24. Examples of heuristics to determine which mereology features are useful

Axioms and definitions	Examples of use
Theory M	
D.3)	<p>COMPETENCY QUESTION: CQ) Do Nordic Countries and EU overlap?</p> <p>HOW TO ANSWER IT: The answer to the reformulation of the CQ will be true, since there is an overlap between Nordic Countries and EU.</p>
D.4)	<p>COMPETENCY QUESTION: CQ) Do Belgium and Luxemburg underlap?</p> <p>HOW TO ANSWER IT: The answer to the reformulation of the CQ will include the common wholes to Belgium and Luxemburg, that is, it will include the Benelux and the European Union.</p>
D.5)	<p>FACTS:</p> <p>Fact 1) The provinces of the Netherlands are Drenthe, Flevoland, Friesland, Gelderland, Groningen, Limburg, North Brabant, North Holland, Overijssel, Utrecht, Zeeland and South Holland.</p> <p>Fact 2) Drenthe, Flevoland, Friesland, Gelderland, Groningen, Limburg, North Brabant, North Holland, Overijssel, Utrecht, Zeeland and South Holland are all different.</p> <p>Fact 3) The only provinces of Holland are North Holland and South Holland.</p> <p>COMPETENCY QUESTION: CQ) What provinces of the Netherlands are disjoint with Holland?</p> <p>HOW TO ANSWER IT: Let's be p a province of the Netherlands different from North Holland and South Holland. From facts 2 and 3, the system can deduce that p is not part of Holland; therefore p and Holland are disjoint. This reasoning is applied to the rest of the provinces of the Netherlands that are not provinces of Holland.</p>
Minimal Mereology (MM)	
A.4)	<p>NEW WRONG AXIOM: Axiom 1) A <i>single-province autonomous community</i> is an autonomous community that has only one proper part.</p> <p>HOW TO DETECT THE MISTAKE: Axiom 1 violates the weak supplementation principle.</p> <p>It is useful to detect the mistake because we cannot say, for example, that the Province of Madrid is the only proper part of the Community of Madrid. If we are only considering its territories, it is not a proper part. If we are considering the political entities, the Community of Madrid contains more elements than the province.</p>

Table 24. Examples of heuristics to determine which mereology features are useful

Axioms and definitions	Examples of use
Extensional Mereology (EM)	
A.5)	<p>FACTS: Fact 1) Nordic Countries is the sum (composition of binary sums) of the following proper parts: Sweden, Norway, Finland and Island. Fact 2) <i>Norden</i>, which is the Scandinavian name for the Nordic Countries, is the sum (composition of binary sums) of the following proper parts: Sweden, Norway, Finland and Island.</p> <p>COMPETENCY QUESTION: CQ) Which territories are identical?</p> <p>HOW TO ANSWER IT: Given that Nordic Countries and Norden have the same proper parts, they are identical.</p>
Closure Extensional Mereology (CM)	
D.6)	<p>COMPETENCY QUESTION: CQ) Is the binary sum of Belgium and Luxemburg a political organization? HOW TO ANSWER IT: The system can calculate the object <i>s</i> whose parts are those of Belgium and Luxemburg, that is, their binary sum. If there is a political organization with the same parts as <i>s</i>, then the answer is yes.</p>
D.7)	<p>COMPETENCY QUESTION: CQ) Is the binary product of Nordic Countries and the European Union a common organization? HOW TO ANSWER IT: The system can calculate the object <i>p</i> whose parts are the common parts of Belgium and Luxemburg, that is, their binary product. If there is an organization with the same parts as <i>p</i>, then the answer is yes.</p>
General (classical) Mereology (GM)	
D.8)	<p>COMPETENCY QUESTION: CQ) Is the general sum of countries with a per capita income greater than 30000 \$ an economical organization? HOW TO ANSWER IT: The system can calculate the object <i>r</i> whose parts are the parts of the countries with a per capita income greater than 30000 \$, that is, their general sum. If there is an economical organization with the same parts as <i>r</i>, then answer is yes.</p>
Atomistic Mereology	
D.9)	<p>RIGHT FACT: Fact 1) The only proper part of Holland is the Netherlands. WRONG FACT: Fact 2) The only proper part of Netherlands is Holland. HOW TO DETECT THE MISTAKE: The system should automatically detect the problem.</p>
A.9)	<p>RIGHT FACT: Fact 1) The only proper part of Holland is the Netherlands. WRONG FACT: Fact 2) The only proper part of Netherlands is Holland. HOW TO DETECT THE MISTAKE: Neither Holland nor the Netherlands have atoms, that is, the atomicity principle is not fulfilled.</p>

Table 24. Examples of heuristics to determine which mereology features are useful

Task 5.3. Determining the most appropriate general ontology.

The goal of this task is to determine which of the candidate general ontologies identified in Activity 3 is the most appropriate to be reused in the ontology being developed. To determine such an ontology, we propose to perform an analysis of all the candidate⁸⁶ general ontologies based on the following four dimensions:

- ❑ *Reuse Cost.* It refers to the estimate of the cost (economic and temporal) needed for the reuse of the candidate ontology. In this case, the following criteria should be analysed:
 - *Reuse Economic Cost.* It refers to an estimate of the economic cost needed for accessing and using the candidate ontology. If the candidate ontology has any type of license, then the cost of acquisition and/or exploitation should be taken into account [Gómez-Pérez and Lozano-Tello, 2005].
 - *Reuse Time Required.* It refers to an estimate of the time required for accessing the candidate ontology. If the candidate ontology is accessible in slow servers or servers with bad connectivity, the time spent in accessing should be taken into account.
- ❑ *Understandability Effort.* It refers to the estimate of the effort needed for understanding the candidate ontology. In this case, the following criteria should be analysed:
 - *Quality of the documentation.* It refers to whether there is any communicable material used to describe or explain different aspects of the candidate ontology (e.g., modelling decisions). The documentation should explain the domain and the knowledge pieces represented in the ontology so that a non-expert could learn enough about the domain and be able to understand the knowledge represented in the ontology [Pinto and Martins, 2001].
 - *Availability of external knowledge sources.* It refers to whether the candidate ontology has references to documentation sources and/or if experts are easily available.
 - *Code clarity.* It refers to whether the code is ease to understand and modify, that is, if the knowledge entities follow unified patterns and are clear [Pinto and Martins, 2001]. It is advantageous to use the same pattern to make sibling definitions, thus improving ontology understanding and making it easier to include new definitions [Gómez-Pérez and Rojas-Amaya, 1999]. This would improve the clarity of the ontology and its monotonic extendibility. This criterion also refers to whether the code is documented, that is, if it includes clear and coherent definitions and comments for the knowledge entities represented in the candidate ontology.
- ❑ *Integration Effort.* It refers to the estimate of the effort needed for integrating the candidate ontology into the ontology being developed. In this case, the following criteria should be analysed:
 - *Adequacy of features.* It refers to how many features of those indentified in Task 5.2 (transitivity, antisymmetry, etc.) are satisfied by the candidate ontology, in the case of general ontologies.
 - *Adequacy of knowledge extraction.* It refers to whether it is easy to identify parts of the candidate ontology to be reused and to extract them. For example, in large and not modularized ontologies (e.g., SUO) the difficulty to extract the part of the knowledge we are interested in is especially high.

⁸⁶ It is worth mentioning that we consider here as candidate general ontologies those that represent at least one feature from the features required by the ontology network being developed.

- *Adequacy of naming conventions.* It refers to whether both ontologies (the candidate and the one being developed) follow the same rules for naming the different ontology components (e.g., concept names should start with capital letters, relation names should start with non-capital letters).
 - *Adequacy of the implementation language.* It refers to whether both languages (the candidate ontology's and the ontology's being developed) are the same, or at least are able to represent similar knowledge with the same granularity.
 - *Knowledge clash.* It refers to whether there are contradictory bits of knowledge between the candidate ontology to be reused and the ontology being developed (e.g., having density in a time ontology and requiring no density in the ontology being developed).
 - *Adaptation to the reasoner.* It refers to whether the adaptation of definitions and axioms that satisfy the existing restrictions of the reasoner are needed (e.g., explicit definitions can be included in OWL ontologies; however, this kind of definitions can not be included in ontologies written in ProLog).
 - *Necessity of bridge terms.* It refers to whether it is necessary to create new linking axioms and/or relations to integrate the candidate ontology to be reused into the ontology being developed
- *Reliability.* It refers to analyzing whether we can trust in the candidate ontology to be reused. In this case, the following criteria should be analysed:
- *Availability of tests.* It refers to whether tests are available for the candidate ontology to be reused.
 - *Former evaluation.* It refers to whether the ontology has been properly evaluated, which here means that there is a set of unit tests that the ontology passed.
 - *Theoretical support.* It refers to whether the candidate ontology is supported by a contrasted theory, in the case of general ontologies.
 - *Development team reputation.* It refers to whether the development team of the candidate ontology is reliable.
 - *Purpose reliability.* It refers to whether the candidate ontology has been developed as a simple academic example; for instance, an ontological resource is less reliable than other resources developed to be used in real projects.
 - *Practical support.* It refers to whether there are well known projects or ontologies reusing the candidate ontology [Lozano-Tello, 2002].

Table 25 shows the criteria (organized by dimensions) and the ways to measure them. In the table, for each criterion we have (a) a range of values (an interval of linguistic values or a natural number), (b) an explanation of how to measure the criterion, and (c) a numerical weight. The numerical weights are proposed here by default, according to the importance we give to the different criteria; for example, the criterion “adequacy of features” is extremely important for us and therefore we establish its numerical weight in 10; however, the criterion “purpose reliability” is not so important for us, therefore, we establish a weight of 3. It is worth mentioning that such numerical weights depend on the importance the ontology developer gives to the different criteria, and that such weights can be modified. The symbols (+) and (-) in the weights are established to note whether the criterion counts in a positive or a negative way, respectively. These symbols can not be modified by the ontology developer.

Criteria	Range of values	How to measure it	Weight	
Reuse cost				
Reuse economic cost	{Unknown, Low, Medium, High}	Asking the owner for an estimate.	(-)	9
Reuse time required	{Unknown, Low, Medium, High}	Trying the connection to the server.	(-)	7
Understandability effort				
Quality of the documentation	{Unknown, Low, Medium, High}	Analyzing if the ontology has documentation, and if such documentation really explains the domain and the ontology itself, as well as modelling criteria used during the ontology development.	(+)	8
Availability of external knowledge	{Unknown, Low, Medium, High}	Analyzing if in the ontology documentation there is any reference to external sources that could be used to better understand the ontology.	(+)	7
Code clarity	{Unknown, Low, Medium, High}	Inspecting the ontology code by analyzing the complexity of the definitions (and axioms) implemented in the ontology.	(+)	8
Integration effort				
Adequacy of features	Natural Number	Counting how many features of those identified in Task 5.2 are covered by the ontology.	(+)	10
Adequacy of knowledge extraction	{Unknown, Low, Medium, High}	Analyzing if the ontology is modularized, or if it can be modularized in an easier way.	(+)	9
Adequacy of naming conventions	{Unknown, Low, Medium, High}	Comparing the naming conventions of both ontologies.	(+)	5
Adequacy of the implementation language	{Unknown, Low, Medium, High}	Comparing the ontology language of both ontologies. If both languages are different, analyzing the loss of knowledge in the translation.	(+)	7
Knowledge clash	{Unknown, Low, Medium, High}	Comparing modelling decisions and knowledge representation decision of both ontologies.	(-)	7
Adaptation to the reasoner	{Unknown, Low, Medium, High}	Comparing the reasoners related to the ontology language of both ontologies.	(+)	7
Necessity of bridge terms	{Unknown, Low, Medium, High}	Inspecting the ontology code.	(-)	6

Table 25. Decision criteria to select a general ontology

Criteria	Range of values	How to measure it	Weight	
Reliability				
Availability of tests	{Unknown, Low, Medium, High}	Analyzing if the ontology documentation refers to existing unit tests.	(+)	8
Former evaluation	{Unknown, Low, Medium, High}	Analyzing if the ontology documentation refers to existing unit tests and the results of such tests.	(+)	8
Theoretical support	{Unknown, Low, Medium, High}	Analyzing if the ontology documentation refers to the theory on which the general ontology is based.	(+)	9
Development team reputation	{Unknown, Low, Medium, High}	Searching for information about the ontology development team (other ontologies developed, papers published, etc.).	(+)	8
Purpose reliability	{Unknown, Low, Medium, High}	Analyzing if the ontology documentation refers to the purpose for which the ontology was developed.	(+)	3
Practical support	{Unknown, Low, Medium, High}	Analyzing if the ontology documentation refers to other ontologies and/or projects reusing the ontology.	(+)	7

Table 25. Decision criteria to select a general ontology

Ontology developers should analyse the candidate ontologies with respect to the abovementioned criteria, taking into account the different ways to measure each criterion and the possible values that can be assigned. And thus, ontology developers should fill a table, like Table 26.

Criteria	Weight	Values		
		Ontology 1	Ontology 2	Ontology 3
Reuse cost				
Reuse economic cost	(-)	9		
Reuse time required	(-)	7		
Understandability effort				
Quality of the documentation	(+)	8		
Availability of external knowledge	(+)	7		
Code clarity	(+)	8		
Integration effort				
Adequacy of features	(+)	10		
Adequacy of knowledge extraction	(+)	9		
Adequacy of naming conventions	(+)	5		
Adequacy of the implementation language	(+)	7		
Knowledge clash	(-)	7		
Adaptation to the reasoner	(+)	7		
Necessity of bridge terms	(-)	6		
Reliability				
Availability of tests	(+)	8		
Former evaluation	(+)	8		
Theoretical support	(+)	9		
Development team reputation	(+)	8		
Purpose reliability	(+)	3		
Practical support	(+)	7		

Table 26. Table to select a general ontology

Having Table 26 filled with different values for each criterion and for each candidate ontology, the ontology developers should obtain the score for each candidate ontology and then decide which one is the most appropriate. To obtain such a score, we propose the following method:

1. To transform the different values (linguistic and numerical) into the same scale [0,3] in the following way:

- To transform *linguistic values* (for all the criteria except for the criterion “Adequacy of features”), we propose the following transformation rules:

Value = Unknown	→	Value _T = 0
Value = Low	→	Value _T = 1
Value = Medium	→	Value _T = 2
Value = High	→	Value _T = 3

where

- Value_T is the transformed value
- Value is the linguistic value provided by the ontology developer
- To transform the *numerical value* provided in the criterion “Adequacy of features”, we propose the following formula:

$$Value_T = \frac{Value \times MaximumNumericalValueinLinguisticTransformation}{NumberofFeatures(obtainedinTask5.2)}$$

where

- Value_T is the transformed value
- Value is the natural number originally obtained in the criterion “Adequacy of features”
- MaximumNumericalValueinLinguisticTransformation is the upper limit in the set of values for transforming the linguistic values. In our case, this value is 3, because we have established the scale [0, 3].
- NumberofFeatures is the number of features obtained in Task 5.2

2. To calculate the score of the different candidate ontologies in the following way:

- We propose to treat independently the criteria weighted with (+) and the criteria weighted with (-). Thus, we propose to use the following formula for obtaining the weighted average for each type of criterion ((+) and (-)):

$$Score_{i(+)} = \sum_{j(+)} Value_{T_{i,j}} \times \frac{Weight_j}{\sum_j Weight_j} \quad Score_{i(-)} = \sum_{j(-)} Value_{T_{i,j}} \times \frac{Weight_j}{\sum_j Weight_j}$$

where

- Score_{i(+)} is the score for the candidate ontology ‘i’ for the set of criteria weighted with (+)
- Score_{i(-)} is the score for the candidate ontology ‘i’ for the set of criteria weighted with (-)
- i is a particular candidate ontology
- j is a particular criterion of those included in Table 25, j(+) means criteria with positive weight, and j(-) criteria with negative weight

- $Value_{Ti,j}$ is the transformed value for the criterion ‘j’ in the ontology ‘i’
- $Weight_j$ is the numerical weight associated to the criterion ‘j’
- The final score for each candidate ontology is obtained with the following formula:

$$Score_i = Score_{i,(+)} - Score_{i,(-)}$$

After applying the previous formula to all the candidate ontologies, ontology developers should select the candidate ontology best scored, that is, the ontology with the biggest score.

Activity 6. Customizing the selected general ontology

The goal of this activity is to customize the general ontology selected in Activiti 5 according to the needs of the ontology network being developed. This activity is divided in the following tasks:

Task 6.1. Pruning the general ontology according to the features needed.

The goal of this task is to prune the general ontology selected taking into account the features needed in the ontology network that is being developed.

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

Task 6.2. Enriching the general ontology.

The goal of this task is to extend the general ontology selected with the new conceptual structures needed in the ontology network being developed.

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

Task 6.3. Translating the general ontology into the implementation language of the ontology network being developed.

The goal of this task is to translate the general ontology selected into the implementation language of the ontology network being developed if both ontologies are in different languages.

The general ontology can be translated in an automatic way or in a manual way. It is important to mention that often the complete translation into different languages is not possible.

Task 6.4. Evaluating the obtained general ontology.

The goal of this task is to evaluate from a content perspective if there are no errors in the general ontology and also to check that the CQs of the ontology network being developed are already represented in the ontology.

Activity 7. Integrating the general ontology in the ontology network being developed.

The goal of this activity is to integrate the general ontology obtained in Activity 7 in the ontology network being developed.

10.4. Methodological Guidelines for Reusing Domain Ontologies as a Whole

The goal of domain ontology reuse is to find and select one or several domain ontologies that can be reused in the ontology being developed.

Table 27 shows the **Filling Card** for the domain ontology reuse process; it includes the definition, the goal, the inputs and outputs, the performer of the process and the time scheduled to 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 development.</p>	
<i>Input</i>	<i>Output</i>
	Ontology network extended with the reused domain ontology.
<i>Who</i>	
Software developers and ontology practitioners.	
<i>When</i>	
The domain ontology reuse process should be carried out after the ontology specification activity.	

Table 27. Domain Ontology Reuse Filling Card

The activities for performing the domain ontology reuse process are explained in detail next and can be seen in the **workflow** presented in Figure 56. In this section, we describe the guidelines proposed, which include input and output information, at the activity level.

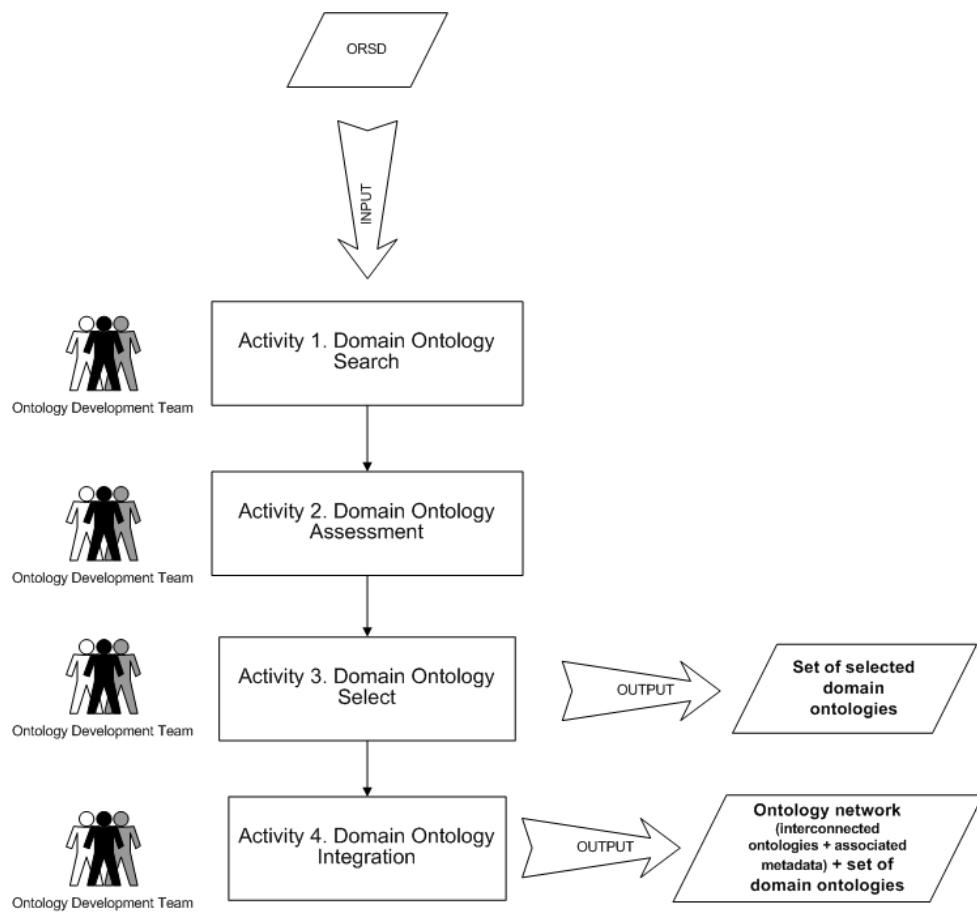


Figure 56. Activities for Reusing Domain Ontologies as a Whole

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 those terms appearing in the pre-glossary of the ORSD (specifically in slot 7) and introducing such terms in Watson or Swoogle.

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. In this activity, the ontology development team should eliminate from the set those domain ontologies not appropriate to be reused. Thus, we propose to analyse the domain coverage for deciding whether a particular domain ontology is useful or not. This means analyzing whether the domain ontology covers (totally or partially) the requirements identified in the ORSD of the ontology network to be developed. To perform this analysis, we propose the following actions:

- Check whether the scope and purpose established in the ORSD are similar to those of the candidate domain ontology.
- Check whether non-functional ontology requirements established in the ORSD are covered by the candidate domain ontology. Examples of non-functional requirements can be “terms to be used in the ontology must be taken from standards”, “multilinguality must be represented in the ontology to be developed”, etc.
- Check whether functional requirements in the form of CQs included in the ORSD are covered (totally or partially) by the candidate domain ontology. This checking can be performed in three different fashions:
 - a) Partially, by analyzing if the essential terms for the new ontology development appear in the candidate domain ontology be reused [Gómez-Pérez and Lozano-Tello, 2005; Peroni et al., 2008].
 - b) Also partially by calculating the precision and coverage of the terminology of the candidate domain ontologies with respect to the terminology included in CQs.

Precision [Baeza-Yates, 1999] is defined as the proportion of the retrieved material that is actually relevant. To adapt this measure to our context we need to define

CandidateDomainOntologyTerminology as the set of terms included in the candidate domain ontology.

ORSDTerminology as the set of identified terms included in the Ontology Requirements Specification Document (ORSD).

Within our context, we can now define precision as the proportion of the terms in the candidate domain ontology included in the identified terms of the ORSD over the terms in the candidate domain ontology. This is expressed as follows:

$$\text{Precision} = \frac{\text{CandidateDomainOntologyTerminology} \cap \text{ORSDTerminology}}{\text{CandidateDomainOntologyTerminology}}$$

Coverage is based on the *recall* measure used in information retrieval [Baeza-Yates, 1999]. Recall is defined as the proportion of relevant material actually retrieved to answer a search request. To adapt this measure to our context, we use the abovementioned definition of *CandidateDomainOntologyTerminology* and *ORSDTerminology*. In this context, coverage is the proportion of the terms of the ORSD identified that are included in the terms included in the candidate domain ontology over the identified terms of the ORSD. This is expressed as follows:

$$\text{Coverage} = \frac{\text{CandidateDomainOntologyTerminology} \cap \text{ORSDTerminology}}{\text{ORSDTerminology}}$$

- c) By determining whether the candidate domain ontology is able to answer the CQs included in the ORSD.

After analysing the set of candidate domain ontologies according to the abovementioned criteria, the ontology development team obtains the assessment table filled (Table 28).

Criteria	Range of Values	Candidate Domain Ontologies		
		<i>Ontology 1</i>	<i>Ontology 2</i>	<i>Ontology 3</i>
<i>Similar Scope</i>	[Yes, No, Unknown]			
<i>Similar Purpose</i>	[Yes, No, Unknown]			
<i>Non-Functional Requirements covered</i>	[Yes-Totally, Yes-Partially, No, Unknown]			
<i>Functional Requirements covered</i>	[Yes-Totally, Yes-Partially, No, Unknown]			

Table 28. Assessment table

To decide whether a candidate domain ontology is not useful, the following heuristic should be taken into account:

- For a particular candidate domain ontology, if the ontology developer has answered *No* to the criteria “Similar Scope” and/or “Similar Purpose” and/or “Functional Requirements covered”, then the candidate ontology should be considered not useful, and thus, it should be eliminated from the set of candidate ontologies.

The ontology development team should shadow the useful candidate ontologies in Table 28.

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 (specifically, those shaded in the table). To distinguish between those candidate domain ontologies which are the most suitable, we propose to use the set of criteria defined in Section 10.3.3 (Task 5.3), with the exception of the criteria “Adequacy of features” and “Theoretical support” that apply only to general ontologies. Thus, ontology developers should fill a table like Table 26, taking into account the ways to measure each criterion and the possible values that can be assigned, as explained in Section 10.3.3 (Task 5.3).

In this activity we also propose to use the method and the formulae explained in Section 10.3.3 (Task 5.3) to obtain the score for each candidate ontology.

As a result of this activity, the ontology developers should select the candidate domain ontology best scored.

Activity 4. Domain Ontology Integration.

The objective of this activity is to integrate the domain ontologies selected in Activity 3 in the ontology network being developed. The ontology development team decides one of the following three modes for integrating each of the domain ontologies already selected:

- ❑ The selected domain ontology is reused as it is. The ontology development team integrates the domain ontology in the ontology network being developed.
- ❑ The selected domain ontology is reused with significant changes. Those changes might be due to the use of the domain ontology in a different implementation language. In this mode, the ontological resource reengineering activity or the ontology restructuring activity could be carried out with the domain ontology selected, according to the type of changes to be performed.
- ❑ 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 27 from Chapter 6) should be followed.

Before reusing the domain ontologies selected by following any reuse mode, it is also convenient to evaluate the domain ontologies through the *ontology evaluation activity* (defined in Section 5.3.1).

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

10.5. Methodological 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 that will be used in the ontology network being developed.

The reuse of large ontologies (such as the NCI ontology) is difficult because they contain a large amount of knowledge that may not be needed when developing a particular ontology. Sometimes, reuse demands to retrieve pieces of knowledge (e.g., statements) to be integrated in the new ontology being built rather than to reuse entire ontologies.

Table 29 shows the **Filling Card** for the ontology statement reuse process. It includes the definition, the goal, the inputs and outputs, the performer of the process and the time scheduled for the process.

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

Situation 1. Building ontology networks from scratch.

Situation 2. Extending or improving existing ontology networks.

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

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 in hand.</p>	
<i>Input</i>	<i>Output</i>
<p>The ORSD 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 at 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 29. Ontology Statement Reuse Filling Card

The activities for carrying out the ontology statement reuse process are explained in detail next and can be seen in the **workflow** shown in Figure 57. In this section, we describe the guidelines proposed at activity level that includes also input and output information.

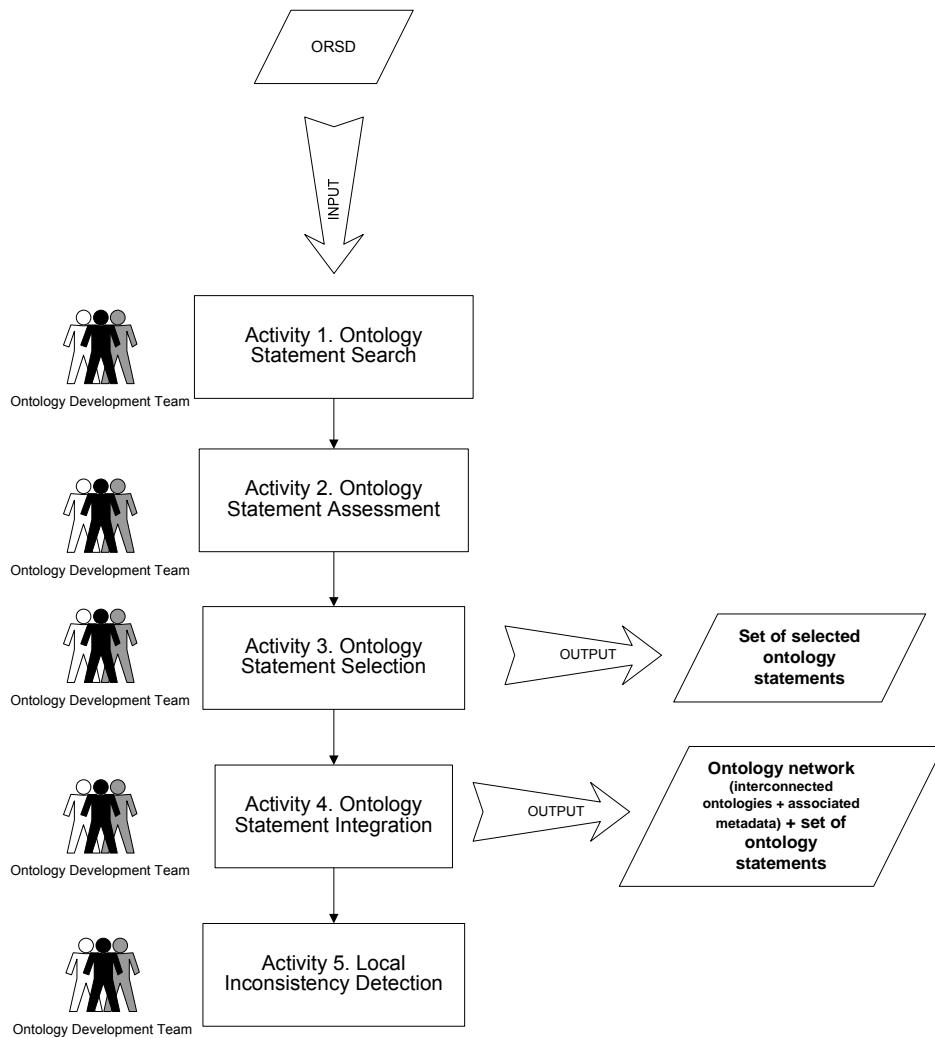


Figure 57. Activities for the Ontology Statement Reuse

Activity 1. Ontology Statement Search.

The goal of this activity is to search the Web for ontology statements candidates that could satisfy the ontological needs in a particular case. The ontology development team carries out this activity taking as input the ORSD, specifically those terms that have a high frequency in the ORSD and using some gateways to the Semantic Web, such as Watson on-line or the Watson plug-in in the NeOn Toolkit.

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 whether a concrete ontology statement is useful 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.

Given the heterogeneity of the ontologies available on-line from a quality perspective, the assessment activity is not trivial and can be a major issue for practitioners that are not ontology engineers. 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 plug-in to build and extend ontologies by reusing statements provided by on-line ontologies. Such experiments are detailed in Annex III. We regard these identified criteria as initial criteria; hence, we wish to refine and extend them in the future based both on further experimentation and on practitioner's feedback.

- Adequacy of the ontology statement.* It refers to the analysis of whether the ontology statement is relevant for the ontology network being developed. In this case, the following features should be analysed:

- If the ontology statement belongs to an ontology with 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.

- If the ontology statement belongs to an ontology with similar purpose to the ontology network being developed.

For example, the statements "Meeting is subclass of Activity"⁸⁷ and "Meeting is subclass of IntervalEvent"⁸⁸ can be true, but they may not be useful for the purpose of the ontology being developed (to represent knowledge about European or research projects).

- If the ontology statement contains contradictory bits of knowledge with respect to the ontology network being developed.

For example, if the ontology statement is "Teacher is subclass of Person" and in the ontology network being built this kind of knowledge is represented with roles.

- Understandability of the ontology statement.* It refers to whether the naming pitfall⁸⁹ [Poveda et al., 2009] occurs in the ontology statement. In this case, the following feature should be analysed:

- The clarity of the ontology statement. It means checking if ontology elements in the statement have a name with no sense, which implies that the statement itself has no sense. Such type of statements should not be reused.

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.

- Information content of the ontology statement.* It refers to the analysis of whether the ontology statement provides additional information.

For example, statements that provide little additional information are those linking a concept to an abstract root concept (e.g., "Publication is subclass of: 'Root', 'Object', 'Thing', 'DEF_ROOT_CONCEPT', and 'Resource'").

⁸⁷ <http://calo.sri.com/core-plus-office>

⁸⁸ <http://pervasive.semanticweb.org/ont/dev/meeting>

⁸⁹ It refers to the ontology usability from the user's point of view and, specifically, to the naming of the ontology elements. It is important to note that the naming of the elements should provide the users with a better understanding of the ontology.

Another example can be the statements that declare that a concept is equivalent to itself (“Publication is equivalent to: ‘Publication’”). This is related to the reasoning pitfall⁹⁰ [Poveda et al., 2009] called recursive definition. This pitfall entails using an ontology element in its own definition.

These types of statements should not be reused.

- Correctness of the ontology statement.* It refers to the analysis of whether the ontology statement is correct from a formal modelling perspective. In this case, the following features should be analysed:

- If the naming of concepts in the ontology statement reflects the intended meaning of the statement given its ontological context. This feature is related to the naming pitfall [Poveda et al., 2009].

For example, “Project is subclass of Event”⁹¹ is not correct, because the name ‘Project’ does not reflect the intended meaning of the statement, which, given its context in the ontology (the ontology is about different events), is that ‘Project’ refers to ‘ProjectEvent’ (it can be ‘project meetings’, ‘project reviews’, etc.). Another example of this kind of error is “Book is subclass of Image”⁹² (in this case, ‘Book’ really refers to ‘BookImage’ in an ontology about images).

When such statements are reused it is important to rename the concepts in a way that these clearly reflect the meaning of the statement.

- If the ontology statement is invalid from a formal perspective, e.g., by confusing the hierarchical relation “subclassOf” with the mereological relation “partOf” or with other type of ad-hoc relations (e.g., “relatedTo”). This feature is related to the inclusion of antipatterns in the modelling [Poveda et al., 2009].

For example, (1) “Chapter subclass of Book”⁹³ where the relation ‘subclassOf’ is incorrectly used for modelling partonomy; and (2) “Article is subclass of Journal”⁹⁴ is an example in which a relatedness relation (e.g., published in) is modelled through subsumption. These types of modelling errors are fairly common in on-line ontologies and the person performing the reuse should avoid introducing such errors. One way to avoid this kind of statements is to ask (search in Internet) if “a chapter is a book”.

The approach to validate ontology statements using lexico syntactic patterns [Aguado et al., 2009] can be used to check this feature.

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 among 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. For comparing ontology statements and selecting the most convenient ones the ontology development team should use the following criterion:

⁹⁰ It refers to the implicit knowledge derived from the ontology when reasoning procedures are applied to such an ontology.

⁹¹ <http://www.ontotext.com/kim/2004/04kimo>

⁹² <http://studioddonline.web.infoseek.co.jp/ns/aniota>

⁹³ http://www.cs.abdn.ac.uk/~cmckenzi/playpen/rdf/akt_ontology_LITE.owl

⁹⁴ Refer to footnote 93

- Minimum effort needed for integrating the ontology statement in the ontology network being developed. For example, consider the case in which several ontology statements are valid, but they use different naming conventions. Then, the proposal (if possible) is to reuse the statements that use the same naming convention as the one used in the ontology network being developed. This approach avoids adapting the statement reused to the ontology network being built.

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

Activity 4. Ontology Statement Integration.

The goal of this activity is to integrate the ontology statements selected in Activity 3 into the ontology network being developed. The ontology development team should use any of the following three integration modes:

- a) statements will be reused as they are (without requiring any effort from the ontology developers)
- b) statements will be reengineered, and
- c) statements will be merged.

Apart from these integration modes, the development team should decide among:

- Importing the ontology statements. The advantage is that importing maintains a link with the ontology from which the statement was originated. As a side effect, other elements of the ontology can have an impact on the ontology being built.
- Copying the ontology statements. As copying reproduces the statement, it ensures that no side effect will appear, that is, only the statement itself is integrated, but loses the link with the original ontology.
- Establishing mappings with the ontology statements. This can be seen as a compromise solution where the statement is first copied in the ontology being built and the newly created entities are aligned with the entities of the original ontology. In this way, links are maintained between these ontologies and the side effects can be more easily controlled.

After integrating an ontology statement, the following tasks will probably have to be done: (a) changing names (concepts, properties) to adapt them to the naming conventions used in the ontology network being developed; (b) adding range to properties and changing cardinalities; and (c) adding restrictions.

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

Activity 5. Local Inconsistency Detection.

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

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

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

10.6. Conclusions

One way of reducing time and also costs associated to the ontology development is by means of reusing the available ontological resources. The reuse of (well-developed) ontological resources allows spreading good practices and increasing the overall quality of ontological models. In this chapter we have systematized the reuse of ontological resources by proposing detailed and prescriptive methodological guidelines.

Since the ontological resource reuse process is often influenced by the type of ontology to be reused, in our method we have considered two different types: general ontologies and domain ontologies.

- The methodological guidelines for reusing general ontologies provide ontology practitioners with (1) documented knowledge about two different theories (time and mereology) and (2) a clear procedure for reusing this type of ontologies.
- The methodological guidelines for reusing domain ontologies provide ontology practitioners with a clear and prescriptive procedure on how to reuse domain ontologies as a whole in the ontology development.

Additionally, we have shown along this chapter that while in the state of the art there are no detailed methods to reuse ontological resources, a precise method is established in this thesis for reusing ontological resources at different levels of granularity: (a) by reusing ontological resources as a whole, and (b) by reusing ontology statements.

- The methodological guidelines for reusing ontology statements provide ontology practitioners with a clear and prescriptive procedure on how to reuse ontology statements in the ontology network development.

The guidelines proposed in this chapter provide the methodological assistance to Scenario 3 in the NeOn Methodology (Chapter 6).

11. Experimentation

11.1. Introduction

This chapter describes a set of experiments carried out to verify the results presented in this thesis. These experiments are divided into two groups:

- The so-called *action research* (Section 11.2) that is the experimentation attempts to solve a real-world problem while simultaneously we studied the experience of solving that problem [Easterbrook et al., 2007]. Action research is a reflective process of progressive problem solving to improve the way in which issues are addressed and problems are solved.

The idea here is to solve a particular problem following preliminary guidelines and at the same time to improve such methodological guidelines.

In this section we provide experiments on scenarios, models and life cycle establishment, and on requirements specification and reuse. These experiments are related respectively to the following hypotheses established in Chapter 3:

- H3: Ontology network development entails combining different scenarios that involve reuse of ontological and non-ontological resources, reengineering, merging, restructuration, and localization.
- H2: A collection of ontology network life cycle models (the waterfall model and the iterative-incremental model) is enough to allow ontology practitioners to develop ontology networks.
- H5: A clear, simple, and useful procedure to select and instantiate an ontology network life cycle model, as part of the scheduling activity, can be established.
- H4: A precise method for specifying the ontology requirements based on competency questions can be established.
- H6: A clear, simple, and useful procedure to reuse ontological resources at different levels of granularity can be established.

It is worth mentioning that for the ontology requirements specification we provide in Annex III how methodological guidelines are applied in different use cases.

- Controlled experiments* (Section 11.3), in which we included experiments on requirements specification and on establishing the ontology network life cycle (without and with gOntt).

Such experiments are related respectively to the following hypotheses established in Chapter 3:

- H4: A precise method for specifying the ontology requirements based on competency questions can be established.
- H2: A collection of ontology network life cycle models (the waterfall model and the iterative-incremental model) is enough to allow ontology practitioners to develop ontology networks.
- H5: A clear, simple, and useful procedure to select and instantiate an ontology network life cycle model, as part of the scheduling activity, can be established.

11.2. Action Research

In this section we include the experimentation attempt we carried out to solve a real-world problem while simultaneously studied the experience of solving that problem [Easterbrook et al., 2007].

For this kind of experimentation, we mainly focused on the following use cases:

- ❑ The invoice NeOn (FP6-027595) use case, whose aim is to build an ontology network for the invoicing management.
- ❑ The pharmaceutical NeOn (FP6-027595) use case, whose aim is to develop an ontology network for the pharmaceutical nomenclator.
- ❑ The SEEMP (FP6-27347) use case, whose aim is to build an ontology network that represents knowledge about Curricula Vitae and job offers among Employment Services (ES) placed in different countries.
- ❑ The Spanish project GeoBuddies⁹⁵ (Collaborative Semantic Annotation with Mobile Devices for St. James' Way. TSI-2007-65677-C02), whose main purpose is to develop an ontology network-based infrastructure to enable pilgrims in St. James' Way to share their experiences through mobile devices.

We use the aforementioned use cases to perform experiments (a) on the scenarios identified in the NeOn Methodology, (b) on the life cycle models proposed, and (c) on the requirements specification and reuse.

11.2.1. Experiments on Scenarios for Building Ontology Networks

In this section we present how the different scenarios identified in the framework of the NeOn Methodology and described in Chapter 6 have been used in several cases.

The NeOn Invoicing Management use case. The ontology network for invoicing management use case, which contains all the concepts related to the invoice management in the pharmaceutical industry, was built following Scenarios 1, 2, 3, 8, and 9 [Gómez-Pérez et al., 2007, Suárez-Figueroa et al., 2007].

- ❑ Ontology developers reused different non-ontological resources for electronic invoicing (e.g., Universal Business Language (UBL), EDIFACT, or other industrial proprietary solutions) as part of *Scenario 2*.
- ❑ A generic description of concepts from SUMO, DOLCE, and W3C Time⁹⁶ ontologies were reused as part of *Scenario 3*, using the methodological guidelines for reusing ontological resources.
- ❑ The general invoice ontology was specialized for each laboratory involved in the use case as part of *Scenario 8*.
- ❑ Regarding *Scenario 9*, ontology users belong to different regions in Spain in which different languages are used.

⁹⁵ <http://www.geobuddies.net/>

⁹⁶ <http://www.w3.org/TR/owl-time/>

The NeOn Semantic Nomenclator use case. The ontology network for the semantic nomenclature use case, whose main objective is the integration of different and heterogeneous pharmaceutical product information repositories, was built following Scenarios 1, 2, 3, and 9 [Gómez-Pérez et al., 2007; Suárez-Figueroa et al., 2007].

- ❑ Different legacy systems, databases (Digitalis⁹⁷, Integra⁹⁷, and BOTPlus⁹⁸), thesauri (ATC classification⁹⁹, Snomed¹⁰⁰, UMLS¹⁰¹, etc.), vocabularies and classifications, modelling information of the pharmaceutical products, were reused and reengineering within Scenario 2.
- ❑ Additionally, some modules from time, geographical, and measure ontologies were reused as part of Scenario 3, using the methodological guidelines for reusing ontological resources.
- ❑ Because of Spain's multilingualism (Spanish, Catalonian, Basque and Galician are the languages used) and the internationalization of the semantic nomenclature ontology, the ontology localization was performed as part of Scenario 9.

The SEEMP use case. The ontology network for the SEEMP use case should represent knowledge about Curricula Vitae and job offers among the Employment Services (ES) placed in different countries. Such a network is formed by (1) a set of local ontology networks, representing each of them the local and particular view that each ES has of the employment market, and (2) a reference ontology network representing a standardized and agreed view of the employment market at the European level. The SEEMP ontology is a network of ontology networks that was built following Scenarios 1, 2, 4 and 5.

- ❑ To build the local and the reference ontology networks, several NORs (international standards like NACE¹⁰², ISCO-88¹⁰³ (COM), FOET (Fields of Education and Training)¹⁰⁴, etc.; ES classifications; and international codes like ISO 3166¹⁰⁵, ISO 6392¹⁰⁶, etc.) were reused and re-engineered as part of Scenario 2.
- ❑ Additionally, the DAML time ontology was reused (using the methodological guidelines for reusing ontological resources) and re-engineered as part of Scenario 4.
- ❑ Finally, as part of Scenario 5, a set of alignments were defined between each local ontology and the reference ontology in order to create the SEEMP network.

⁹⁷ <http://www.msc.es/profesionales/farmacia/nomencladorDI.htm>

⁹⁸ <http://botplusweb.portalfarma.com/>

⁹⁹ http://www.whocc.no/atc_ddd_index/

¹⁰⁰ http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

¹⁰¹ <http://www.nlm.nih.gov/research/umls/>

¹⁰²

http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_NOM_DTL&StrNom=NACE_REV2&StrLanguageCode=EN&IntPcKey=&StrLayoutCode=HIERARCHIC&CFID=903021&CFTOKEN=4b888be3cfcb1390-81A9FDB3-C49D-1480-52856A147AA560A1&jsessionid=1f51f24e7dc95178d2a9624942316c6e3d15TR

¹⁰³ <http://www.ilo.org/public/english/bureau/stat/isco/isco88/index.htm>

¹⁰⁴

http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_NOM_DTL&StrNom=EDU_TRAINI&StrLanguageCode=EN&IntPcKey=&StrLayoutCode=HIERARCHIC&CFID=903031&CFTOKEN=afadecc560ee09f6-81AA5BC7-9840-C4FF-8E811706C0241A48&jsessionid=1f51f24e7dc95178d2a9624942316c6e3d15TR

¹⁰⁵ http://www.iso.org/iso/country_codes/iso_3166_databases.htm

¹⁰⁶ http://www.loc.gov/standards/iso639-2/php/code_list.php

The GeoBuddies use case. The ontology network for the Spanish project GeoBuddies should represent knowledge about art, geography, architecture, points of interest, and services related to St. James' Way. This ontology network was built following Scenarios 1, 3, 4, 7, 8 and 9.

- ❑ The main approach in the development of this ontology network was to reusing (using the methodological guidelines for reusing ontological resources) and reengineering ontological resources (domain and general ontologies) instead of building the ontology network from scratch, thus it involved *Scenarios 3 and 4*.
- ❑ Additionally, ontology design patterns (concretely logical, arquitectural and content patterns [Suárez-Figueroa et al., 2007b, Presutti et al., 2008]) were reused as part of *Scenario 7*.
- ❑ Different types of restructuration were needed in the ontology network development within *Scenario 8*.
- ❑ Finally, the ontology was build in Spanish, but it was also needed in English and Galician languages, and thus, the ontology localization was performed as part of *Scenario 9*.

11.2.2. Experiments on Life Cycle Models

In this section we describe how ontology developers involved in two of the NeOn use cases selected the most appropriate life cycle model and the processes and activities to be used to establish the life cycle for the ontology development.

The NeOn Invoicing Management use case. The preliminary workplan for developing the Invoice Management ontology network included 4 main tasks and followed a waterfall life cycle model. The tasks were the following:

- ❑ *Task 1. Search of resources related to the invoicing problem.*
- ❑ *Task 2. Design of the initial ontology network.*
- ❑ *Task 3. Validation and verification of the ontology.*
- ❑ *Task 4. Instantiation of the invoice management ontology network.*

The aforementioned initial workplan for developing the ontology network in the Invoice Management use case was modified using (a) the preliminary collection of life cycle models (summarized in Section 7.1.2) and (b) the preliminary guidelines for establishing the ontology network life cycle included in [Suárez-Figueroa et al., 2007].

In this regard, ontology developers selected as the ontology network life cycle model in the invoicing management use case the *incremental ontology network life cycle model*. This decision was taken because several prototypes and intermediate ontology networks should be produced and used during the evolution of the use case.

After the selection of the ontology network life cycle model, the processes and activities to be mapped in the incremental model were chosen by ontology developers using the table of “Required-If Applicable” processes and activities (Table 5). Having the model and the set of processes and activities, ontology developers established the ontology network life cycle.

The “if applicable” processes and activities selected by the ontology developers were the non-ontological resource reuse and reengineering, the ontological resource reuse (particularly, the reuse of upper level ontologies, general ontologies, and domain ontologies), the ontology specialization, and the ontology localization.

The NeOn Semantic Nomenclator use case. The main scenario in the Semantic Nomenclature case study is to create the Nomenclature ontology network. The preliminary Semantic Nomenclature workplan for developing the Nomenclature ontology network is the following:

- ❑ Task 1. Modelization of ontologies related to the main pharmaceutical databases of the case study (*Digitalis, Integra, and BOTPlus*).
- ❑ Task 2. Population of ontologies.
- ❑ Task 3. Creation of a reference ontology in the pharmaceutical domain based in the previous ones.
- ❑ Task 4. Connection of the ontologies via mappings.

However, based on the preliminary collection of life cycle models (summarized in Section 7.1.2) and on the preliminary guidelines for establishing the ontology network life cycle included in [Suárez-Figueroa et al., 2007], the ontology developers team agreed to modify the initial workplan for developing the ontology network in the semantic nomenclature use case.

After identifying and analysing the different ontology network requirements and restrictions, reviewing the possible ontology network life cycle models (summarized in Section 7.1.2) and taking into consideration the past experiences of the ATOS team, the model selected was the *incremental ontology network life cycle model*. Such selection was done by following the preliminary guidelines on how to establish the ontology network life cycle included in [Suárez-Figueroa et al., 2007]. The reason behind this decision was mainly based on the necessity of producing different versions of the ontology network.

After this selection, the processes and activities to be mapped in the incremental model were chosen by ontology developers using the table of “Required-If Applicable” processes and activities (Table 5). Having the model and the set of processes and activities, ontology developers established the ontology network life cycle.

The “if applicable” processes and activities selected by the ontology developers were the non-ontological resource reuse and reengineering, the ontological resource reuse, and the ontology localization.

11.2.3. Experiments on Requirements Specification and Reuse

In this section we briefly describe how ontology developers specified the ontology requirements in the ontology requirements specification document (ORSD) and reused ontological resources.

The Geobuddies use case. The main approach in the development of the ontology network for the Geobuddies project was to reuse ontological resources instead of building the ontology network from scratch. As a starting point, the set of requirements that the ontology network should fulfil was established following the NeOn methodological guidelines for the ontology requirements specification activity (explained in Chapter 8). Based on such a set of requirements, the ontology network development was performed according to a combination of several scenarios proposed in the NeOn Methodology. In the case of the ontological resource reuse, ontology developers stated the usefulness of the pre-glossary of terms during the reuse process and the conceptualization, and the usefulness of the methodological guidelines for such a reuse process.

The Sixth Summer School on Ontologies and Semantic Web use case. In the hands-on session during the Sixth Summer School on Ontologies and Semantic Web we tested the combination of the ontology requirements specification activity and the ontology reuse process using Watson with 50 students. In this case, we obtained a very positive feedback from most of the students who carried out the hands-on. They affirmed that the proposed guidelines for the activity and the process were useful, and that they would use again such guidelines in future ontology developments. They also expressed that the writing of the ORSD before going into the ontology development was not a waste of time, and that the ORSD was useful in the ontology development.

In both use cases, some of the students commented they would prefer to have such guidelines both for ontology requirements specification and for ontological resource reuse integrated in a tool.

11.3. Controlled Experiments

In this section we include three different experiments carried out with students attending the Master Course at UPM. The main aim of the two first experiments was to test methodological guidelines proposed in this thesis for: (a) ontology requirements specification activity and (b) the establishment of ontology network life cycle, as part of the scheduling activity. The main aim of the third experiment was to learn about the understandability and usability of the gOntt plug-in.

11.3.1. Experiments for the Ontology Requirements Specification Activity

Methodological guidelines to carry out the ontology requirements specification activity have been included in the framework of the NeOn Methodology for building ontology networks (Chapter 8). In this section, we propose an experiment to learn about the understandability and usability of the proposed methodological guidelines for carrying out the ontology requirements specification activity.

11.3.1.1. Assumptions and user study setup

To gather the results of this experiment we propose a questionnaire about the methodological guidelines for the ontology requirements specification activity

People carrying out the experiment have different experience level and background in databases, software engineering, etc., but no extensive experience in ontology engineering.

11.3.1.2. Findings and observations

This experiment has been divided into two studies:

1. User study 1. Experiment carried out with preliminary methodological guidelines on ontology requirements specification is described in Section 11.3.1.3.
2. User study 2. Experiment carried out with quasi-final methodological guidelines on ontology requirements specification is described in Section 11.3.1.4.

11.3.1.3. User study 1

In this first study, we carried out the experiment within the “Artificial Intelligence (AI)” master course at Facultad de Informática (Universidad Politécnica de Madrid) with 14 master students working in pairs, having background in databases, software engineering, and artificial intelligence, but not in ontology engineering. The experiment was carried out during November 2007.

This study was performed with preliminary guidelines for ontology requirements specification. Figure 58 shows the workflow corresponding to such preliminary guidelines.

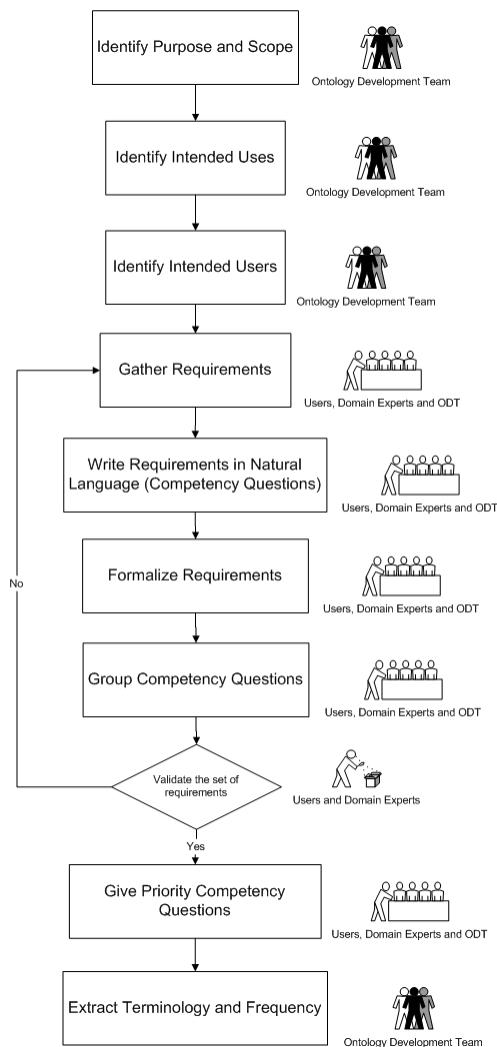


Figure 58. Workflow corresponding to the preliminary methodological guidelines for Ontology Requirements Specification

The experiment consists of the following parts:

1. Lectures provide to students the proposed preliminary methodological guidelines for the ontology requirements specification activity.
 2. Student groups follow the methodological guidelines proposed to carry out the ontology requirements specification activity.
- Students were divided into two sets and each set followed a different instantiation of the methodological guidelines (Guideline-Instantiation-1 and Guideline-Instantiation-2 included in Annex IV). Students had two weeks for carrying out the experiment.
3. Students document in detail each task proposed in the methodological guidelines and performed during the ontology requirements specification activity.
 4. Students fill the questionnaire included in Annex IV about the preliminary methodological guidelines.

11.3.1.4. User study 2

In this second study, we carried out the experiment within the “Artificial Intelligence (AI)” master course at Facultad de Informática (Universidad Politécnica de Madrid) with 12 master students working in groups of one or two, having background in databases, software engineering, and artificial intelligence, but not in ontology engineering.

The experiment was carried out during November 2008.

This study was performed with quasi-final methodological guidelines for ontology requirements specification. Figure 59 shows the workflow corresponding to such methodological guidelines. Such guidelines were produced from the preliminary ones and taking into account the general comments obtained during user study 1.

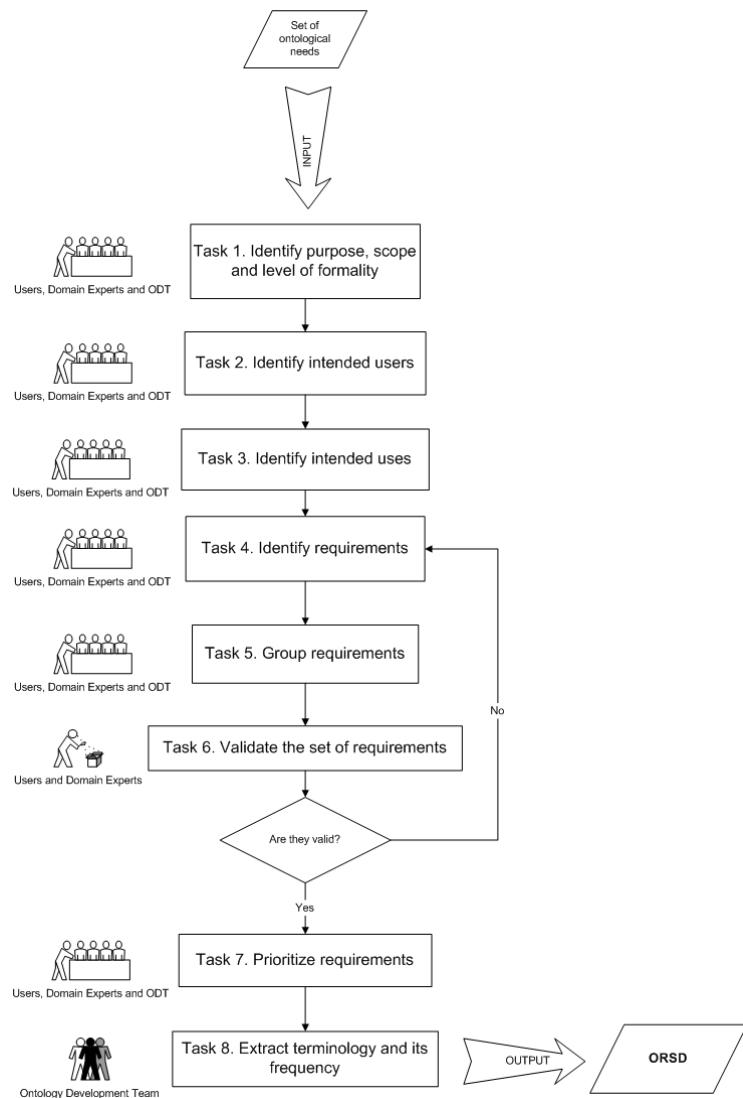


Figure 59. Workflow corresponding to the quasi-final methodological guidelines for Ontology Requirements Specification

The experiment consists of the following parts:

1. Lectures provide to students the quasi-final methodological guidelines for the ontology requirements specification activity.
2. Student groups follow the methodological guidelines to carry out the ontology requirements specification activity.
Students had two weeks for carrying out the experiment using the provided material.
3. Students document in detail each task proposed in the methodological guidelines and performed during the ontology requirements specification activity.
4. Students fill the questionnaire included in Annex V about the quasi-final methodological guidelines for the ontology requirements specification activity.

11.3.1.5. Analysis and discussion

In this section we include the analysis of the two user studies carried out with the methodological guidelines for the ontology requirements specification activity.

Analysis of user study 1

As already mentioned, in user study 1 the students used preliminary methodological guidelines for the ontology requirements specification activity. Here, we analyse the questionnaire filled by the 7 groups of master students (composed by 2 people) that carried out the experiment.

Regarding *general issues* we asked about the guidelines in the questionnaire (Annex IV), these are the main conclusions:

- 86% of the students considered that the preliminary methodological guidelines were well explained.
General comments regarding this aspect were:
 - Some tasks are explained in a very brief way.
 - Some kind of pre-knowledge about general process of building ontologies is needed in order to obtain benefit of the guidelines.
- 100% of the students commented that more detail was needed in the preliminary guidelines. Concretely, most of the comments were in the following line:
 - More detail on how to apply the proposed techniques should be included in the guidelines, concretely on how to build the mind map diagram and on how to write the competency questions (CQs).
 - Detailed explanations should be included in the tasks corresponding to the CQs development and validation.
 - More help about recommended tools should be also included in the guidelines.

- 72% of the students thought that more techniques and especially more tools should be provided as recommendation in the methodological guidelines.
- 43% of the students did not miss an integrated tool to carry out the proposed tasks in the ontology requirements specification activity; however, 57% missed such an integrated tool.

In the latter case, the general comment was that in large ontology projects, to have an integrated environment with all the tools (for writing CQs, for grouping CQs, for validating CQs, and for extracting terminology) would be very useful.

The following suggestions were provided by the students to improve the preliminary methodological guidelines:

- More detail on how to carry out some tasks, and more variety of techniques and tools to be used should be included.
- More free tools and examples should be included.
- More detail in the three first tasks of the ontology requirements specification activity should be included.
- The task of validating CQs should be divided into subtasks to facilitate the performance of such task, and more examples on how to validate CQs should be included.

Additionally, we obtained the following general comments given by students:

- One of the most difficult parts was to decide the group criteria in the task of grouping CQs and to validate the set of CQs.
- Mind map tools are very useful, specially to sort and classify ideas and the questions themselves.
- To group CQs is useful because it permits clearly to identify the essential parts to be covered by the ontology.
- Modulation approach is a good idea. It lets you find faster the questions, and increases the specification cohesion.
- To extract the terminology and its frequency is useful to know the terms that will form part of the core of the ontology. This provides the necessary input (terminology to be used) to the conceptualization activity and/or to the reuse process. However, tools for extracting automatically the terms were not useful in most of the cases.

Analysis of user study 2

Here, we analyse the questionnaire filled by the 7 groups of master students (composed by 1 or 2 people) that carried out the experiment.

The main conclusions obtained from the questionnaire filled by master students (Annex V) are:

- 100% of the students commented that the proposed methodological guidelines for the ontology requirements specification activity were well explained and very detailed.

General comments regarding this aspect were:

- Most of the students mentioned that the provided example clarifies the guidelines.
 - Due to the students had no experience in ontology engineering, they pointed out that they would have had more problems during the ontology requirements specification activity if they would not have had the proposed methodological guidelines.
- 86% of the students considered that the methodological guidelines needed more detail.

General comments regarding this aspect were:

- Concretely, they mentioned that more detail is needed in the task of writing CQs, and also more detailed explanation and/or more examples are needed in the task of grouping CQs.
- Other students considered that the guidelines should explain with more details the task of giving priority to the CQs. In this case, if several groups are involved in the ontology requirements specification activity it is probable that a protocol to achieve consensus is needed and such a protocol should be proposed and explained in the guidelines.

- Other comment was that the provided example is very useful, and that more examples should be provided in the guidelines.

Additionally, 14% of the students considered that the level of detail provided in the guidelines is enough and that the complete example provided in the guidelines facilitates the performance of the activity.

- 72% of the students considered that the methodological guidelines were not complete.

General comments regarding this aspect were:

- Some of the students expressed that task 1 in the guidelines was not well explained and that the meaning of "level of formality" was not clear enough.
- Other point not well explained according to student's opinion was the task of obtaining the terminology. Students commented that it was not clear enough if the instances are objects and when a concrete term should be considered as an object.

- 57% of the students answered that no more techniques and/or tools were needed in the guidelines.

On the contrary, 43% of the students considered that including more tools in the guidelines could be useful. They suggested the following tools: (1) a visual tool that provides the needed information about each task of the guidelines, and (2) some tools to calculate the terminology frequency.

- 57% of the students considered that an integrated tool for carrying out the proposed tasks was not needed.

However, 43% of the students mentioned that an integrated tool that guides the user through the different tasks to be carried out and provides the information need in each task based on the methodological guidelines should be very useful. Additionally, students commented that such a tool should integrate a mind map editor and a tool to count the terminology.

- 86 % of the students considered that the guidelines for the ontology requirements specification activity were useful.
- 100% of the students expressed their intention of using again the methodological guidelines for the ontology requirements specification activity. But, some of the students commented that they would prefer to have such guidelines integrated in an ontology requirements specification tool.
- 100% of the students commented that they found useful to write the ontology requirement specification document before going into the ontology development.

Concretely, most of the comments were in the following line:

- Similar to any software development, a previous specification of what is required in the final product is necessary. Otherwise the final product (in this case the ontology) could not fulfil the expectations.
- CQs are a good way to think about the problem before going into the development.
- Ontology requirements allow having an ontology verification element, as in any software development.
- An ontology requirements specification is needed during the ontology development in order to focus in the knowledge to be covered by the ontology.
- 86 % of the students considered that they will create ontology requirements specifications in the future when the ontology is to be used in a real application.

Additionally to the previous comments, the following suggestions were provided by the students to improve the methodological guidelines:

- It would be useful to include in the guidelines standard CQs (if possible) applicable to any domain.
- To translate the guidelines to other natural languages (e.g., Spanish) would be also valuable.
- To include more examples and complete ontology requirement specification documents (ORSDs) from different ontology projects would be very useful as guidelines.
- To have a graphic tool that creates the final ORSD would be very worthwhile.

11.3.1.6. Identified strengths and weaknesses

After analysing the results obtained in the two user studies performed with the guidelines for the ontology requirements specification activity, we can mention as strengths the following ones:

- The final methodological guidelines for the ontology requirement specification activity used in the user study 2 are well explained according to 100% of the students. This means that the changes we performed in the preliminary guidelines (used in the user study 1) to obtain the quasi-final guidelines allowed us to improve them.
- All the students performed the user study 2 agreed that they would use again the methodological guidelines for the ontology requirement specification activity. However, in some cases, students mentioned that they would prefer to use such methodological guidelines included in a tool for performing the ontology requirements specification activity.

With respect to the weaknesses, we can comment the following:

- The three first tasks in the methodological guidelines should be better explained.
- More detail should be included in the tasks about writing and validating CQs and giving priority to CQs.
- The task about extracting terminology should be clarified.

It is worth to mention that the percentage of students that missed an integrated tool for performing the ontology requirement specification activity and those that did not miss such a tool was similar in both user studies.

11.3.1.7. Improvement Actions Performed

Based on the analysis carried out with the data extracted from the questionnaires of both studies, we already modified and improved the methodological guidelines used in both studies, obtaining the methodological guidelines presented in Chapter 8. Such methodological guidelines include the following changes:

- More detail in the three first tasks of the methodological guidelines.
- Clarification of the task about extracting terminology.
- More examples and complete ontology requirement specification documents (ORSDs) from different ontology projects in the methodological guidelines.

11.3.1.8. Perspective further work

We are currently studying the possibility of:

- Including more detail in the tasks about writing and validating CQs and giving priority to CQs.
- Implementing an integrated tool for performing the ontology requirement specification activity.

11.3.2. Supporting Ontology Network Life Cycle Establishment

To build an ontology network, ontology developers should devise first a concrete plan for the ontology network development, that is, they should establish the ontology network life cycle. To do this, ontology developers should answer two key questions: a) which ontology life cycle model is the most appropriate for their ontology project?, and b) which particular processes and activities should be carried out in their ontology life cycle?

11.3.2.1. Overview and objectives

Methodological guidelines to establish the ontology network life cycle, as part of the scheduling activity have been included in the framework of the NeOn Methodology for building ontology networks (Chapter 9). In this section we propose an experiment to learn about the understandability and usability of the preliminary proposed guidelines for helping ontology developers to decide (1) which ontology network life cycle model is the most appropriate for their ontology network and (2) which concrete processes and activities should be carried out in their ontology network life cycle; and at the end, to establish the concrete ontology network life cycle.

The main goal of the experiment is to test the benefits of using the proposed methodological guidelines for obtaining the ontology network life cycle, as part of the scheduling activity.

Software developers and ontology practitioners involved in developing ontologies will obtain a benefit of this experiment that serve us to improve, if necessary, the proposed methodological guidelines and to validate them.

11.3.2.2. Assumptions and user study setup

In this experiment we propose a questionnaire about the methodological guidelines for establishing the ontology network life cycle, in order to be answered by people carry out the experiment.

People carrying out the experiment have different experience level and background in databases, software engineering, etc., but no extensive experience in ontology engineering.

11.3.2.3. Findings and observations

This experiment has been carried out with preliminary methodological guidelines for establishing the ontology network life cycle included in [Suárez-Figueroa et al., 2007]. Figure 60 shows schematically how the establishment of the ontology network life cycle should be carried out according to such proposed methodological guidelines.

The same experiment has been carried out with two set of students as described in Section 11.3.2.4 and Section 11.3.2.5, but the proposed questionnaire was different for each set.

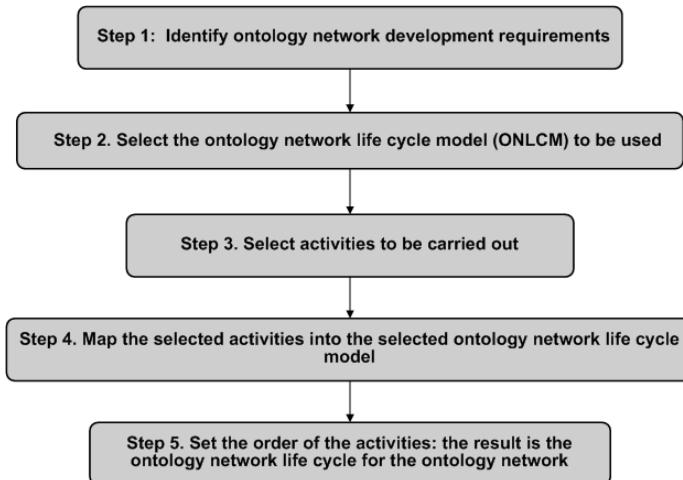


Figure 60. Steps to be carried out for the ontology network life cycle establishment

11.3.2.4. User study 1

In this first study, we carried out the experiment within the “Artificial Intelligence (AI)” master course at Facultad de Informática (Universidad Politécnica de Madrid) with 16 master students, working in groups of two, having background in databases, software engineering, and artificial intelligence, but not in ontology engineering.

The experiment was carried out one time during November 2007.

As mentioned before, this study was performed with the methodological guidelines for establishing the ontology network life cycle included in [Suárez-Figueroa et al., 2007].

The experiment is divided in the following parts:

1. Lectures provide to students the proposed methodological guidelines for establishing the ontology network life cycle, explanations and additional documents (collection of ontology network life cycle models, NeOn Glossary, and Table of “Required-If Applicable” processes and activities).
2. Student groups follow the methodological guidelines to establish the ontology network life cycle for their ontology project.
Students had two weeks for carrying out the experiment using the provided material (methodological guidelines and additional documentation).
3. Student groups document in detail each task performed, using the methodological guidelines, during the establishment of the ontology network life cycle.
4. Students fill the questionnaire included in Annex VI about the proposed methodological guidelines.

11.3.2.5. User study 2

In this second study, we carried out the experiment within the “Artificial Intelligence (AI)” master course at Facultad de Informática (Universidad Politécnica de Madrid) with 8 master students, working in groups of one or two, having background in databases, software engineering, and artificial intelligence, but not in ontology engineering.

The experiment was carried out one time during November 2008.

As mentioned before, this study was performed with the methodological guidelines for establishing the ontology network life cycle included in [Suárez-Figueroa et al., 2007].

The experiment is divided in the same parts as user study 1 (Section 11.3.2.4) with the exception of part 4 in which students fill the questionnaire included in Annex VII.

11.3.2.6. Analysis and discussion

In this section we include the analysis of the two user studies carried out with the methodological guidelines for the ontology network life cycle establishment.

Analysis of user study 1

Here we analyse the questionnaire filled by the 8 groups of master students (composed by 2 people) that carried out the experiment.

Regarding *general issues* we asked about the guidelines in the questionnaire (Annex VI), these are the main conclusions:

- 100% of the students considered that the guidelines were well explained.

General comments regarding this aspect were:

- Guidelines were clear, structured, well written, concise, and understandable.
- Guidelines were in general well explained, but some parts were quite confusing.

- 62% of the students commented that the guidelines needed more detail. Concretely, most of the comments were in the following line.

- Steps 2 and 3 in the guidelines were very well explained. However, steps 4 and 5 were not enough detailed.
- More detail is needed in the NeOn Glossary, because some activities were not well explained and it was easy to mistake one for another.
- Figures in the collection of models (concretely for incremental, iterative and evolutionary models) should be included.
- In general, students commented that more examples should be included in the guidelines.

- 62% of the students considered that the guidelines were complete. But 38% considered that something was missing.

In the later case, general comments were the following:

- Some steps required more details.
- It would be very useful to have the mapping among the activities and the phases in each model.
- More real examples explaining why the different decisions were taken should be included in the guidelines.

Regarding step 2 of the guidelines, the following general comments were provided by the students:

- The decision tree to select the ontology network life cycle model (ontology network life cycle model) was very useful.
- If ontology developer knows software engineering models, then students considered that the collection of ontology network life cycle model was well explained.

In other cases, they mentioned that more details should be included. Concretely, waterfall and its variants and spiral models are considered to be well explained; however the rest of the models in the collection are considered to be not enough detailed.

Students suggested that it would be recommended to include a detailed description of the evolving prototyping model (at least), and a graphical representation of all the models.

With respect to step 3, students provided us with the following comments:

- The set of natural language questions was especially useful, because the activities were not familiar to students and these questions were very intuitive to understand and to select each activity.
- Students considered that the set of natural language questions is very useful to guide naïve users in the activity selection.
- Students commented that the NeOn Glossary was complete, and they found very useful the references of other activities in activity definitions, concretely, useful to distinguish those very similar.
- Additionally, students suggested that it would be useful to present in the NeOn Glossary figures for sub-activities (like in the case of ontology evaluation).

In the case of step 4, most of the students commented that this step was not very useful because it did not provide real guidelines to distribute the activities along the selected model. The mappings among activities and phases in each model were missing and they are crucial.

Regarding step 5, students asked in general for some kind of examples.

Finally, we obtained the following general comments and suggestions given by students:

- To improve the methodological guidelines, (a) steps 4 and 5 should be improved a lot, and (b) more explanations and examples should be included.
- To carry out the establishment of the ontology network life cycle in a faster way, students commented that experience in ontology engineering would be needed.

Additionally, they suggested (a) having an automatic way to map activities with phases in the models, and (b) including the selection of scenarios in the guidelines.

Analysis of user study 2

In this section we analyse the questionnaire filled by the 6 groups of master students (composed by 1 or 2 people) that carried out the experiment.

Regarding *general comments* we asked about the guidelines in the questionnaire (Annex VII), these are the main conclusions:

- 83% of the students considered that the guidelines were well explained.

General comments regarding this aspect were:

- Guidelines were explained with an adequate level of detail.
- Guidelines were concrete, but more details could be included in some steps.

- The order in the guidelines should be improved.
- 67% of the students commented that the guidelines needed more detail. Concretely, most of the remarks were in the following line:
 - More details were required in the description of the ontology activities, because some of them are confusing.
 - More details and examples should be included in the ontology network life cycle models.
 - Complete real examples should be included in the guidelines.

Regarding *step 2* of the guidelines, the following general comments were provided by the students:

- The decision tree to select the ontology network life cycle model was very useful.
- In general, the collection of models is clear and enough explained. It was very useful to have the ontology models related to software engineering ones.
- The different phases of ontology network life cycle models should be explained in more detail.

Waterfall models are enough explained; however, there are no graphics, examples, and detailed explanations for the other models.

- Model descriptions are enough detailed; sometimes too much (common details in similar models should be not repeated)

With respect to *step 3*, students provided us with the following comments:

- The set of natural language questions in combination with the classification of activities in different groups (management, development, support) were useful to select activities.
- The set of natural language questions were a quick reference to select activities; however, sometimes it is confusing because some activities are very similar.
- The NeOn Glossary is in general well explained, but the number of activities is excessive.

More details are required in the following activities: ontology modularization, ontology forward engineering, ontology environment study, ontology elicitation, ontology population, ontology reverse engineering, ontology formalization, and ontology modification.

In the case of *step 4*, most of the students mentioned the following:

- The classification of activities into groups (management, development, support) should be used in this step to facilitate its execution.
- Mappings among activities and phases in each model were missing, and they should be included.
- Real examples should be included in the guidelines.

Regarding *step 5*, the following general comments were provided by the students:

- To establish priorities or restrictions among activities could be useful to order them.
- Real examples to build the Gantt diagram should be included.

Finally, we obtained the following general comments and suggestions given by students:

- To improve the methodological guidelines, it should be useful:
 - To include in which phases we should include each activity.
 - To include complete real examples with explanations about each decision taken in each step.
 - To have a reduced version of the methodological guidelines for a quick overview of them, and to have another version of the guidelines more extended with all the details for consulting and for going in them into depth.
 - To translate the guidelines into other languages (e.g., Spanish).
- To carry out the establishment of the ontology network life cycle in a faster way, students commented that experience in ontology engineering would be needed.

Additionally, they suggested having an integrated automatic assistant, like an expert system, to guide them during the establishment of the ontology network life cycle.

11.3.2.7. Identified strengths and weaknesses

After analysing the results obtained in the two user studies performed with the methodological guidelines for establishing the ontology network life cycle, we can mention as strengths the following ones:

- The methodological guidelines for establishing the ontology network life cycle were clear and understandable according to most of the students.
- The decision tree for selecting the ontology network life cycle model and the set of natural language question to help in the activity selection were considered by the students very useful.
- Regarding the collection of ontology network life cycle models, students stated that it was very valuable to have references to similar models in software engineering.

With respect to the weaknesses, we can comment the following:

- Steps 4 and 5 should be improved, because as they currently are do not provide concrete guidelines to help ontology developers.
- More details and clarifications should be included in some activities of the NeOn Glossary.
- More detail and figures should be included in the collection of ontology network life cycle models.
- Mappings among activities and phases in each ontology network life cycle model should be provided because they are crucial in guidelines to help ontology developers to establish the life cycle.
- Real examples on how to use the proposed methodological guidelines should be included.

11.3.2.8. Improvement Actions Performed

Based on the analysis carried out with the data extracted from the questionnaires of both studies, we already modified and improved the methodological guidelines used in both experiments, obtaining as result the methodological guidelines presented in Figure 61.

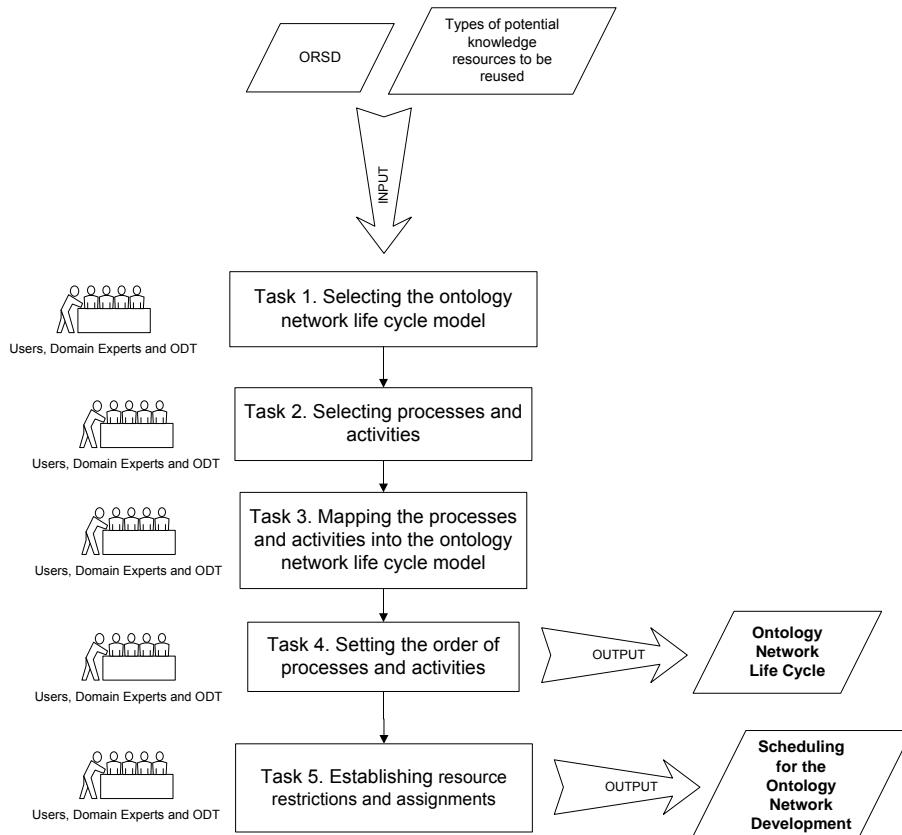


Figure 61. Tasks for scheduling ontology development projects

We implemented the NeOn plug-in gOntt (Section 9.4) that guides ontology developers during the establishment of the ontology network life cycle. We based on the methodological guidelines presented in Figure 61 and on the following updates:

- Revision and update of the collection of ontology network life cycle models (Chapter 7).
- Inclusion of natural language questions to select the ontology network life cycle model.
- Inclusion of natural language questions, referred to the nine identified scenarios in the NeOn Methodology, to decide among the different versions of a particular model.
- Inclusion of natural language questions, also related to scenarios, to select which processes and activities should be included in the life cycle.
- Incorporation of the mappings among processes and activities and phases in each ontology network life cycle model.

11.3.2.9. Perspective further work

Additionally, based on the analysis carried out, we are currently working on:

- Including more details and clarifications in some activities of the NeOn Glossary.
- Including real examples on how to use the proposed methodological guidelines.
- Improving Task 4 in Figure 61.

11.3.3. Scheduling Ontology Development with gOntt

11.3.3.1. Overview and objectives

gOntt (described in Section 9.4) is a plug-in to schedule ontology development projects and to help developers during the project execution.

In this section we propose a preliminary experiment to learn about the understandability and usability of the gOntt plug-in.

The main goal of the experiment was to test that the students (1) scheduled their ontology development project with gOntt, (2) followed the scheduling with the methodological guidelines provided by gOntt and (3) used the necessary NeOn Toolkit plug-ins for every process and activity. The plug-ins could be triggered by gOntt in the right moment of the schedule.

11.3.3.2. Assumptions and user study setup

In this experiment we propose a questionnaire that should be filled by the students at the end of the experiment. Such a questionnaire was divided in two main parts: (1) one related to background knowledge, and (2) the other related to the experiences gained by using gOntt. The second part included 10 questions with fixed answers and an open question in which they were asked to express other comments.

People carrying out the experiment have different experience level and background in databases, software engineering, etc., but no extensive experience in ontology engineering.

11.3.3.3. User study 1

We performed the experiment using gOntt with two settings that involved Master students and PhD students. Both groups of students had taken the Ontology Engineering and Semantic Web subject. The students executed the experiment during the period of October-November 2009.

The students (working in pairs) had to develop an ontology network in a free domain. They were provided with a set of mandatory activities to be carried out (e.g., to specify requirements and to reuse different types of knowledge resources) and a set of optional activities (e.g., localization, evaluation, and modularization).

The experiment is divided in the following parts:

1. Lectures provide to students the NeOn Methodology for developing ontology networks.
2. Student groups follow the NeOn Methodology to develop an ontology network in a free domain.
3. Student groups use the gOntt plug-ing both for scheduling and for executing the project. Students had two weeks for carrying out the experiment using the provided material.
4. Student groups document in detail each task performed.
5. Students fill the questionnaire included in Annex VII.

11.3.3.4. Analysis and discussion

In this section we include the analysis of the user study carried out with the gOntt plug-in. In order to analyse the students' perception of gOntt, we selected three questions from the questionnaire in Annex VIII. The answers are displayed in Table 30 and Table 31.

Question	Distribution of answers		
	Yes	No	Other
Is gOntt useful for scheduling ontology development projects?	14	1	1

Table 30. Answer to the question selected

Question	Distribution of answers			
	A good and useful idea?	Just a nice functionality?	A useless idea?	I don't know
Is the idea of having methodological guidelines in gOntt	10	6	0	0
Is the idea of triggering NeOn Toolkit plug-ins from gOntt	6	4	0	6

Table 31. Answers to the questions selected

Analyzing the answers in both tables, we can say that

- The students perceived gOntt as a useful scheduling tool.
- The students in general perceived as a positive and useful issue the fact that gOntt provided them with methodological guidelines for each process and activity involved in the ontology development.
- With regard to the functionality of triggering NeOn Toolkit plug-ins, the opinion was divided, but it tended to consider it as a good idea.

Additionally, the students provided general comments that can be summarized as follows:

- gOntt is considered as a good scheduling tool that is based on activities and processes of the Ontology Engineering field.
- gOntt is considered as a good help in providing a preliminary schedule for a development project.
- gOntt is considered one of the easiest tool for creating Gantt charts, thanks to the simple wizards that guide the user.
- gOntt is seen as a centralized environment that allows the user to obtain methodological guides without searching in external locations.
- gOntt is considered a useful and didactic tool.

- gOntt provides the means for both structuring and centralizing the management of the development.

The students also commented that the tool established at random the length of the processes and activities planned and not based on previous experiences. They also missed the functionality of exporting the schedule to Microsoft Project.

11.4. Conclusions

In this chapter we described a set of experiments carried out to verify the results presented in this thesis. Such experiments allowed us to verify the hypotheses of this thesis in the following way:

First we have shown that the set of two models provided in this thesis is enough to allow ontology practitioners to developed ontology networks (Hypothesis 2).

We have also shown that the ontology network development in different contexts is based on combining different scenarios (Hypothesis 3). Such scenarios mainly involve the reuse of ontological and non-ontological resources, the reengineering, the merging, and the localization.

Both in action research experiments and in controlled ones, we verified that the methodological guidelines for specifying ontology requirements can be applied by ontology practitioners (Hypothesis 4). Additionally, we verified that the support given to schedule ontology development projects is simple, clear, and can be applied by ontology practitioners (Hypothesis 5).

Finally, we verified in action research experiments that the guidelines for reusing ontological resources are useful for ontology practitioners (Hypothesis 6).

In short, Table 32 summarized how the different experiments presented in this chapter are related to the hypothesis identified in Chapter 3.

Action Research		
<i>Experiments on Scenarios for Building Ontology Networks</i>	<i>Experiments on Life Cycle Models and Life Cycle Establishment</i>	<i>Experiments on Requirements Specification and Reuse</i>
Hypothesis 3	Hypothesis 2 Hypothesis 5	Hypothesis 4 Hypothesis 6
Controlled Experiments		
<i>Experiments for the Ontology Requirements Specification Activity</i>	<i>Supporting Ontology Network Life Cycle Establishment</i>	<i>Scheduling Ontology Development with gOntt</i>
Hypothesis 4	Hypothesis 2 Hypothesis 5	Hypothesis 5

Table 32. Experiments and the related hypothesis

12. Conclusions and Future Work

As stated in Chapter 3, one of the goals of this thesis is to advance the current state of the art in the Ontology Engineering field, and specifically in the methodological area. Another goal is to create a ***methodology for building ontology networks*** by (1) providing the identification and definition of the development process and life cycle for networks of ontologies, (2) proposing the methodological framework, based on scenarios for building ontology networks, and (3) proposing detailed methodological guidelines for ontology requirements specification, scheduling, and ontological resource reuse.

Having in mind these goals, in this chapter we present how the open research problems identified in Chapter 3 are solved by the main thesis contributions.

- A first contribution has been the **NeOn Glossary of Processes and Activities** that tries to solve the problem of “*the lack of consensus and standardization about the definitions of processes and activities that potentially could be carried out when single ontologies and ontology networks are developed*”.

As mentioned in Chapter 5 ontology experts had already a certain implicit consensus on what activities should be undertaken in ontology development. This implicit consensus was made explicit by means of a glossary of processes and activities in the Ontology Engineering field, as shown in Chapter 5. The glossary identifies and defines the processes and activities potentially involved in the ontology network construction. Contrary to the Software Engineering field, which boasts the IEEE Standard Glossary of Software Engineering Terminology [IEEE, 1990], the Ontology Engineering Field does not have a standard glossary, and this thesis is a first step to bridge this gap.

As a result of this first contribution, we have the following publications:

- M. C. Suárez-Figueroa, A. Gómez-Pérez. “*First Attempt towards a Standard Glossary of Ontology Engineering Terminology*”. The 8th International Conference on Terminology and Knowledge Engineering. Managing ontologies and lexical resources. TKE 2008 8th International Conference on Terminology and KE. ISBN: 87-91242-50-9. Pages: 1-15. Copenhagen, DENMARK. August 18-August 21, 2008.
- M. C. Suárez-Figueroa, A. Gómez-Pérez. “*Towards a Glossary of Activities in the Ontology Engineering Field*”. The 6th Language Resources and Evaluation Conference. 6th Conference on Language Resources and Evaluation. ISBN: 2-9517408-4-0. Marrakech, MOROCCO. May 28-May 30, 2008.

- A second contribution is the proposal of a **set of 9 flexible scenarios for building ontologies and ontology networks** in the framework of the NeOn Methodology, as shown in Chapter 6. This proposal tries to solve the problem of “*the lack of a methodology that covers complex scenarios in which reuse and reengineering of ontological and non-ontological resources are needed*”.

Following the new trend in ontology development, which emphasizes (1) the reuse and possible subsequent reengineering of knowledge resources (ontologies, ontology modules, ontology statements and ontology design patterns as well as non-ontological resources such as thesauri, lexicons, DBs, UML diagrams and classification schemas built by others that already have some degree of consensus) to speed up the ontology development, (2) the collaborative and argumentative ontology development, and (3) the building of ontology networks, as opposed to custom-building new ontologies from scratch, we can state that ontology networks can be developed by combining different scenarios that mainly involve reuse of ontological and non-ontological resources, reengineering, merging, restructuration, and localization. Thus, we have identified a set of 9 flexible scenarios for building ontologies and

ontology networks in the framework of the NeOn Methodology as shown in Chapter 6. The scenarios proposed are flexible because they can be combined among them, in contrast to the rigid scenario encountered when building ontologies from scratch and presented in METHONTOLOGY, On-To-Knowledge and DILIGENT. This set of scenarios is the basis of the NeOn Methodology, presented in this thesis.

The NeOn Methodology is a methodology that defines each process or activity in a precise manner by stating clearly its purpose, inputs and outputs, the actors involved, when its execution is more convenient, and a set of proposed methods, techniques and tools to be used. This methodology tries to solve the problems of (a) "*the lack of any methodology targeted to software developers and ontology practitioners, instead of targeted to ontology researchers*" and of (b) "*the lack of any methodology explaining the ontology building process with the same style and granularity than those methodologies for developing software*". The Neon Methodology is presented in a prescriptive way and non-oriented to researchers.

This thesis presents four different experiments that show how the different scenarios identified in the framework of the NeOn Methodology have been used in several cases (invoicing management use case, semantic nomenclature use case, SEEMP use case, and GeoBuddies use case). The experiments are described in Section 11.2.1.

This contribution has originated the following publications:

- M.C. Suárez-Figueroa, A. Gómez-Pérez. "NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology". International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES (S3T 2009). Proceedings of the International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES (S3T 2009). Sofia, Bulgaria. 28-29 October 2009.
 - A. Gómez-Pérez, M. C. Suárez-Figueroa. "Scenarios for Building Ontology Networks within the NeOn Methodology". Fifth International Conference on Knowledge Capture (K-CAP 2009). Poster in Proceedings of the Fifth International Conference on Knowledge Capture (K-CAP 2009) (ISBN: 978-1-60558-658-8). Pages: 183-184. Redondo Beach, California, USA. 1st – 4th September 2009.
 - A. Gómez-Pérez, M. C. Suárez-Figueroa. "NeOn Methodology: Scenarios for Building Networks of Ontologies". 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008). Poster in Proceedings of 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008). Acitrezza, Catania, ITALY. September 29-October 3, 2008.
 - A third contribution is the proposal of **two ontology network life cycle models**, the waterfall model and the iterative-incremental model. This proposal tries to solve the problem of "*having available only one ontology life cycle model (evolving prototypes), in contrast to the set of software life cycle models*".
- As mentioned in Chapter 7, these models have been created following those defined in the Software Engineering field and taking into account the specific features of the ontology network development. Such models are used in the NeOn Methodology for establishing the ontology life cycle and thus the scheduling of ontology development projects as explained in Chapter 9.
- These two models have been evaluated in two experiments, in which ontology developers from two of the NeOn use cases (invoicing management use case and semantic nomenclature use case) were involved, as presented in Section 11.2.2. The ontology developers selected the appropriate life cycle model for their needs following the guidelines presented in this thesis.

This contribution has originated the following publication:

- M. C. Suárez-Figueroa, A. Gómez-Pérez. “*Building Ontology Networks: How to Obtain a Particular Ontology Network Life Cycle?*”. International Conference on Semantic Systems (I-SEMANTICS’08). Proceedings of I-SEMANTICS 2008. ISSN: 0948-695x. Online edition: ISSN 0948-6968. Pages: 142-149. Graz, AUSTRIA. September 3-5, 2008.
- A fourth contribution is the proposal of **methodological guidelines for specifying ontology requirements**. These guidelines try to solve the problem of “*not having detailed prescriptive guidelines to carry out the ontology requirements specification activity*”.

One of the critical activities in the ontology development is to identify the functional and non-functional ontology requirements. In this thesis, concretely in Chapter 8, we have presented detailed and prescriptive methodological guidelines for specifying ontology requirements. The guidelines are based on the so-called competency questions (CQs) and on a template for writing the ontology requirement specification document (ORSD).

The ORSD plays a key role during the ontology development process because it facilitates different activities. In the context of this PhD Thesis, the ontology requirements specification document is a crucial input for the scheduling of ontology development projects and for the search and reuse of ontological resources (ontologies and ontology statements).

As in other disciplines, the requirements specification (a) establishes the basis for the agreement between users and ontology developers, (b) reduces the development effort, (c) provides a basis for estimating costs and schedules, and (d) offers a baseline for verification.

We conducted an experiment to test the methodological guidelines proposed in this thesis for the ontology requirements specification activity. The experiment, described in Section 11.3.1, has as a main goal to learn about the understandability and usability of the proposed methodological guidelines.

This contribution has originated the following publication:

- M.C. Suárez-Figueroa, A. Gómez-Pérez, Boris Villazón-Terrazas. “*How to write and use the Ontology Requirements Specification Document*”. The 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2009). Proceedings of the 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2009). Vilamoura, Algarve-Portugal. 3-5 November 2009.
- A fifth contribution is the proposal of the **methodological guidelines for scheduling ontology development projects**. With these guidelines we intend to solve the problem of “*the lack of detailed prescriptive guidelines to select a specific life cycle model to create a particular ontology life cycle, as part of the scheduling activity*”.

Planning and scheduling activities are extremely important in order to set order and time in the processes and activities involved in the ontology network development. In this regard, we provide in Chapter 9 (a) the methodological basis for establishing the concrete life cycle for an ontology network as part of the scheduling activity; these groundings are based on the NeOn Glossary of Processes and Activities (Chapter 5), on the set of scenarios identified in the NeOn Methodology (Chapter 6), and on the set of life cycle models (Chapter 7), (b) a tool called gOntt that supports the scheduling and execution of ontology networks development projects, and (c) the methodological guidelines for scheduling ontology development projects using gOntt. gOntt supports the automatic generation of the initial ontology development plan in the form of a Gantt chart. Furthermore, gOntt is considered the first meta-tool that helps ontology developers to execute ontology development projects. It includes prescriptive methodological guidelines and informs about the recommended tool to be used for performing a particular process or activity in an ontology development project.

This thesis includes an experiment whose aim is to learn about the understandability and usability of the preliminary guidelines proposed for helping software developers and ontology practitioners to decide (1) which ontology network life cycle model is the most appropriate for their ontology network and (2) which specific processes and activities should be carried out in their ontology network life cycle and how to establish the concrete ontology network life cycle. This experiment is described in Section 11.3.2. Additionally, Section 11.3.3 presents a preliminary experiment with the purpose of learning about the understandability and usability of the gOntt plug-in.

This contribution has originated the following publications:

- M.C. Suárez-Figueroa, A. Gómez-Pérez, O. Muñoz, M. Vigo. “*gOntt, a Tool for Scheduling and Executing Ontology Development Projects*”. The 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010). San Francisco Bay, USA. July 1 - July 3, 2010.
- A. Gómez-Pérez, M.C. Suárez-Figueroa, M. Vigo. “*gOntt: a Tool for Scheduling Ontology Development Projects*”. The 8th International Semantic Web Conference (ISWC 2009). Demo paper in Proceedings of the 8th International Semantic Web Conference (ISWC 2009). Washington, DC. 25-29 October 2009.
- The last contribution of this thesis is the proposal of **methodological guidelines for reusing ontological resources**. This proposal tries to solve “*the lack of detailed prescriptive guidelines to reuse ontological resources at different levels of granularity (as a whole, by modules, or by statements)*”.

One way of reducing time and also costs associated to the ontology development is by reusing available ontological resources. The reuse of (well-developed) ontological resources also allows spreading good practices and increasing the overall quality of ontological models. In this regard, Chapter 1 presents (a) the methodological guidelines for reusing general ontologies together with documented knowledge about two different theories (time and mereology), (b) the methodological guidelines for reusing domain ontologies, and (c) the methodological guidelines for reusing ontology statements.

In this thesis we briefly describe how ontology developers reused ontological resources using the ontology requirements specification document (ORSD). This experiment is presented in Section 11.2.3.

This contribution has originated the following publication:

- M. Fernández-López, A. Gómez-Pérez, M. C. Suárez-Figueroa. “*Selecting and Customizing a Mereology Ontology for its Reuse in a Pharmaceutical Product Ontology*”. The 5th International Conference on Formal Ontologies in Information Systems (FOIS 2008). In C. Eschenbach and M. Grüninger (Eds.). 2008. Formal Ontologies in Information Systems. IOS Press. doi: 10.3233/978-1-58603-923-3-181. Saarbrücken, GERMANY. October 31-November 3, 2008.

In a nutshell, Figure 62 shows a summary of the main results of this thesis.

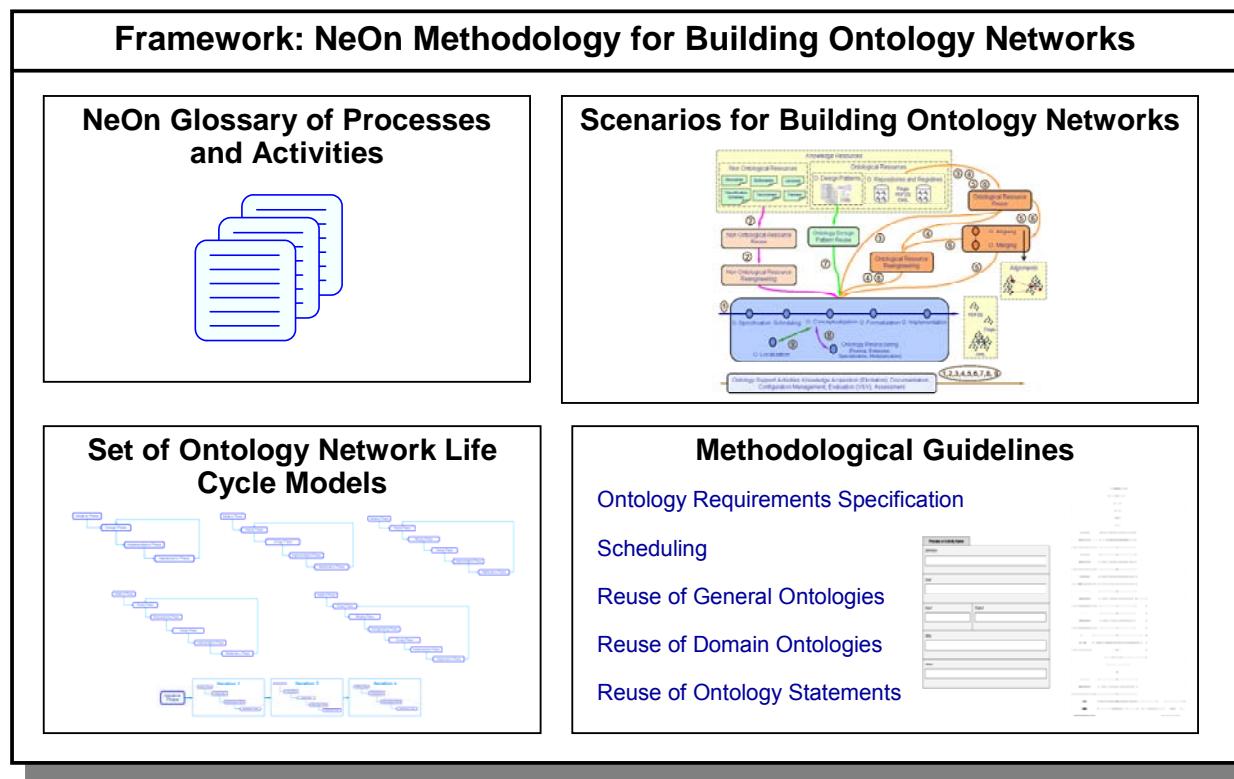


Figure 62. Overview of the results presented in this thesis

Complementary to the processes and activities described in the thesis, we should mention that some methodological aspects not covered here have been or are being dealt with by three members of the OEG group. These aspects are the following:

- ❑ Collaboration and dynamism [Palma, 2009]
- ❑ Reuse and reengineering non-ontological resources [Villazón-Terrazas, in progress]
- ❑ Localization [Espinoza, in progress]

To conclude, Table 33 shows the comparison between the three methodologies (METHONTOLOGY, On-To-Knowledge, and DILIGENT) and the NeOn Methodology for building ontology networks regarding the following characteristics: (1) collaboration; (2) degree of coverage of the process or activities by means of providing detailed guidelines; and (3) methodology audience, which were analysed within the state of the art chapter.

	METHONTOLOGY	On-To-Knowledge	DILIGENT	NeOn Methodology
Dimension				
Collaboration	Not mentioned	Not mentioned	Treated	<i>Mentioned, but not treated in detail</i>
Detailed Guidelines for Processes and Activities				
Ontology Requirements Specification	Not provided Only Competency Questions are proposed	Not provided Only Competency Questions are proposed	This activity is not proposed by the methodology	<i>Provided</i>
Scheduling	Not provided	Not provided	Not provided	<i>Provided</i>
Reusing Ontological Resources	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 for ontologies and ontology statements</i>
Audience				
Targeted to Software Developers and Ontology Practitioners	Targeted to ontology engineers and researchers	Targeted to ontology engineers and researchers	Intended to domain experts and users	<i>Targeted to ontology practitioners, general public, and software engineers</i>

Table 33. Comparative analysis of the three methodologies analysed and the NeOn Methodology

Going back to the set of requirements presented in Chapter 4, we show here how the NeOn Methodology satisfies such requirements. This set of requirements is divided into necessary requirements and sufficient requirements.

With respect to *necessary requirements*, we have

- **Generality.** The NeOn Methodology treats the development of ontology networks in general by proposing different scenarios for building networks of ontologies, without being driven to particular domains or cases.
- **Completeness.** The NeOn Methodology for building ontology networks deals with 9 scenarios, described in Chapter 6, which consider all the possible cases. This thesis deals with a subset of the possible cases that are part of Scenario 1 and Scenario 3.
- **Effectiveness.** The NeOn Methodology is described in a simple and a prescriptive way, and any person (a software developer or an ontology practitioner) is able to understand and use it with no special effort, as shown in Chapter 11.
- **Efficiency.** A methodology must be efficient, that is, be able to achieve its objective or goal. This means that the methodology should allow the construction of ontology networks.

In our case, whenever it is possible, we describe and conduct experiments with the NeOn Methodology with the purpose of confirming its efficiency, as described in Chapter 11.

- **Consistency.** The NeOn Methodology identifies clearly which are the outputs of the different processes and activities involved in the development of ontology networks. Thus, the same set of outputs will be obtained after applying the NeOn Methodology for a given case.

- **Finiteness.** The NeOn Glossary of Processes and Activities presented in Section 5.3 includes a finite set of processes and activities involved in the NeOn Methodology. The number of elements used to describe the process and the activities are also finite.
- **Discernment.** A methodology must be composed of a set of structural, functional and representational components. This set should be as small as possible.
 - Regarding structural components, the NeOn Methodology provides a set of 9 scenarios for building ontology networks. These scenarios are a combination of heterarchical and hierarchical structures.
 - Regarding functional components, the NeOn Methodology includes processes, activities, tasks, inputs, outputs, and restrictions.
 - Finally, with respect to representational components, the NeOn Methodology provides a graphical representation of the scenarios and of each process or activity.
- **Environment.** The NeOn Methodology for building ontology networks can be considered as a technological methodology because the main result after applying it should be a technological product, that is, an ontology network. Thus, we need to establish the life cycle for the ontology network. Guidelines for establishing a particular ontology network life cycle are included in Chapter 9.
- **Transparency.** The NeOn Methodology explicitly defines the actors, inputs, and outputs of each process or activity covered by the methodology.
- **Essential Questions.** The following six questions: “what”, “who”, “why”, “when”, “where”, and “how” must be considered for each process or activity included in the methodology.

The question about “where” is not covered in the NeOn Methodology. However, the other questions are responded in detail in the NeOn Methodology: the NeOn Glossary of Processes and Activities explains “what” each process and activity involved in the methodology refers to, and the methodological guidelines for the processes or activities included in this thesis answer the other questions.

With respect to *sufficient requirements*, we have:

- **Domain or Scope.** The NeOn Methodology has been created for developing ontology networks, putting special emphasis on the existence of multiple ontologies in ontology networks, the reuse of knowledge resources, the dynamic dimension that treats the ontology evolution, and the collaborative ontology development¹⁰⁷.
- **Perspectives.** The NeOn Methodology provides 9 different scenarios for building ontology networks and allows combining them in different ways, which permits applying the methodology following different approaches.
- **Understanding.** The NeOn Methodology for building ontology networks is explained with simple descriptions and graphical representations, and thus it is easily understood by both ontology practitioners and software developers, as shown in Chapter 11.
- **Usability.** The NeOn Methodology has a Software Engineering approach with different levels of complexity to facilitate a promptly assimilation, as shown in Chapter 11.
- **Grounded on existing practices.** The NeOn Methodology grounds on methodologies in Software and Ontology Engineering fields and on experience in ontology development.
- **Flexibility.** The NeOn Methodology allows its adaptation to specific needs and users; it also allows the inclusion of new processes or activities involved in the development of ontology networks.

¹⁰⁷ The dynamic dimension and the collaborative dimension are out of the scope of this thesis.

- **Tool-Independent.** The NeOn Methodology is independent of the technology. The methodology has a close relation with the NeOn Toolkit and its plug-ins, but it can also be used with other tools, such as Protégé, Top Braid Composer, etc.

Finally, it is worth mentioning that the hypotheses 2, 3, 4, 5, and 6, established in Chapter 3, have been proved in Chapter 11.

For the **current and future work** in the NeOn Methodology framework, we would like to present the following lines of research regarding the activities and processes treated in this thesis:

- Regarding the *ontology requirements specification activity*, we are working on a more flexible way to identify the requirements (in the form both of competency questions and of natural language sentences). Additionally, we are investigating collaboration techniques to exchange requirements and information among the actors involved in identifying the requirements.

We are also investigating in depth what kind of non-functional requirements can be established according to the type of ontology to be developed to create a catalog of this type of requirements, which would facilitate the identification of them by ontology practitioners. In this sense, we propose the inclusion of general non-functional requirements, such as the naming conventions to be used in the ontology network development.

We are also trying to develop different methods to write competency questions (and their responses). Methods could be based on simple grammar and lexico syntactic patterns [Aguado et al., 2008].

We also plan to analyse different techniques, such as clustering natural language sentences or information extraction, to perform automatically the grouping of requirements.

We are investigating methods to validate the requirements and the way to transform (semi)automatically functional requirements in SPARQL queries, which can serve as unit test to verify the ontology network.

Finally, we are studying the possibility of implementing an integrated tool for performing the ontology requirements specification activity.

- Regarding the *scheduling activity*, we plan to integrate the methodological guidelines proposed and gOntt with the works done by the ONTOCOM team for predicting the total costs of the ontology development project and the estimated duration for each process and activity scheduled.

We intend to analyse which are the processes and activities as well as the gOntt functionalities more used in order to create different use profiles and to customize gOntt according to such profiles.

We want to investigate the use of gOntt for scheduling and executing other types of development projects (such as the development of semantic applications).

- With regard to the *reuse of ontological resources*, the missing issue here is the creation of methodological guidelines for reusing only one part or *module* of an ontology relevant to the ontology development. Thus, we are working on providing methodological guidelines for reusing ontology modules.

Additionally, we are investigating the improvement of the methodological guidelines provided in this thesis. Specifically, we are working on refining and extending the criteria proposed for assessing ontology statements.

We are analysing the inclusion of the methodological guidelines for reusing ontology statements on tools, such as Watson.

We also plan to analyse in depth the cost of reusing knowledge resources in general, and ontological resources in particular.

Furthermore, we intend to study some lines of research related to other processes and activities not dealt with until now. These lines of research are

- The *control activity*, which refers to the activity of guaranteeing that the activities scheduled in the ontology development process are completed and performed in the fashion intended. One of the tasks to be carried out in this activity is that of *ontology requirements management*, which involves performing the control of the requirements for their exploitation and evolution along the development of the ontology network. As part of this requirements management, the requirements traceability should be carried out. This traceability mainly refers to the identification of which requirements affect and/or are satisfied by each one of the outputs of the different processes and activities in the development. Currently, the NeOn Methodology does not provide any methodological guidelines to help ontology developers in the control activity.

Thus, we are investigating on methodological guidelines in order to maintain the traceability between requirements and (a) subdomains, (b) ontology models, and (c) tests to verify the ontology network. Such guidelines should be implemented in a tool that allows ontology developers to maintain the traceability of the ontology requirements.

We are analysing how the traceability of the requirements should affect the evolution and versioning of the ontology network.

- The *ontology design pattern reuse* process, which refers to the process of using available ontology design patterns for solving different modelling problems during the development of new ontologies.

Ontology design patterns are perceived as an aid to modelling ontologies, a development guide, and a way to improve the quality of the resulting ontologies. Therefore, we are investigating the improvement of existing patterns as well as the proposal of new patterns to solve typical modelling problems.

We are investigating on methodological guidelines to identify straightaway which ontology design patterns should be reused according to the competency questions.

- The *ontology evaluation activity*, which refers to the activity of checking the technical quality of an ontology against a frame of reference.

One of its subactivities is the *ontology verification activity*, which is the ontology evaluation that compares the ontology against the ontology requirements specification document (ontology requirements and competency questions), thus ensuring that the ontology is built correctly (in compliance with the ontology requirement specification). Currently, the NeOn Methodology does not provide any methodological guidelines to help ontology developers in the verification activity.

We are investigating on methodological guidelines to verify the ontology by following a set of tests obtained from the ontology requirements and some guidelines to document those tests taking into account different features of the requirements (e.g., priority, complexity, cost, etc.).

Besides, we are investigating both the possibility of transforming (semi)automatically the competency questions in SPARQL queries and the use of such queries in a tool for verifying the ontology network. The (semi)automatic transformation could be related to the lexico syntactic patterns.

Other evaluation subactivity is the *ontology validation*, which is the ontology evaluation that compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize. In this regard, we plan to investigate whether in some cases linguistic and/or syntactic information can be used in order to decide if the knowledge represented in the ontology is correct.

Additionally, we are investigating pattern-based methods to evaluate ontologies. In this regard, we are studying different errors included in ontology modelling. These errors can be classified into two types: (1) errors associated to ODPs, named as antipatterns and (2) errors not

associated to ODPs, named as pitfalls [Poveda et al., 2009]. Our plan is to provide ontology developers with methodological guidelines to avoid such errors as well as to evaluate their ontologies.

- When developing ontology networks by reusing knowledge resources, as proposed in the NeOn Methodology, ontology developers can find diverse situations that may involve taking different decisions. In general, ontology developers should have to face questions similar to the following ones: (a) how to select a resource among different ones that model similar knowledge?; (b) how to extend a particular knowledge resource?; (c) how to combine different knowledge resources?; and (d) how to add a new ontological resource to an ontology network?. Currently, the NeOn Methodology does not provide prescriptive methodological guidelines to help ontology developers to respond these questions.

Therefore, we are investigating on methodological guidelines for selecting, comparing and combining non-ontological resources, ontological resources, and ontology design patterns with the aim of building ontology networks. We are analysing the main steps to be performed in order to integrate different knowledge resources, taking into account the interaction among the different methods proposed for reusing the different knowledge resources.

We are also analyzing the possibility of using (a) the future profiles (processes and activities more frequently used) obtained from gOntt in order to decide which resource to be reused in case of having different types that model similar knowledge and (b) the estimated effort associated to each of the possible reuses. In this sense, a good idea would be to have a tool that provides the possibility of showing the expected results according to the different possibilities in the reuse.

- *Conceptualizing or modelling* ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. In recent years, the emergence of ontology design patterns has supposed a great help to ontology developers when modelling ontologies. However, in some cases, ontology developers experience difficulties when reusing the patterns during modelling, and because of these difficulties they include errors in the modelling. Currently, the NeOn Methodology does not provide methodological guidelines to help ontology developers to prevent the appearance of such errors.

For this reason, we are working on the creation of methodological guidelines in order to prevent the inclusion of errors during the ontology modelling. These guidelines are mainly based on the identification and classification of modelling errors. We classify errors into two types: (1) errors related to ODPs, called anti-patterns; and (2) errors not related to ODPs, called pitfalls.

Finally, it is worth mentioning that we plan to improve and maintain the *NeOn Glossary of Processes and Activities* and to continue with the process of standardizing such a glossary.

References

- [Aguado et al., 2009] G. Aguado de Cea, A. Gómez-Pérez, E. Montiel-Ponsoda, M.C. Suárez-Figueroa. *Using Linguistic Patterns to Enhance Ontology Development*. Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2009). (ISBN: 978-989-674-012-2). Pages: 206-213. Madeira, Portugal. 6-8 October 2009.
- [Aguado et al., 2008] G. Aguado de Cea, A. Gómez-Pérez, E. Montiel-Ponsoda, M.C. Suárez-Figueroa. *Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns*. Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008. ISBN: 978-3-540-87695-3. Pages: 32-47. Acitrezza, Italy, September 29 - October 2, 2008.
- [Allen, 1984] J. F. Allen. *Towards a General Theory of Actions and Time*. Artificial Intelligence 23:123-154, 1984.
- [Alonso et al., 1996] F. Alonso, N. Juristo Juzgado, J. L. Maté, J. Pazos. *Software engineering and knowledge engineering: Towards a common life cycle*. Journal of Systems and Software 33(1): 65-79. 1996.
- [Alonso et al., 1995] F. Alonso, N. Juristo, J. Pazos. *Trends in life-cycle models for SE and KE: Proposal for a spiral-conical life-cycle approach*. 1995. International Journal of Software Engineering and Knowledge Engineering, Vol. 5, No. 3 (1995) 445-465.
- [Arpírez et al., 2003] J. C. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez. *WebODE in a nutshell*. AI Magazine. 2003.
- [Baeza-Yates, 1999] R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley. 1999. ISBN 0-201-39829.
- [Beck and Andres, 2004] K. Beck, C. Andres. *Extreme Programming Explained: Embrace Change* (2nd Edition), 2004. Addison-Wesley, Boston.
- [Bettini et al., 1996] C. Bettini, X. S. Wang, S. Jajodia. *A General Framework and Reasoning Models for Time Granularity*. In L. Chittaro, S. D. Goodwin, H. J. Hamilton, A. Montanari (editors), Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME'96), Key West, Florida, 1996.
- [Blázquez et al., 1998] M. Blázquez, M. Fernández-López, J. M. García-Pinar, A. Gómez-Pérez. *Building Ontologies at the Knowledge Level using the Ontology Design Environment*. In: Gaines BR, Musen MA (eds) 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW 1998). Banff, Canada, SHARE4:1–15.

- [Blomqvist et al., 2009] E. Blomqvist, A. Gangemi, V. Presutti. *Experiments on pattern-based ontology design*. Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA. ISBN: 978-1-60558-658-8.
- [Boehm, 1988] B. W. Boehm. *A spiral model of software development and enhancement*. ACM SIGSOFT Software Engineering Notes. Vol. 11, no. 4, pp. 14-24, August 1986; reprinted in Computer, vol. 21. no. 5, pp. 61-72, May 1988.
- [Boehm, 1976] B. W. Boehm. *Software engineering*. IEEE Transactions on Computers. Vol. C-25, pp. 1226-1241, December. 1976.
- [Borgo et al., 1997] 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.
- [Borgo et al., 1996] 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.
- [Borst, 1997] W. N. Borst. *Construction of Engineering Ontologies*. Centre for Telematica and Information Technology, University of Twente. Enschede, The Netherlands. 1997.
- [Brickley and Guha, 2004] 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.
- [Buzan, 1974] T. Buzan. *Use your head*. BBC Books. 1974.
- [Byrne, 1992] 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.
- [Casati et al., 1998] 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.
- [Casati and Varzi, 1995] R. Casati, A. Varzi. *Holes and other superficialities*. MIT Press, Cambridge, Massachusetts. 1995.
- [Cockburn and Williams, 2000] A. Cockburn, L. Williams. *The Costs and Benefits of Pair Programming*. (2000). Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2000).
- [Cohen et al., 2003] W. Cohen, P. Ravikumar, S. Fienberg. *A Comparison of String Distance Metrics for Name-Matching Tasks*. Proceedings of IIWEB Workshop at the IJCAI03, 2003.

- [Coté et al., 1994] R. A. Coté, D. J. Rothwell, L. Brochu, eds. *SNOMED International* (3rd ed.), Northfield, Ill, College of American Pathologists, 1994.
- [Cuenca-Grau et al., 2007] B. Cuenca-Grau, I. Horrocks, Y. Kazakov, U. Sattler. *Just the right amount: extracting modules from ontologies*. Proceedings of the 16th International Conference on World Wide Web. Banff, Alberta, Canada. Pages: 717 - 726. ISBN: 978-1-59593-654-7. 2007.
- [Davis, 1993] A. Davis. *Software Requirements: Objects, Functions and States*. Upper Saddle River, New Jersey: Prentice Hall, 1993.
- [Davis et al., 1988] A. M. Davis, E. H. Bersoff, E. R. Comer. *A Strategy for Comparing Alternative Software Development Life Cycle Models*. IEEE Transactions on Software Engineering 14-10. 1453-1461. 1988.
- [d'Aquin et al., 2007] M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, E. Motta. *Watson: A Gateway for Next Generation Semantic Web Applications*. Poster session of the International Semantic Web Conference, ISWC 2007.
- [d'Aquin et al., 2007b] M. d'Aquin, M. Sabou, M. Dzbor, C. Baldassarre, L. Gridinoc, S. Angeletou, E. Motta. *Watson: A Gateway for the Semantic Web*. Poster session of the European Semantic Web Conference, ESWC 2007.
- [d'Aquin et al., 2007c] 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.
- [d'Aquin et al., 2007d] M. d'Aquin, L. Gridinoc, S. Angeletou, M. Sabou, E. Motta. *Characterizing Knowledge on the Semantic Web with Watson*. Workshop: Evaluation of Ontologies and Ontology-based tools, 5th International EON Workshop (EON 2007), International Semantic Web Conference (ISWC'07), Busan, Korea.
- [de Hoog, 1998] 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.
- [Dellschaft et al., 2008] 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. <http://www.uni-koblenz.de/~klaasd/Downloads/papers/Dellschaft2008CTD.pdf>.
- [Dellschaft et al., 2008b] K. Dellschaft, A. Gangemi, J. M. Gómez, H. Lewen, V. Presutti, M. Sini. *Neon Deliverable D231. Practical Methods to Support Collaborative*. NeOn Project. <http://www.neon-project.org>. February 2008.
- [Ding and Fensel, 2001] Y. Ding, D. Fensel. *Ontology library systems: The key to successful ontology reuse*. In Proceedings of The first Semantic Web Working Symposium (SWWS'01), pages 93-112, Stanford University, California, USA, July 30 - August 1. 2001.

- [Dzbor et al., 2009] M. Dzbor, M. C. Suárez-Figueroa, E. Blomqvist, H. Lewen, M. Espinoza, A. Gómez-Pérez, R. Palma. *D5.6.2 Experimentation and Evaluation of the NeOn Methodology*. NeOn project. <http://www.neon-project.org>. (February 2009).
- [Easterbrook et al., 2007] S. Easterbrook, J. Singer, M. A. Storey, D. Damian. *Selecting Empirical Methods for Software Engineering Research*. Guide to Advanced Empirical Software Engineering, by F. Shull (Editor), J. Singer (Editor), and D.I.K. Sjøberg (Editor). Pages 285-311. ISBN: 978-1-84800-043-8. 2007. Springer London.
- [Engler et al., 2006] M. Engler, D. Vrandecic, 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.
- [Espinoza, progress] in M. Espinoza. PhD Thesis: *Ontology Localization*. Spain. Universidad Politécnica de Madrid. To be appeared.
- [Espinoza et al., 2009] M. Espinoza, E. Montiel-Ponsoda, A. Gómez-Pérez. *Ontology localization*. Proceedings of the Fifth International Conference on Knowledge Capture (KCAP 2009). Redondo Beach, California, USA. Pages 33-40. ISBN: 978-1-60558-658-8.
- [Farquhar et al., 1997] A. Farquhar, R. Fikes, J. Rice. *The Ontolingua Server: A Tool for Collaborative Ontology Construction*. International Journal of Human Computer Studies 46(6):707–727. 1997.
- [Fensel et al., 2007] 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.
- [Fernández et al., 2006] M. Fernández, I. Cantador, P. Castells. *CORE: A Tool for Collaborative Ontology Reuse and Evaluation*. Proceedings of the 4th Int. Workshop on Evaluation of Ontologies for the Web (EON06), at the 15th International World WideWeb Conference (WWW06). Edinburgh, UK, 2006.
- [Fernández-López and Gómez-Pérez, 2004] M. Fernández-López, A. Gómez-Pérez. *Searching for a Time Ontology for Semantic Web Applications*. In Vari A, Vieu L (eds.), 3th International Conference on Formal Ontology in Information Systems (FOIS). Turin, Italy. Pp 331-441. 2004.
- [Fernández-López and Gómez-Pérez, 2002] M. Fernández-López, A. Gómez-Pérez. *Overview and Analysis of Methodologies for Building Ontologies*. Knowledge Engineering Review (KER). Vol. 17, Nº 2, pp 129-156. 2002.

- [Fernández-López et al., 2000] 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.
- [Fernández-López et al., 1997] 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.
- [Gangemi, 2007] A. Gangemi. *Design Patterns for Legal Ontology Construction*. Trends in Legal Knowledge: The Semantic Web and the Regulation of Electronic Social Systems. Editors: P. Casanovas, P. Noriega, D. Bourcier, F. Galindo. European Press Academic Publishing. 2007.
- [Gangemi, 2005] A. Gangemi. *Ontology Design Patterns for Semantic Web Content*. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference, Galway, Ireland, 2005. Springer.
- [García-Silva et al., 2008] A. García-Silva, A. Gómez-Pérez, M. C. Suárez-Figueroa, B. Villazón-Terrazas. *A Pattern Based Approach for Re-engineering Non-Ontological Resources into Ontologies*. The Semantic Web. Poceedings of the 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. LNCS 5367. ISBN: 978-3-540-89703-3. Pp: 167-181.
- [García-Castro et al., 2007] R. García-Castro, M. C. Suárez-Figueroa, A. Gómez-Pérez, D. Maynard, S. Costache, R. Palma, J. Euzenat, F. Lécué, A. Léger, T. Vitvar, M. Zaremba, D. Zyskowski, M. Kaczmarek, M. Dzbor, J. Hartmann, S. Dasiopoulou. *Knowledge Web. Deliverable 1.2.4. Architecture of the Semantic Web Framework*. February 2007. <http://knowledgeweb.semanticweb.org/>.
- [Golbeck et al., 2003] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, B. Parsia, J. Oberthaler. *The national cancer institute's thesaurus and ontology*. Journal of Web Semantics, 1(1), 2003.
- [Genesereth and Fikes, 1992] 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.
- [German Ministry of Defense, 1992] German Ministry of Defense. *V-Model: Software lifecycle process model*. 1992. General Reprint Nº250. Bundesminister des Innern, Koordinierungs-und Beratungstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung.
- [Gómez et al., 1997] A. Gómez, N. Juristo, C. Montes, J. Pazos. *Ingeniería del Conocimiento*. Editorial Centro de Estudios Ramón Areces, S.A. 1997. ISBN: 84-8004-269-9.

- [Gómez-Pérez, 2004] A. Gómez-Pérez. *Ontology Evaluation*. Handbook on Ontologies. Pages: 251-274. 2004. ISBN: 3-540-40834-7.
- [Gómez-Pérez et al., 2003] 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.
- [Gómez-Pérez and Lozano-Tello, 2005] 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.
- [Gómez-Pérez and Rojas-Amaya, 1999] 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.
- [Gómez-Pérez et al., 2007] J. M. Gómez-Pérez, C. Buil-Aranda, T. Pariente Lobo, G. Herrero Cárcel, A. Baena. *NeOn Deliverable D8.3.1 Ontologies for the Pharmaceutical Case Studies*. 2007.
- [Gómez-Pérez et al., 2006] J. M. Gómez-Pérez, C. Daviaud, B. Morera, R. Benjamins, T. Pariente Lobo, G. Herrero Cárcel, G. Tort. *NeOn Deliverable D8.1.1 Analysis of the Pharma Domain and Requirements*. September 2006. <http://www.neon-project.org/>.
- [Greenwood, 1973] E. Greenwood. *Metodología de la investigación social*. Editorial Paidós, Buenos Aires, Argentina. 1973.
- [Grüninger and Fox, 1995] M. Grüninger, 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).
- [Grüninger and Fox, 1994] M. Grüninger, 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.
- [Haase et al., 2006] 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/>.
- [Hayes, 1995] P. J. Hayes. *A Catalog of Temporal Theories*. Technical Report UIUC-BI-AI-96-01 at the Beckman Institute and Departments of Philosophy and Computer Science University of Illinois, 1995. <http://www.coginst.uwf.edu/users/phayes/TimeCatalog1.ps> and <http://www.coginst.uwf.edu/users/phayes/TimeCatalog2.ps>
- [Herrera-Viedma et al., 2005] E. Herrera-Viedma, F. Mata, L. Martínez, L. G. Pérez. *An Adaptive Module for the Consensus Reaching Process in Group Decision Making Problems*. V. Torra et al. (Eds.): MDAI 2005, LNAI 3558, pp. 89–98, 2005. Springer-Verlag Berlin Heidelberg 2005.

- [Hodge, 2000] 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>.
- [Horrocks and van Harmelen, 2001] I. Horrocks, F. van Harmelen (editors). *Reference Description of the DAML+OIL (March 2001) Ontology Markup Language*. Technical report, 2001. <http://www.daml.org/2001/03/reference.html>
- [Hristozova and Sterling, 2003] 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.
- [IEEE, 2006] *IEEE Standard for Developing a Software Project Life Cycle Process*. IEEE Std. 1074-2006. (Revision of IEEE Std. 1074-1997).
- [IEEE, 1997] *IEEE Standard for Developing Software Life Cycle Processes*. IEEE Std. 1074-1997 (Revision of IEEE Std. 1074-1995; Replaces IEEE Std. 1074.1-1995).
- [IEEE, 1995] *IEEE Guide for Developing Software Life Cycle Processes*. IEEE Std. 1074.1-1995.
- [IEEE, 1993] *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std. 830-1993.
- [IEEE, 1990] *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std. 610.12-1990 (Revision and redesignation of IEEE Std. 792-1983).
- [IEEE, 1990b] *IEEE Standard Glossary of Data Management Terminology*. IEEE Std. 610.5-1990.
- [Iglesias et al., 2006] M. Iglesias, C. Caracciolo, Y. Jaques, M. Sini, F. Calderini, J. Keizer, F. Le Hunte Ward, M. Nissim, A. Gangemi. NeOn Deliverable D7.1.1 WP7 User Requirements. September 2006. <http://www.neon-project.org/>.
- [ISO, 1999] ISO 12200:1999. *Computer applications in terminology. Machine-readable terminology interchange format (MARTIf). Negotiated interchange*.
- [ISO, 2000] ISO 1087-1:2000. *Terminology work. Vocabulary. Part 1: Theory and application*.
- [ISO, 2006] ISO 12207. *Systems and Software Engineering - Software Life Cycle Processes*. 2006.
- [Jiménez-Ruiz et al., 2008] E. Jiménez-Ruiz, B. Cuenca-Grau, U. Sattler, T. Schneider, R. Berlanga. *Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support*. The Semantic Web: Research and Applications. Proceedings of the 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008. Pp: 185-199. ISBN: 978-3-540-68233-2.

- [Jiménez-Ruiz et al., 2008b] E. Jiménez-Ruiz, B. Cuenca-Grau, U. Sattler, T. Schneider, R. Berlanga. *ProSe: A Protégé plugin for Reusing Ontologies, Safe and Economic*. Demo Paper. 13th edition of the Spanish Conference on Software Engineering and Databases (JISBD 2008), Gijón, Spain, October 7-10, 2008.
- [Jimeno-Yepes et al., 2009] A. Jimeno-Yepes, E. Jimenez-Ruiz, R. Berlanga-Llavori, D. Rebholz-Schuhmann. *Reuse of terminological resources for efficient ontological engineering in Life Sciences*. BMC Bioinformatics. Volume 10. ISSN: 1471-2105. 2009.
- [Jones, 1984] T. C. Jones. *Reusability in programming: A survey of the state of the art*. IEEE Transaction on Software Engineering. Vol. SE-IO, pp. 488-494, September 1984.
- [Juristo and Pazos, 1993] N. Juristo, J. Pazos. *Towards a joint life cycle for software and knowledge engineering*. 1993. Knowledge Oriented Software Design (A-27). Ed. J. Cuena. Pages: 119-138.
- [KACTUS Consortium, 1996] KACTUS Consortium. *The KACTUS Booklet version 1.0*. Esprit Project 8145 KACTUS. (1996).
<http://www.swi.psy.uva.nl/projects/NewKACTUS/Reports.html>
- [Kalfoglou and Schorlemmer, 2003] Y. Kalfoglou, M. Schorlemmer. *Ontology mapping: the state of the art*. The Knowledge Engineering Review. Volume 18, Issue 1 (January 2003). Pages: 1-31. 2003. ISSN: 0269-8889.
- [Kifer and Lausen, 1989] M. Kifer, G. Lausen. *F-Logic: a higher-order language for reasoning about objects, inheritance, and schema*. ACM Sigmod Record. Pages: 134-146. 1989.
- [Kim et al., 2009] S. Kim, M. Iglesias Sucasas, C. Caracciolo, V. Viollier, J. Keizer. *Integrating country-based heterogeneous data at the United Nations: FAO's geopolitical ontology and services*. Semantic Technology Conference 2009.
- [Kingston, 1994] J. Kingston. *Linking Knowledge Acquisition with CommonKADS Knowledge Representation*. BCS SGES Expert Systems 1994 conference, St John's College, Cambridge, 12-14 December 1994.
- [Kotis and Vouros, 2004] K. Kotis, G. Vouros. *The HCONE Approach to Ontology Merging*. In: The Semantic Web: Research and Applications. First European Semantic Web Symposium (ESWS 2004). Pp. 137-151.
- [Kruchten, 2000] P. Kruchten. *The Rational Unified Process: An Introduction, Second Edition*. ISBN: 0201707101. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 2000.
- [Landau, 1984] S. L. Landau. *Dictionaries. The art and craft of lexicography*, Cambridge: Cambridge University Press. 1984.

- [Larman, 2002] C. Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Second edition. Pearson Education, Inc. Publishing as Prentice Hall PTR. 2002. ISBN: 0-13-092569-1.
- [Lenat and Guha, 1990] D. B. Lenat, R.V. Guha. *Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Boston, Massachusetts. 1990.
- [Leuf and Cunningham, 2001] B. Leuf, W. Cunningham. *The Wiki Way*. Addison-Wesley, 2001.
- [Lozano-Tello, 2002] A. Lozano-Tello. PhD Thesis: *Métrica de idoneidad de ontologías*. Spain. Universidad de Extremadura, 2002. ISBN: 84-7723-537-6.
- [MacGregor, 1991] R. MacGregor. *Inside the LOOM classifier*. SIGART bulletin 2(3):70–76. 1991.
- [Masolo et al., 2003] 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.
- [Mizoguchi et al., 1995] 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.
- [Motta, 1999] E. Motta. *Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design*. IOS Press, Amsterdam, The Netherlands. 1999.
- [Newell, 1982] A. Newell. *The Knowledge Level*. Artificial Intelligence 18(1):87–127. 1982.
- [NeOn Consortium, 2006] NeOn Consortium. NeOn: Lifecycle Support for Networked Ontologies. *NeOn Technical Annex*. 2006.
- [Noy and McGuinness, 2001] 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.
- [Noy et al., 2000] N. F. Noy, R. W. Fergerson, M. A. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins. 2000.
- [Palma, 2009] R. Palma. PhD Thesis: *Ontology Metadata Management in Distributed Environments*. Spain. Universidad Politécnica de Madrid, 2009.

- [Pan et al., 2007] J. Z. Pan, L. Lancieri, D. Maynard, F. Gandon, R. Cuel, A. Leger. *Knowledge Web Deliverable D1.4.2.v2. Success Stories and Best Practices*. January 2007.
<http://www.cs.abdn.ac.uk/~jpan/pub/TR/D142v2-final.pdf>.
- [Paradela, 2001] L. Paradela. PhD Thesis: *Una Metodología para la Gestión del Conocimiento*. Spain. Universidad Politécnica de Madrid, 2001.
- [Paslaru and Mochol, 2005] E. Paslaru, M. Mochol. *Towards a reuse-oriented methodology for ontology engineering*. In Proceedings of 7th International Conference on Terminology and Knowledge Engineering (TKE 2005), pages 175-188, 2005.
- [Patel-Schneider et al., 2004] 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.
- [Pease et al., 2002] R. A. Pease, I. Niles, J. Li. *The suggested upper merged ontology: a large ontology for the semantic web and its applications*. Workshop on Ontologies and the Semantic Web at the AAAI 2002. 2002.
- [Pease and Niles, 2002] R. A. Pease, I. Niles. *IEEE Standard Upper Ontology: A Progress Report*. The Knowledge Engineering Review 17(1):65–70. 2002.
- [Peroni et al., 2008] S. Peroni, E. Motta, M. d'Aquin. Identifying Key Concepts in an Ontology, through the Integration of Cognitive Principles with Statistical and Topological Measures. The Semantic Web, 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Pages: 242-256. ISBN: 978-3-540-89703-3. 2008
- [Pfleeger, 2001] S. Pfleeger. *Software Engineering: Theory and Practice*. 2nd edition. Prentice Hall 2001. ISBN: 0-13-029049-1.
- [Pinto et al., 2004] H. S. Pinto, C. Tempich, S. Staab. *DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG 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.
- [Pinto and Martins, 2001] H. S. Pinto, J. P. Martins. *A methodology for ontology integration*. Proceedings of the 1st International Conference On Knowledge Capture. Victoria, British Columbia, Canada. Pages: 131-138. 2001. ISBN: 1-58113-380-4.
- [Poveda et al., 2009] M. Poveda, M.C. Suárez-Figueroa, A. Gómez-Pérez. *Common Pitfalls in Ontology Development*. To appear in “Current Topics in Artificial Intelligence, CAEPIA 2009 Selected Papers”. Editors: P. Meseguer, L. Mandow, R. M. Gasca. Sevilla, Spain, 9-13 November, 2009.

- [Pressman, 2001] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. New York, NY: McGraw-Hill, 2001.
- [Presutti et al., 2008] 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.
- [Presutti and Gangemi, 2008] V. Presutti, A. Gangemi. Content Ontology Design Patterns as practical building blocks for web ontologies. Proceedings of the 27th International Conference on Conceptual Modeling (ER2008). Barcelona, Spain. 2008.
- [Royce, 1970] W. W. Royce. *Managing the development of large software systems: concepts and techniques*. In Proceedings Western Electronic Show and Convention (WESCON), Los Angeles. August 25th-28th 1970.
- [Saint and Lawson, 1994] S. Saint, J. R. Lawson. *Rules for Reaching Consensus. A Modern Approach to Decision Making*. Jossey-Bass, San Francisco (1994).
- [Schneider, 2004] 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.
- [Schreiber et al., 1999] 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.
- [Schreiber et al., 1994] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, W. van de Velde. *CommonKADS: A comprehensive methodology for KBS development*. IEEE Expert: Intelligent Systems and Their Applications. Volume 9, Issue 6 (December 1994). Pages: 28-37. ISSN: 0885-9000.
- [Schreiber et al., 1994b] 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.
- [SEEMP Consortium, 2006] SEEMP Consortium. *SEEMP D31a. Supporting the State of the Art*. July 2006. <http://www.seemp.org/>.
- [SEEMP Consortium, 2006b] SEEMP Consortium. *SEEMP D11. User Requirement Definition*. May 2006. <http://www.seemp.org/>.
- [Simons, 1987] P. Simons. *Parts: A Study in Ontology*. Clarendon Press, Oxford, United Kingdom. 1987.

- [Simperl et al., 2009] E. Simperl, I. Popov, T. Bürger. *ONTOCOM Revisited: Towards Accurate Cost Predictions for Ontology Development Projects*. In: Proceedings of the European Semantic Web Conference 2009 (ESWC '09), Heraklion, Greece, May 20 - Jun, 04, 2009.
- [Simperl, 2009] E. Simperl. *Reusing ontologies on the Semantic Web: A feasibility study*. Data Knowledge Engineering. Volume 68. Number 10. Pages: 905-925. 2009.
- [Smith, 1997] 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.
- [Smith, 1996] B. Smith. *Mereotopology: A Theory of Parts and Boundaries*. Data and Knowledge Engineering, 20, 287-303. 1996.
- [Smith and Varzi, 2000] B. Smith, A. Varzi. *Fiat and Bona Fide Boundaries*. Philosophy and Phenomenological Research 60: 401–420. 2000.
- [Sommerville, 2007] I. Sommerville. *Software Engineering*. Eighth Edition 2007. Addison-Wesley. ISBN 0-321-31379-8.
- [Staab et al., 2001] S. Staab, H.P. Schnurr, R. Studer, Y. Sure. *Knowledge Processes and Ontologies*. IEEE Intelligent Systems 16(1):26–34. (2001).
- [Stellman and Greene, 2005] A. Stellman, J. Greene. *Applied Software Project Management*. ISBN: 0-596-00948-8. O'Reilly. (2005).
- [Stojanovic, 2004] L. Stojanovic. PhD Thesis: *Methods and Tools for Ontology Evolution*. University of Karlsruhe, Germany. 2004.
- [Stojanovic et al., 2002] L. Stojanovic, N. Stojanovic, S. Handschuh. *Evolution of the Metadata in the Ontology-based Knowledge Management Systems*. 1st German Workshop on Experience Management: Sharing Experiences about the Sharing of Experience. Berlin. March 7-8, 2002.
- [Studer et al., 1998] R. Studer, V. R. Benjamins, D. Fensel. *Knowledge Engineering: Principles and Methods*. Data & Knowledge Engineering 25 (1998). Pages: 161-197.
- [Suárez-Figueroa and Gómez-Pérez, 2008] M. C. Suárez-Figueroa, A. Gómez-Pérez. *Building Ontology Networks: How to Obtain a Particular Ontology Network Life Cycle?*. In Proceedings of the International Conference on Semantic Systems (ISEMANTICS 2008), pages 142 – 149, 2008.
- [Suárez-Figueroa et al., 2010] M. C. Suárez-Figueroa, A. Gómez-Pérez, M. Poveda, J. A. Ramos, J. Euzenat, C. Le Duc. *NeOn Deliverable D5.4.3. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. NeOn Project. <http://www.neon-project.org>. January 2010.

- [Suárez-Figueroa et al., 2009] M. C. Suárez-Figueroa, E. Blomqvist, M. D'Aquin, M. Espinoza, A. Gómez-Pérez, H. Lewen, I. Mozetic, R. Palma, M. Poveda, M. Sini, B. Villazon-Terrazas, F. Zablith, M. Dzbor. (2009). *NeOn Deliverable D5.4.2. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. NeOn Project. <http://www.neon-project.org>. February 2009.
- [Suárez-Figueroa et al., 2008] M. C. Suárez-Figueroa, G. Aguado de Cea, C. Buil, K. Dellschaft, M. Fernández-López, A. García, A. Gómez-Pérez, G. Herrero, E. Montiel-Ponsoda, M. Sabou, B. Villazon-Terrazas, Z. Yufei. *NeOn Deliverable D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks*. NeOn Project. <http://www.neon-project.org>. February 2008.
- [Suárez-Figueroa et al., 2007] 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.
- [Suárez-Figueroa et al., 2007b] 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.
- [Sure et al., 2002] 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.
- [Taylor, 2008] J. C. Taylor. *Project Scheduling and Cost Control: Planning, Monitoring and Controlling the Baseline*. J. Ross Publishing, 2008, ISBN: 9781932159110.
- [Uschold, 1996] 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>.
- [Uschold et al., 1998] M. Uschold, M. Healy, K. Williamson, P. Clark, S. Woods. *Ontology Reuse and Application*. In: Proc. International Conference on Formal Ontology and Information Systems - FOIS'98 (1998) <http://www.cs.utexas.edu/users/pclark/papers>.
- [US Department of Transportation, 2007] US Department of Transportation. *Systems Engineering for Intelligent Transportation Systems*. 2007. Available at: <http://www.ops.fhwa.dot.gov/publications/seitsguide/seguide.pdf>.
- [van Heijst et al., 1997] 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.

- [Varzi, 2007] 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.
- [Varzi, 2003] A. Varzi. *Mereology*. In Zalta (ed) Stanford Encyclopedia of Philosophy, Stanford: CSLI (on line publication at "<http://plato.stanford.edu/entries/mereology/>"). Last access: 6th February 2008). 2003.
- [Viégas et al., 2004] F. B. Viégas, M. Wattenberg, K. Dave. *Studying cooperation and conflict between authors with history flow visualizations*. Conference on Human Factors in Computing Systems 2004. Vienna, Austria. Pages. 575-582. ISBN: 1-58113-702-8.
- [Villazón-Terrazas, in progress] B. Villazón-Terrazas. PhD Thesis: *A Method for Reuse and Re-engineering Non-Ontological Resources into Ontologies*. Spain. Universidad Politécnica de Madrid. To be appeared.
- [Wiegers, 2003] E. Wiegers. *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle*. 2nd ed., Redmond: Microsoft Press. ISBN 0-7356-1879-8. 2003.
- [Wieringa, 1996] R. Wieringa. *Requirements Engineering: Frameworks for Understanding*. New York: John Wiley & Sons, 1996.
- [Zhou and Fikes, 2000] Q. Zhou, R. Fikes. A Reusable Time Ontology. Technical Report KSL-00-01. Knowledge Systems Laboratory, Stanford, California. 2000.

ANNEX I. Ontology Requirements Specification: Examples

In this annex we include three examples on how to use the guidelines for the ontology requirements specification activity proposed in Chapter 8.

1. SEEMP Reference Ontology Requirements Specification

The main objective of the SEEMP project is to develop an interoperable architecture for public e-Employment services (PES). The resultant architecture will consist of (a) 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; (b) a set of Local Ontologies, each PES uses its own Local Ontology, which describes the employment market in its own terms; (c) a set of mappings between each Local Ontology and the Reference Ontology; and (d) a set of mappings between the PES schema sources and the Local Ontologies. The SEEMP project relies on WSMO [Fensel et al., 2007], which permits to semantically describe Web Services, ontologies and mediators. WSML [de Bruijn et al., 2006] is the concrete language used in SEEMP for encoding those descriptions.

Here we present the requirements specification of the SEEMP Reference Ontology following the proposed guidelines of the NeOn Methodology. This requirements specification is not intended to be exhaustive, but it just describes the most important points. A detailed and complete requirements specification is described in [SEEMP Consortium, 2006]. Next we described the tasks performed.

Task 1. Identifying purpose, scope and implementation language.

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 [SEEMP Consortium, 2006b]. 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 PES, more specifically within the ICT (Information and Communication Technology) domain. Since SEEMP project relies on WSMO¹⁰⁸, the implementation language of the resultant ontology will be WSML.

Task 2. Identifying the intended end-users.

Since 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 [SEEMP Consortium, 2006b] allowed us to identify the following intended end-users of the ontology:

- User 1. Candidate who is unemployed and searches for a job or searches for 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 (curriculum vitae) or job postings and to prepare some data and statistics.
- User 4. National and Local Governments which want to analyse the situation on the employment market in their countries and prepare documents on employment, social and educational policy.

¹⁰⁸ <http://www.wsmo.org/index.html>

User 5. European Commission and the governments of EU countries which want to analyse the statistics and prepare international agreements and documents on the employment, social and educational policy.

Task 3. Identifying the intended uses.

The analysis of the motivating scenarios described in [SEEMP Consortium, 2006b] allowed us to identify the following main intended uses of the ontology:

- Use 1. To publish CVs. A job seeker places his/her CV on the PES Portal.
- Use 2. To publish Job Offers. An Employer places a Job Offer on the PES Portal.
- Use 3. To search for Job Offers. The Employer looks for candidates for the Job Offer through PES Portal.
- Use 4. To search for Employment information. Job Seeker looks for general information about employment in a given location at the PES Portal.
- Use 5. To provide Job Statistics. The PES Portal provides employment statistics to the Job Seeker and Employer.

Task 4. Identifying requirements.

The *non-functional ontology requirements* identified were:

- NFR1. The ontology must support a multilingual scenario in the following languages: English, Spanish, Italian, and French.
- NFR2. The ontology must be based on the international, European or de-facto standards in existence or under development.

For specifying the *functional ontology requirements* we used the competency question technique. We followed the *bottom up* approach for identifying them, because it was the more direct way to work with the domain experts. Competency questions were stored in an Excel file and then rewritten in a mind map tool as appears in Figure 63 and Figure 64, respectively.

In total we identified sixty competency questions, which are described in detail in [SEEMP Consortium, 2006]. 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?

N	Competency Questions	Answers
CQ1	What is the Job Seeker Name?	Lewis Hamilton
CQ2	What is the Job Seeker nationality?	British; Spanish; Italian; French; German
CQ3	When is the Job Seeker birthdate?	13/09/1984; 30/03/1970; 15/04/1978
CQ4	What is the Job Seeker contact information?	Telf: 34600654231. Email: jarg@fi.upm.es
CQ5	What is the Job Seeker current job?	Programmer, Computer Engineer, Computer Assistant
CQ6	What is the Job Seeker desired job?	Radio engineer, Hardware designer, Software Engineer
CQ7	What are the Job Seeker desired working conditions?	Autonomous; Seasonal Job, Traineeship, Consultant
CQ8	What kind of contract does the Job Seeker want?	Full time; Partial time; Autonomous; Seasonal Job;
CQ9	How much salary does the Job Seeker want to earn?	3000 euros per month, 40000 euros per year
CQ10	What is the Job Seeker education level?	Basic education; Higher education/University
CQ11	What is the Job Seeker work experience?	3 months, 6 months, 1 year, 2 years, 3 years
CQ12	What is the Job Seeker knowledge?	Java Programming, C Programming, Database administration
CQ13	What is the Job Seeker expertise?	Software Engineering
CQ14	What are the Job Seeker skills?	SQL programming, network administration
CQ15	What publications does the Job Seeker have?	International Congress publication
CQ16	What hobbies does the Job Seeker have?	Reading, playing cards, listening music
CQ17	What is the employer information?	CEFRIEL Research Company, Milano, Italy
CQ18	What kind of job does the employer offer?	Java Programmer, C Programmer, Database administration
CQ19	What kind of contract does the employer offer?	Seasonal Job; Autonomous
CQ20	How much salary does the employer offer?	3500 euros, 3000 USD, 2000 euros

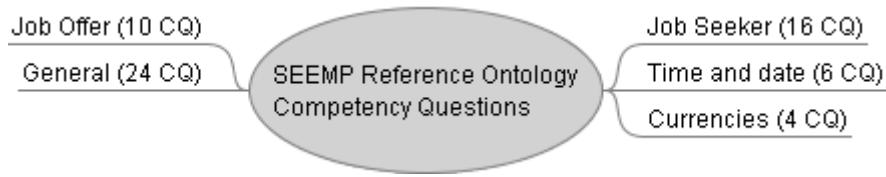
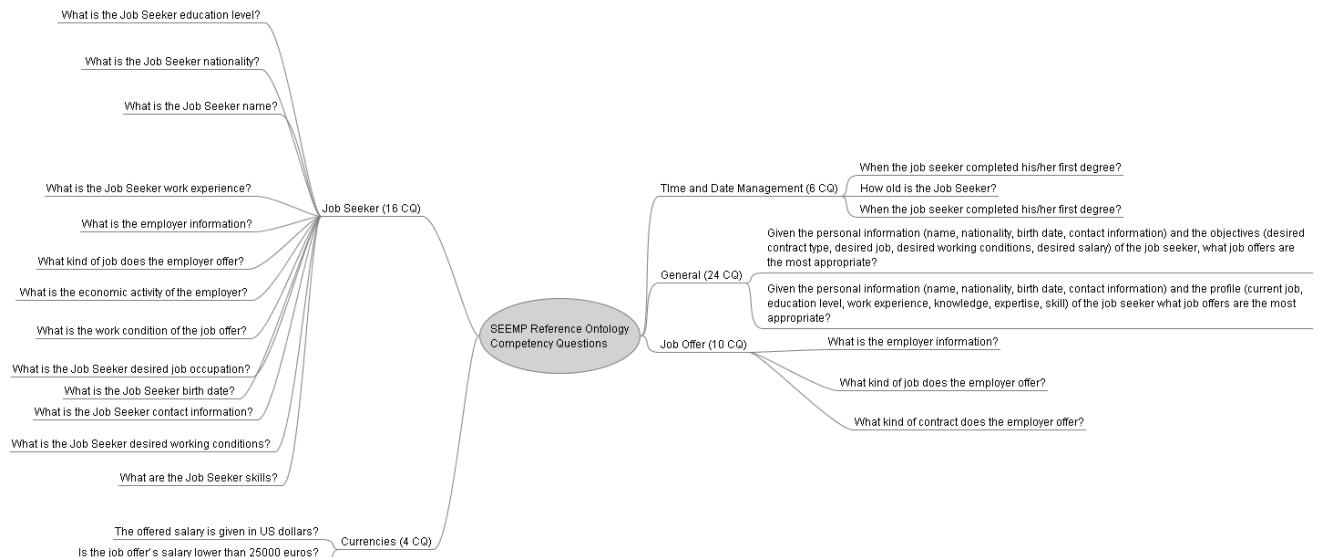
Figure 63. Excerpt of the Competency Questions and answers in an Excel file



Figure 64. Excerpt of the Competency Questions in a Mind Map tool

Task 5. Grouping functional requirements.

The sixty competency questions, described in [SEEMP Consortium, 2006], were manually grouped into five groups with the domain experts' help. Figure 65 shows the final 5 groups: Job Offer, Job Seeker, Currencies, Time and Date, and General. General competency questions are the result of the composition of simple queries into complex ones. The criteria for grouping the competency questions were based on the identified uses, the identified users and the domain expert suggestions. Figure 66 shows the 5 groups with some competency questions.

**Figure 65. Competency Questions groups****Figure 66. Competency Questions groups in detail**

Task 6. Validating 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 set of requirements (both non-funcional and functional requirements):

- Correctness. Domain experts checked the correctness of each non-functional requirement and of each competency question, verifying that its formulation and answers were correct.
- Consistent. Domain experts also verified that the non-functional requirements and the competency questions did not have any possible inconsistency.

Task 7. Prioritizing requirements.

Within the SEEMP Reference Ontology requirements 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 and be able to cover all the non-functional requirements.

Task 8. Extracting 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 identified are shown in Table 34.

Terms from Competency Questions	
Job Seeker CV Personal Information Name Gender Birth date	Address Nationality Contact (phone, fax, mail) Objective Job Category
Terms from Answers	
SW engineer, programmer British, Spanish, Italian, French Autonomous, Seasonal Job Basic education, Higher education	Research, Financial, Education 1 year, 2 years, 3 years 3000 Euros per month CEFRIEL Research Company
Objects	
Andorra, Angola, Argentina, Australia, Bolivia, France, Italy, Malta, Spain, etc. Euro, Zloty, Great British Pound, US Dollar, Peso, etc. CEFRIEL, ATOS, etc.	

Table 34. Examples of terminology

After following these tasks, the output of the ontology requirements specification activity is the Ontology Requirements Specification Document. An excerpt of this document, which has been written for this thesis, is shown in Table 35.

	SEEMP Reference Ontology Requirements Specification Document
1 Purpose	The purpose of building the Reference Ontology is to provide a consensual knowledge model of the employment domain that can be used by public e-Employment services.
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 Implementation Language	The ontology has to be implemented in WSML language.
4 Intended End-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. User 4. National and Local Governments which want to analyse 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 analyse the statistics and prepare international agreements and documents on the employment, social and educational policy.
5 Intended Uses	Use 1. To publish CVs. A job seeker places his/her CV on the PES Portal. Use 2. To publish Job Offers. An Employer places a Job Offer on the PES Portal. Use 3. To search for Job Offers. The Employer looks for candidates for the Job Offer through

SEEMP Reference Ontology Requirements Specification Document	
	<p>PES Portal.</p> <p>Use 4. To 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. To provide Job Statistics. The PES Portal provides employment statistics to the Job Seeker and Employer.</p>
6 Ontology Requirements	
	<p>a. Non-Functional Requirements</p> <p>NFR1. The ontology must support a multilingual scenario in the following languages: English, Spanish, Italian, and French.</p> <p>NFR2. The ontology must be based on the international, European or de-facto standards in existence or under development.</p>
	<p>b. Functional Requirements: Groups of Competency Questions</p> <p>CQG1. Job Seeker (14 CQ)</p> <p>CQ1. What is the Job Seeker's name? Lewis Hamilton</p> <p>CQ2. What is the Job Seeker's nationality? British, Spanish, Italian, French, etc.</p> <p>CQ3. What is the Job Seeker's birth date? '13/09/1984; 30/03/1970; 15/04/1978</p> <p>CQ4. What is the Job Seeker's contact information? Tel: 34600654231. Email: jsanz@fi2.upm2.es</p> <p>CQ5. What is the Job Seeker's current job? Programmer, Computer Engineer, etc.</p> <p>CQ6. What is the Job Seeker's desired job? Hardware designer, Software Engineer, etc.</p> <p>CQ7. What are the Job Seeker's desired working conditions? Autonomous, Seasonal Job, Traineeship, Consultant, etc.</p> <p>CQ8. What kind of contract does the Job Seeker want? Full time, Partial time, Autonomous, Seasonal Job, etc.</p> <p>CQ9. How much salary does the Job Seeker want to earn? 3000 Euros per month, 40000 Euros per year, etc.</p> <p>CQ10. What is the Job Seeker's education level? Basic education, Higher education, etc.</p> <p>CQ11. What is the Job Seeker's work experience? 6 months, 1 year, 2 years, etc.</p> <p>CQ12. What is the Job Seeker's knowledge? Java Programming, Database Administration, etc.</p> <p>CQ13. What is the Job Seeker's expertise? Software Engineering, etc.</p> <p>CQ14. What are the Job Seeker' skills? SQL programming, network administration, etc.</p> <p>CQG2. Job Offer (11 CQ)</p> <p>CQ15. What is the employer's information? CEFRIEL Research Company, Milano, Italy; ATOS, Madrid, Spain; etc.</p> <p>CQ16. What kind of job does the employer's offer? Java Programmer, C Programmer, Database administration, etc.</p> <p>CQ17. What kind of contract does the employer's offer? Seasonal Job, Autonomous, etc.</p> <p>CQ18. How much salary does the employer's offer? 3500 Euros, 3000 USD, etc.</p> <p>CQ19. What is the economic activity of the employer? Research, Education, Industrial, etc.</p> <p>CQ20. What is the description of the job offer? Sun Certified Java Programmer, etc.</p> <p>CQ21. What are the working conditions of the job offer? Full time, Partial time, Autonomous, Seasonal Job, etc.</p> <p>CQ22. What is the required education level for the job offer? Basic education, Higher education, etc.</p> <p>CQ23. What is the required work experience for the job offer? 1 year, 2 years, 3 years, 4 years, 5 or more years</p>

SEEMP Reference Ontology Requirements Specification Document			
	CQ24. What is the required knowledge for the job offer? Java, Haskell, Windows, etc.		
	CQ25. What are the required skills for the job offer? ASP Programmer, Data warehouse, Hardware programming, etc.		
7 Pre-Glossary of Terms			
a. Terms from Competency Questions			
Job Seeker	27	Address	1
CV	2	Nationality	1
Personal Information	3	Contact (phone, fax, mail)	3
Name	4	Objective	3
Gender	1	Job Category	3
Birth date	1		
b. Terms from Answers			
SW engineer, programmer	5	Research, Financial, Education	4
British, Spanish, Italian, French	1	1 year, 2 years, 3 years	1
Autonomous, Seasonal Job	2	3000 Euros per month	1
Basic education, Higher education	1	CEFRIEL Research Company	1
c. Objects			
Andorra, Angola, Argentina, Australia, Bolivia, France, Italy, Malta, Spain, etc.			
Euro, Zloty, Great British Pound, US Dollar, Peso, etc.			
CEFRIEL, ATOS, etc.			

Table 35. Excerpt of SEEMP Reference Ontology Requirement Specification Document

2. Invoice Reference Ontology Requirements Specification

The invoice reference ontology designed in the scope of the Pharmaceutical case study within the NeOn project aims to solve the lack of interoperability between invoice emitters and receivers.

Next we describe how we applied the NeOn methodological guidelines for the ontology requirements specification activity to this use case.

Task 1. Identifying purpose, scope and implementation language.

The scope of the invoice reference ontology is to cover the most important standards of the electronic invoicing domain. Electronic invoicing standards such as EDIFACT¹⁰⁹ (Electronic Data Interchange for Administration, Commerce and Transport) or UBL¹¹⁰ (Universal Business Language) should be present along with other proprietary solutions and all the concepts needed for representing any invoice that can be emitted by a company, as well as concepts to represent the invoicing workflow. The implementation language selected is OWL.

Task 2. Identifying the intended end-users.

The intended end-users of the invoice reference ontology are:

- User 1. User of the invoicing application who is going to model a new invoice.
- User 2. User who emits invoices.
- User 3. User who receives invoices.
- User 4. User who administrates the invoicing system.

Task 3. Identifying the intended uses.

The intended uses of the invoice reference ontology are:

- Use 1. To create new types of invoices.
- Use 2. To create and send particular invoices.
- Use 3. To obtain data for an invoice.
- Use 4. To manage data from different types of invoices.

Task 4. Identifying requirements.

The ***non-functional ontology requirements*** identified were:

- NFR1. The ontology must support a multilingual scenario in the following languages: Spanish, Catalan, Basque, and Galician.
- NFR2. The ontology must be based on terminology from the standards UBL and EDIFACT.

For specifying the ***functional ontology requirements*** we wrote a list of competency questions following the bottom-up strategy. We created first simple questions and by composition we derived other complex questions. The tool for gathering the competency questions was *MS Word*. Examples of the competency questions are:

1. What is the activity of the invoice emitter?
2. How many different concepts have the different invoices emitted by e.g., wholesaler and a laboratory?

¹⁰⁹ <http://www.unece.org/trade/untdid/welcome.htm>

¹¹⁰ <http://www.oasis-open.org/committees/ubl>

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. Is possible to apply any special price to this invoice?
6. What product have we received?
7. What invoicing technologies are using the emitters of this invoice?

Task 5. Grouping functional requirements.

The functional requirements in the form of competency questions have been grouped into seven groups.

The result of grouping CQs is:

- CQG1. Competency questions regarding the invoicing workflow (11 CQs)
- CQG2. Competency questions regarding inference rules (4 CQs)
- CQG3. Competency questions related to the receiver of invoices (16 CQs)
- CQG4. Competency questions regarding the technology used by the emitters (5 CQs)
- CQG5. Competency questions related to the emitter of invoices (16 CQs)
- CQG6. Competency questions related to time and date management (15 CQs)
- CQG7. Competency questions related to currencies (8 CQs)

Task 6. Validating the set of requirements.

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

Task 7. Prioritizing requirements.

The ontology requirements were not prioritized. This means the first version of the ontology must be able to represent the knowledge contained in all the competency questions and be able to cover all the non-functional requirements.

Task 8. Extracting terminology and its frequency.

This task was not considered.

After following these tasks, the output of the ontology requirements specification activity is the Ontology Requirement Specification Document. An excerpt of this document, which has been written for this thesis, is shown in Table 36.

Invoice Reference Ontology Requirements Specification Document	
1 Purpose	The purpose of building the Invoice Reference Ontology is to solve the lack of interoperability between invoice emitters and receivers.
2 Scope	The ontology has to focus the most important standards of the electronic invoicing domain as well as the invoicing workflow.
3 Implementation Language	The ontology has to be implemented in OWL.
4 Intended End-Users	User 1. User of the invoicing application who is going to model a new invoice. User 2. User who emits invoices. User 3. User who receives invoices. User 4. User who administrates the invoicing system.
5 Intended Uses	Use 1. To create new types of invoices. Use 2. To create and send particular invoices. Use 3. To obtain data for an invoice. Use 4. To manage data from different types of invoices.
6 Ontology Requirements	<p>a. Non-Functional Requirements</p> <p>NFR1. The ontology must support a multilingual scenario in the following languages: Spanish, Catalan, Basque, and Galician.</p> <p>NFR2. The ontology must be based on terminology from the standards UBL and EDIFACT.</p> <p>b. Functional Requirements: Groups of Competency Questions</p> <p>CQG1. Competency questions regarding the invoicing workflow (11 CQs) CQG2. Competency questions regarding inference rules (4 CQs) CQG3. Competency questions related to the receiver of invoices (16 CQs) CQG4. Competency questions regarding the technology used by the emitters (5 CQs) CQG5. Competency questions related to the emitter of invoices (16 CQs) CQG6. Competency questions related to time and date management (15 CQs) CQG7. Competency questions related to currencies (8 CQs)</p>

Table 36. Excerpt of Invoice Ontology Requirements Specification Document

3. Semantic Nomenclature Reference Ontology Requirements Specification

The main objectives of the Semantic Nomenclature case study in the NeOn project are (a) helping in the systematization of the creation, maintenance and keeping up-to-date drug-related information, and (b) 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 network based on the nomenclature of products in the pharmaceutical sector in Spain. This reference ontology model should be a compilation of the main terms and objects related to drugs, the general aspects of them and classify this pharmaceutical terms according to the Anatomical Therapeutic Chemical (ATC) classification. Additionally, to create the Nomenclature ontology network the reference ontology model should be 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 requirements specification in the Semantic Nomenclature case study based on the methodological guidelines proposed in this thesis.

Task 1. Identifying purpose, scope and implementation language.

The development of the Nomenclature ontology network 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 [Gómez-Pérez et al., 2006]. The ontology network should provide a consensual knowledge of the pharmaceutical domain, and solve the lack of communication between stakeholders in the pharmaceutical sector. The purpose of the Nomenclature ontology network 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 pharmaceutical sector. The implementation language selected is OWL.

Task 2. Identifying the intended end-users.

The different intended end-users of the Nomenclature ontology network are:

- User 1. Pharmacists. Pharmacists are the end-users of the ontology and navigate across the ontology searching for drug information.
- User 2. GSCoP (General Spanish Council of Pharmacists) technicians. GSCoP technicians navigate across the ontology network and search for 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 analysts. Spanish Government analysts study the situation of the pharmaceutical product information in the Spanish market or update the content.

Task 3. Identifying the intended uses.

The analysis of the motivating scenarios described in [Gómez-Pérez et al., 2006], allowed us to identify the following main intended uses of the ontology:

- Use 1. To search for updated information about the characteristics of pharmaceutical products
- Use 2. To connect heterogeneous pharmaceutical models
- Use 3. To update pharmaceutical product information databases

Task 4. Identifying requirements.

The ***non-functional ontology requirement*** identified was:

- NFR1. The ontology must support a multilingual scenario in the following languages: Spanish, Catalan, Basque, and Galician.

We adopted a bottom-up approach for a better understanding of the domain and for the identification of the ***functional requirements*** in the form of competency questions. Competency questions were described in a *Word file* as appears in the excerpt shown in Figure 67.

Specific competency questions related with the Pharmaceutical Product are:
CQ1. What is the drug commercial name?
CQ2. What is the drug main active ingredient (molecule)?
CQ3. What is its Spanish national code?
CQ4. What is the drug registration date?
CQ5. What is the drug withdrawal date?
CQ6. What is the drug reference price?
CQ7. How much euros costs it?
CQ8. Which is the drug laboratory manufacturer?
CQ9. Which one is the drug therapeutical WHO group?
CQ10. What is the drug commercial price?
CQ11. Which is the drug generic name?
CQ12. Which is the drug defined daily doses DDDs?
CQ13. Which is the drug composition?
CQ14. Is it a narcotic?
CQ15. Which are the drug contraindications?
CQ16. What is the drug dosage?
CQ17. Which method of administration has the drug?
CQ18. What is the drug pharmaceutical form?
CQ19. What indications does the drug have?
CQ20. Is the drug financed by Social Security (Spanish Health Care System)?

Figure 67. Excerpt of the Semantic Nomenclature Competency Questions in a Word file

We have identified sixty-one competency questions; they are described in detail in [Gómez-Pérez et al., 2007]. 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*

Task 5. Grouping functional requirements.

The sixty-one competency questions described in [Gómez-Pérez et al., 2007] were grouped into the following 3 groups:

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

The criteria for grouping competency questions were 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, obtaining a new group.

CQG4. Composite ones (16 competency questions)

Figure 68 shows examples of CQs in the Laboratory and Pharmaceutical Product groups.

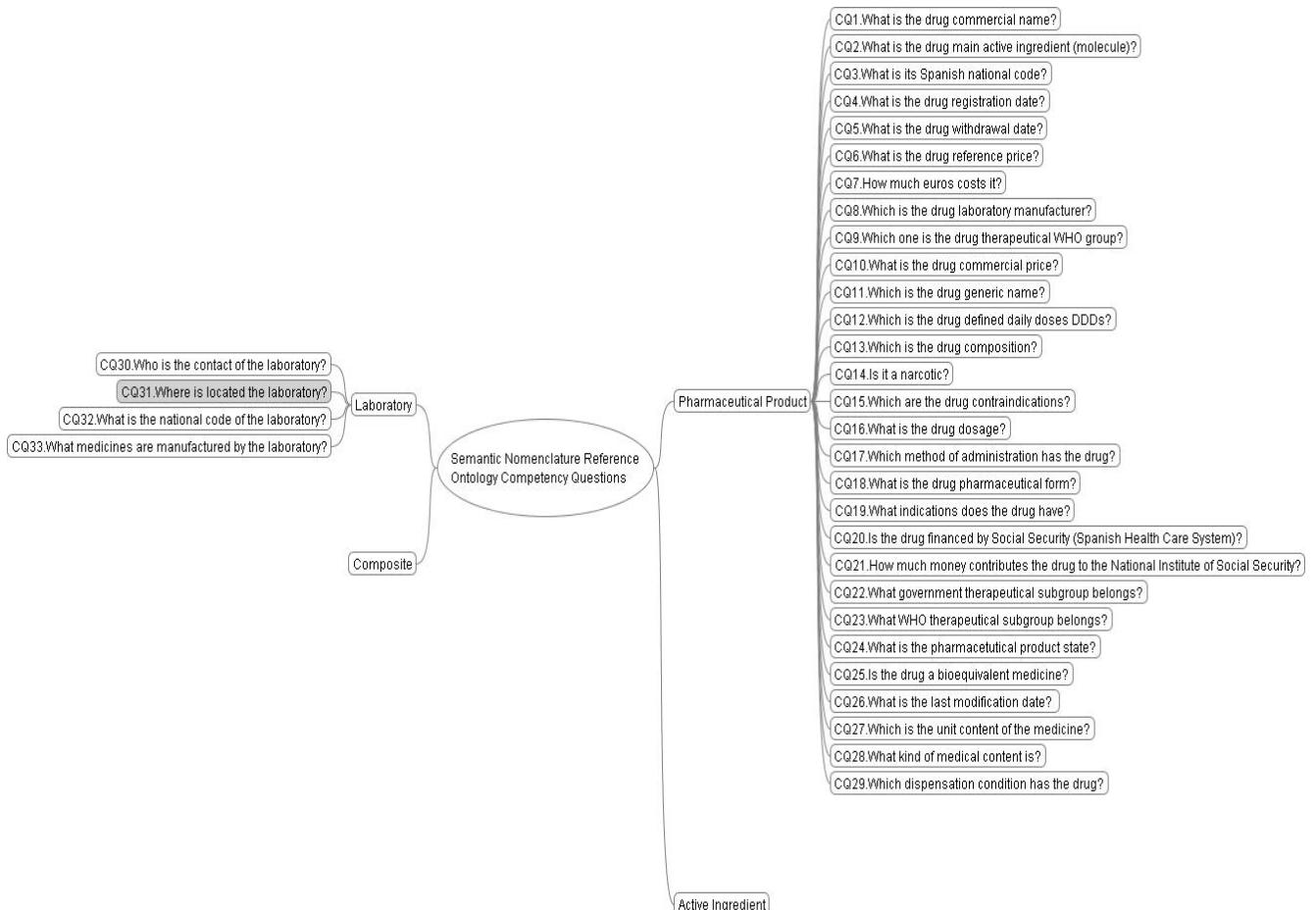


Figure 68. Examples of Competency Questions in groups

Task 6. Validating 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 mainly used the following criteria for validating the set of requirements (both non-functional and functional requirements):

- ❑ Correctness. Domain experts checked the correctness of each non-functional requirement and of each competency question, verifying that its formulation and answers were correct.
- ❑ Consistent. Domain experts also verified that the non-functional requirements and the competency questions did not have any possible inconsistency.

Task 7. Prioritizing requirements.

In the Semantic Nomenclature case study, 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 and be able to cover all the non-functional requirements.

Task 8. Extracting 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 37.

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

Table 37. 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 38.

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

Table 38. Examples of objects

The output of the ontology requirements specification activity is the Ontology Requirement Specification Document. An excerpt of this document is shown in Table 39.

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 Implementation Language	The ontology has to be implemented in OWL.												
4 Intended End-users	User 1. Pharmacist. Pharmacists are the end-users of the ontology and navigate across the ontology searching for drug information. User 2. GSCoP technicians. User 3. Spanish Government analysts.												
5 Intended Uses	Use 1. To search for updated information about the characteristics of pharmaceutical products. Use 2. To connect heterogeneous pharmaceutical models. Use 3. Use 3. To update pharmaceutical product information databases.												
6 Ontology Requirements	<p>a. Non-Functional Requirements</p> <p>NFR1. The ontology must support a multilingual scenario in the following languages: Spanish, Catalan, Basque, and Galician.</p> <p>b. Functional Requirements: Groups of Competency Questions</p> <p>CQG1. Pharmaceutical Product (29 competency questions) CQG2. Laboratory (4 competency questions) CQG3. Active Ingredient (12 competency questions). CQG4. Composed ones (16 competency questions).</p>												
7 Pre-Glossary of Terms	<p>a. Terms from Competency Questions</p> <table border="1"> <tr> <td>Drug</td><td>29</td><td>Active Ingredient</td><td>20</td></tr> <tr> <td>Medicine</td><td>15</td><td>Laboratory</td><td>14</td></tr> </table> <p>b. Terms from Answers</p> <table border="1"> <tr> <td>Aspirina C Ácido Acetilsalicílico</td><td>7127291 01/01/1976</td></tr> </table> <p>c. Objects</p> <table border="1"> <tr> <td>Ibuprofeno Butibufeno</td><td>Galamina Procaina</td></tr> </table>	Drug	29	Active Ingredient	20	Medicine	15	Laboratory	14	Aspirina C Ácido Acetilsalicílico	7127291 01/01/1976	Ibuprofeno Butibufeno	Galamina Procaina
Drug	29	Active Ingredient	20										
Medicine	15	Laboratory	14										
Aspirina C Ácido Acetilsalicílico	7127291 01/01/1976												
Ibuprofeno Butibufeno	Galamina Procaina												

Table 39. Excerpt of Semantic Nomenclature Reference Ontology Requirement Specification Document

ANNEX II. General Ontology Reuse: Example

In this annex we include an example of how to use the guidelines for the reuse of general ontologies presented in Chapter 1 (Section 10.3.3).

We present an example of reusing a general ontology in the Pharmaceutical Product ontology (PPO) that represents pharmaceutical products. This ontology will be used as a bridge between proprietary systems for managing financial and product knowledge interoperability in pharmaceutical laboratories, companies and distributors in Spain. The reuse of a general ontology in the PPO has been performed following the methodological guidelines described in this thesis. To perform such a reuse, we took into account the four¹¹¹ (out of 61) competency questions (CQs) [Gómez-Pérez et al., 2007] shown in Table 40.

Identifier	Competency Question	Answer
CQ1	<i>What is the composition of Frenadol ©?</i>	<i>Paracetamol, Dextrometorphan, Caffeine, and Citrate clofenaamine</i>
CQ2	<i>Is Caffeine a substance of Frenadol ©?</i>	<i>Yes</i>
CQ3	<i>Which is the main active ingredient of Frenadol ©?</i>	<i>Paracetamol</i>
CQ4	<i>Which is the main substance of Frenadol ©?</i>	<i>Paracetamol</i>

Table 40. Excerpt of the competency questions of PPO

Activity 1. Identifying the type of general ontology to be reused.

Let's focus on CQ1, in particular, in the relation *is composition of*. We can note, on the one hand, that a substance *x* can be a component of *y*, *y* can be a component of *z*, etc, in such a way that an order *x < y < z* is established. For example, oxygen is a part of every alcohol, there is an alcohol that is part of paracetamol, and a paracetamol is part of Frenadol ©. On the other hand, if *x ≠ y* and *x* is a component of *y*, then *y* necessarily has another component *z* such that *z* does not have components in common with *x*. For example, paracetamol is not Frenadol ©, but a part of it; consequently, Frenadol © necessarily has parts different to paracetamol. Therefore, *is composition of* is candidate to be modelled using mereology terms (the formalization of the *part of* relation). Consequently and taking into account the heuristics presented in Table 16 and the examples presented in Table 17, it seems reasonable to consider the reuse of a **mereology ontology** in the development of PPO.

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

As already mentioned, Section 10.3.2 provides definitions and axioms for the mereology theory that are useful in this example.

¹¹¹ For the sake of clarity in the example description, we have assigned new identifiers to the CQs involved in this example (the new identifiers are CQ1, CQ2, CQ3, and CQ4).

Activity 3. Searching for general ontologies supported by the theory.

We should search for ontologies that support partially or completely the mereology theory. Using a general purpose search engine (Google) and a specific one (Swoogle), we have found a series of mereology ontologies. The list of general ontologies that we have studied is the following one:

- *KACTUS* [KACTUS Consortium, 1996] ontology library, implemented in CML [Schreiber et al., 1994b], it is maintained by the University of Amsterdam. Such a library contains the Mereological Ontology (MO), which is an adapted version of Borst's proposals [Borst, 1997].
- *DOLCE* [Masolo et al., 2003] is one of the ontologies developed within the WonderWeb European project¹¹². It is available in KIF and OWL.
- The *Standard Upper Ontology*¹¹³ (SUO) [Pease and Niles, 2002] is the result of a joint effort to create a large, general-purpose, formal ontology. It is promoted by the IEEE Standard Upper Ontology working group, and its development began in May 2000. This ontology is implemented in KIF and Protégé; on the other hand, SUO formally describes mereology and topology terms. The general predicates in this section of the ontology are adapted from Barry Smith, Borgo et al.'s work, and from Casati and Varzi's mereologies.
- *The ontology based on Barry Smith and other authors' work* [Smith, 1997; Smith, 1996; Smith and Varzi, 2000; Casati et al., 1998] implemented in KIF, which is referred in the SUO web page. It represents various mereological definitions and axioms concerning boundaries and objects¹¹⁴.
- The mereology based on the work of *Borgo et al.*, which is another ontology referred in SUO web page (see footnote 113). These authors describe a set of definitions and axioms concerning mereology in [Borgo et al., 1997; Borgo et al., 1996]. Such an ontology is currently implemented in KIF. The ontology formalizes a CEM mereology (except the product principle).
- *Casati and Varzi's mereology* [Casati and Varzi, 1995] can be also found implemented in KIF and is also referred in the SUO web page.

Activity 4. Performing a comparative study.

As a result of this activity in the case of reusing a mereology ontology in the Pharmaceutical Product ontology (PPO), we obtained Table 41. It is worth mentioning that for the case of the ontology based on 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 ontology being developed, but not A.5, then A.4 should be completely implemented.

¹¹² <http://wonderweb.semanticweb.org/>

¹¹³ <http://suo.ieee.org/SUO/ontologies/Guarino.txt>

¹¹⁴ <http://suo.ieee.org/>

Theory	Principles and definitions	KACTUS MO	DOLCE OWL	SUO	Ontology based on Smith et al.	Ontology based on Borgo et al.	Ontology based on 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 + (A.4)	A.4) Weak supplementation	Yes	No	No	Yes	Inferred	No
EM = M + (A.5) (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	D.9) Atom	No	Yes	No	No	No	No
	A.9) Atomicity	No	No	No	No	No	No

Table 41. Features of ontologies that implement mereology theories

Activity 5. Selecting the general ontology to be reused.

Task 5.1. Reformulating the CQs.

We should reformulate the CQs included in the ORSD of the PPO using the vocabulary of the mereology theory. Concretely, we reformulated the CQs included in Table 40 in the way shown in Table 42, using Table 19 and Table 20 provided in the methodological guidelines for this task.

Original CQ		Transformed CQ	
Identifier	CQ	Identifier	CQ
CQ1	Which is the composition of Frenadol ©?	CQ1'	<i>Which are the proper parts of Frenadol ©?</i>
		CQ1''	<i>Which are the direct parts of Frenadol ©?</i>
CQ2	Is Caffeine a substance of Frenadol ©?	CQ2'	<i>Is Caffeine a part of Frenadol ©?</i>
CQ3	Which is the main active ingredient of Frenadol ©?	CQ3	This CQ does not require a reformulation. However, it is worth mentioning that "active ingredient" requires the definition of "part of"
CQ4	Which is the main substance of Frenadol ©?	CQ4	This CQ does not require a reformulation. However, it is worth mentioning that "main substance" requires the definition of "part of"

Table 42. Competency questions reformulation

Task 5.2. Identifying the features of the general ontology to be reused.

To identify what typical definitions, axioms and principles of the mereology theory are needed in the development of PPO we based on the heuristics provided in Table 23 and Table 24. We obtained the following conclusions:

- A.1) Reflexivity. In general, we are not interested in the substance S itself when we ask for its parts.
- A.2) Antisymmetry. It is ever reused.
- A.3) Transitivity. We are interested in all the levels of parts when we ask: *which are the parts of drug D?*
- D.1) Proper part. In general, we are not interested in the substance S itself when we ask for its parts.
- D.2) Direct part. We are interested in knowing the direct parts of a substance.
- D.3) Overlap. We are not interested in the common parts of different substances.
- D.4) Underlap. We are not interested in the common wholes of different substances.
- D.5) Disjoint. We are not interested in knowing if different substances are disjoint.
- A.4) Weak supplementation. It is ever reused.
- A.5) Strong supplementation. We are not interested in distinguishing between different substances that are isomers.
- We are not interested in the rest of axioms and definitions of the merology theory.

The aforementioned features are shaded in Table 43.

Theory	Principles and definitions	KACTUS MO	DOLCE OWL	SUO	Ontology based on Smith et al.	Ontology based on Borgo et al.	Ontology based on 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 + (P4)	A.4) Weak supplementation	Yes	No	No	Yes	Inferred	No
EM = M + (A5) (Let's note that (A5) implies (A4))	A.5) Strong supplementation	No	No	No	Yes	Yes	No
CM = M + (A6) + (A7)	A.6) Sum principle	No	No	No	Yes	Yes	No
	A.7) Product principle	No	No	No	Yes	No	No
CEM = CM + (A5)	D.6) Binary sum	No	Yes	Yes	Yes	Yes	Yes
	D.7) Binary product	No	No	Yes	Yes	Yes	Yes
GM = M + (A8)	A.8) Unrestricted fusion principle	No	No	No	Yes	No	No
GEM = GM + (A5)	D.8) General sum	No	Yes	No	Yes	No	No
AX = (A9) + a mereology X	D.9) Atom	No	Yes	No	No	No	No
	A.9) Atomicity	No	No	No	No	No	No

Table 43. Features required for the PPO¹¹⁵¹¹⁵ The characteristics that appear shaded are those required for the Pharmaceutical Product Ontology (PPO).

Task 5.3. Determining the most appropriate general ontology.

We analysed the candidate¹¹⁶ mereology ontologies with respect to the criteria identified in Section 10.3.3 (Task 5.3). Taking into account the ways to measure each criterion and the possible values that can be assigned, we obtained the results shown in Table 44.

To obtain the score for each candidate mereology ontology, we used the formula proposed in Section 10.3.3 (Task 5.3) obtaining the following scores for each candidate ontology.

- KACTUS MO → Score = 1,25
- DOLCE OWL → Score = 0,78
- SUO → Score = 0,75
- The ontology based on Barry Smith and other authors' work* → Score = 0,42
- The ontology based on the work of Borgo and other authors* → Score = 0,47
- The ontology based on Casati and Varzi* → Score = 0,33

Thus, we selected the candidate mereology ontology best scored, that is, the KACTUS MO ontology. 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. Customizing the selected general ontology

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

Activity 7. Integrating the general ontology in the ontology network being developed.

The product of the customization, that is the customized KACTUS MO, has been included in the PPO, carried out for this example.

¹¹⁶ We consider here all the general ontologies obtained in Activity 3, because all of them represent at least one of the features required by the PPO, as shown in Table 43.

Criteria	Weight	Values					
		KACTUS MO	DOLCE OWL	SUO	Ontology based on Smith et al.	Ontology based on Borgo et al.	Ontology based on Casati and Varzi
Reuse cost							
Reuse economic cost	(-)	10	High	High	High	High	High
Reuse time required	(-)	7	High	High	High	High	High
Understandability effort							
Quality of the documentation	(+)	8	High	High	High	Low	Low
Availability of external knowledge sources	(+)	7	High	High	High	High	High
Code clarity	(+)	8	High	Low	Medium	Low	Low
Integration effort							
Adequacy of features	(+)	10	5	1	3	3	4
Adequacy of knowledge extraction	(+)	9	High	Low	Low	Medium	Medium
Adequacy of naming conventions	(+)	5	Low	Medium	High	Low	Low
Adequacy of the implementation language	(+)	7	Medium	High	Low	Low	Low
Knowledge clash	(-)	7	High	High	High	High	High
Adaptation to the reasoner	(+)	7	High	High	Low	Low	Low
Necessity of bridge terms	(-)	6	Medium	Medium	Medium	Medium	Medium
Reliability							
Availability of tests	(+)	8	Low	Low	Low	Low	Low
Former evaluation	(+)	8	Unknown	Unknown	Unknown	Unknown	Unknown
Theoretical support	(+)	10	High	High	High	High	High
Development team reputation	(+)	8	High	High	High	Low	Low
Purpose reliability	(+)	3	High	Medium	Medium	Low	Low
Practical support	(+)	7	Medium	Unknown	Unknown	Unknown	Unknown

Table 44. Analysis of the candidate mereology ontologies

ANNEX III. Hands-on Experiments in using the NeOn Watson Plug-in

In this annex we describe the hands-on experiments in using the NeOn Watson plug-in in two different use cases:

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

Experiments were performed by three ontology engineers, and their experiences were discussed and compared. Here we provide a summary of their final conclusions.

Methodological guidelines described in Section 10.5 have been inspired by these hands-on experiments in using the NeOn Watson plug-in in two use-cases.

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.

The ontology extension activity was easy to perform because the European Research Project domain is well-covered by on-line 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 is subclass of Report”, in others “Report is subclass of 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 is subclass of Book”.
- In some cases, on-line 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.

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 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 paella on festival days which is big enough to feed everybody.

INGREDIENTS:

Small onion, finely chopped	Green pepper, finely chopped
Red pepper, boiled until soft and then cut into long thin strips	Medium-sized tomatoes, skinned and finely chopped
Carrots, finely chopped	Peas, cooked
Prawns	Small clams
Squid	Mussels
Rice	Garlic, coarsely chopped
Saffron	Parsley, finely chopped
Olive oil	Water

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 plug-in. 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).

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.

ANNEX IV. Ontology Requirements Specification: User Study 1

Guideline-Instantiation-1

Sequence of tasks for the ontology requirements specification activity, including techniques and tools.

1. Identify Purpose and Scope for the ontology.
2. Identify Intended Uses.
3. Identify Intended Users.
4. Gather requirements for the ontology.
 - a. Technique: Exploit scenarios and use cases (using templates, which you should create).
 - b. Tool: You can use a wiki (if you want) for writing the informal requirements.
5. Write Competency Questions (CQs).
 - a. Technique/Approach: Top-Down.
 - b. Tool: MindMap Tool to write the CQs.
6. Group CQs.
 - a. Technique: To choose between:
 - i. Card Sorting.
 - ii. Clustering NL sentences.
 - b. Tool: MindMap Tool to represent the groups of CQs.
7. Validate CQs.
8. Extract Terminology and Frequency.
 - a. Technique: To choose between:
 - iii. Manual extract terms and count their appearance number in CQs.
 - iv. Automatic: using terminology extraction and frequency techniques and tools.

Table 45. Instantiation 1 of the preliminary methodological guidelines for Ontology Requirements Specification

Guideline-Instantiation-2

Sequence of tasks for the ontology requirements specification activity, including techniques and tools.

1. Identify Purpose and Scope for the ontology.
2. Identify Intended Uses.
3. Identify Intended Users.
4. Gather requirements for the ontology.
 - a. Technique: Brainstorming.
 - b. Tool: You can use a wiki (if you want) for writing the informal requirements.
5. Write Competency Questions (CQs).
 - a. Technique/Approach: Bottom-Up.
 - b. Tool: Excel for writing the CQs.
6. Group CQs.
 - a. Technique: To choose between:
 - i. Card Sorting.
 - ii. Clustering NL sentences.
 - b. Tool: Excel for represent the groups of CQs.
7. Validate CQs.
8. Give Priority to CQs.
9. Extract Terminology and Frequency.
 - a. Technique: To choose between:
 - i. Manual extract terms and count their appearance number in CQs.
 - ii. Automatic: using terminology extraction and frequency techniques and tools.

Table 46. Instantiation 2 of the preliminary methodological guidelines for Ontology Requirements Specification

Questionnaire

Questionnaire about the preliminary methodological guidelines proposed to carry out the ontology requirements specification activity.

Based on the guidelines instantiation you followed (Instantiation 1 or Instantiation 2), please answer each question in detail and be honest! Thank you very much.

General issues:

1. Are the proposed guidelines well explained?
2. Is more detail needed in the guidelines? If so, in which sense?, and in which tasks?
3. In your opinion, are these guidelines complete? If not, what is missing?
4. Do you think more techniques and tools should be provided?
5. Did you miss an integrated tool for carrying out the proposed tasks?
6. How we can improve the proposed guidelines?

Guidelines Instantiations 1 and 2. Tasks 1-3:

1. How difficult was to carry out these tasks?
2. Did you expect more guidelines for carrying out these tasks? If so, what kind of guidelines did you miss? And in which tasks?

Guidelines Instantiations 1 and 2. Task 4:

1. How difficult was to carry out this task?
2. Was the proposed technique useful for you? If not, please explain why.
3. Was the proposed tool useful for you? If not, please explain why.

Guidelines Instantiations 1 and 2. Task 5:

1. How difficult was to carry out this task?
2. Was the proposed technique useful for you? If not, please explain why.
3. Was the proposed tool useful for you? If not, please explain why.

Guidelines Instantiations 1 and 2. Task 6:

1. How difficult was to carry out this task?
2. Was the proposed technique useful for you? If not, please explain why.
3. Was the proposed tool useful for you? If not, please explain why.
4. If you chose the clustering NL sentences technique, how difficult was to apply such technique? Did you find tools for applying the technique?
Please explain in detail how you carried out this task.
5. Do you think to group CQs is useful? If so, please explain cases in which you think the classification of CQs would be useful.

Guidelines Instantiations 1 and 2. Task 7:

1. How difficult was to carry out this task?
2. Please explain in detail how you carried out the task, which criteria you used, and why you chose those criteria.

Questionnaire

Guidelines Instantiation 1. Task 8:

1. How difficult was to carry out this task?
2. Was the proposed technique useful for you? If not, please explain why.
3. If you chose the proposed automatic technique, how difficult was to apply such technique? Did you find tools for applying the technique?
Please explain in detail how you carried out this task.
4. Do you think to extract the terminology and its frequency is useful? Please explain cases in which you think so.

Guidelines Instantiation 2. Task 8:

1. How difficult was to carry out this task?
2. Please explain in detail how you carried out the task, which criteria you used, and why you chose those criteria.

Guidelines Instantiation 2. Task 9:

1. How difficult was to carry out this task?
2. Was the proposed technique useful for you? If not, please explain why.
3. If you chose the proposed automatic technique, how difficult was to apply such technique? Did you find tools for applying the technique?
Please explain in detail how you carried out this task.
4. Do you think to extract the terminology and its frequency is useful? Please explain cases in which you think so.

Table 47. Questionnaire about the preliminary methodological guidelines for Ontology Requirements Specification

ANNEX V. Ontology Requirements Specification: User Study 2

Questionnaire
<p>Questionnaire about the proposed methodological guidelines to carry out the ontology requirements specification activity.</p> <p>Please answer each question in detail and be honest! Thank you very much.</p> <p><i>General comments:</i></p> <ol style="list-style-type: none"> 1. Are the proposed guidelines well explained? 2. Is more detail needed in the guidelines? If so, please explain in detail in which sense and in which tasks. 3. In your opinion, are these guidelines complete? If not, what is missing? 4. Do you think more techniques and tools should be provided? 5. Did you miss an integrated tool for carrying out the proposed tasks? 6. How we can improve the proposed guidelines? 7. Did you find useful the ontology requirement guidelines? 8. Do you think you will use again the proposed guidelines for the ontology requirements specification? 9. Did you find useful to write the ontology requirements specification before going into the ontology development? 10. Do you think you will create ontology requirements specifications?

Table 48. Questionnaire about the methodological guidelines for Ontology Requirements Specification

ANNEX VI. Ontology Network Life Cycle Establishment: User Study 1

Questionnaire

Questionnaire about the proposed guidelines to decide which ontology network life cycle model is the most appropriate for their ontology network and which concrete activities should be carried out in their ontology network life cycle.

Please answer each question with detail and be honest! Thank you very much.

General issues:

1. Are the proposed guidelines well explained?
2. Is more detail needed in the guidelines?
3. In your opinion, are these guidelines complete? If not, what is missing?

Guidelines. Step 1:

1. How difficult was it to establish the requirements for your ontology?

Guidelines. Step 2:

1. How difficult was it to select the ontology network life cycle model?
2. How useful was the proposed decision tree?
3. If you needed to define a new ontology network life cycle model (not included in the current collection), please explain why.
4. Was the collection of ontology network life cycle models well explained? Is more detail needed in the explanation of each model?

Guidelines. Step 3:

1. If you have developed more than five ontologies before this experiment, can you easily identify the activities needed for your project based on the "Required-If Applicable" activities? If not, please explain why.
2. If you have not developed more than five ontologies before this experiment, was useful the set of natural language questions for identifying the activities needed for your project? If not, please explain why.
3. Is the NeOn Glossary well explained? Is something missing?

Guidelines. Step 4:

1. Which kind of detailed explanation did you expect in this step?

Guidelines. Step 5:

1. Which kind of detailed explanation did you expect in this step?

Final Comments:

1. How we can improve the proposed guidelines?
2. How the activity of establishing the life cycle for a concrete ontology network could be carried out faster?

Table 49. Questionnaire about the methodological guidelines for establishing the Ontology Network Life Cycle

ANNEX VII. Ontology Network Life Cycle Establishment: User Study 2

Questionnaire

Questionnaire about the proposed guidelines to decide which ontology network life cycle model is the most appropriate for their ontology network and which concrete activities should be carried out in their ontology network life cycle.

Please answer each question with detail and be honest! Thank you very much.

Guidelines. Step 2:

1. How difficult was to select the ontology network life cycle model?
2. How useful was the proposed decision tree?
3. If you needed to define a new ontology network life cycle model (not included in the current collection), please explain why.
4. Was the collection of ontology network life cycle model enough explained? Is more detail needed in the explanation of each model?

Guidelines. Step 3:

1. If you have developed more than five ontologies before this experiment, can you easily identified the activities needed for your project based on the “Required-If Applicable” activities? If not, please explain why.
2. If you have not developed more than five ontologies before this experiment, was useful the set of natural language questions for identified the optional activities needed for your project? If not, please explain why.
3. Is the NeOn Glossary well explained? Is something missing?

Guidelines. Step 4:

1. Which kind of detailed explanation you expected in this step?

Guidelines. Step 5:

1. Which kind of detailed explanation you expected in this step?

General Comments:

1. Are the proposed guidelines well explained?
2. Is more detail needed in the guidelines?
3. How we can improve the proposed guidelines?
4. How the activity of establishing the life cycle for a concrete ontology network could be carried out faster?

Table 50. Questionnaire about the methodological guidelines for establishing the Ontology Network Life Cycle

ANNEX VIII. Scheduling using gOntt: User Study

Questionnaire

Questionnaire about the gOntt plug-in.

Please answer each question honestly! Thank you very much.

Background Knowledge:

1. Are you familiar with software engineering?
 - Yes, I work on software engineering
 - I know it but I am not consider myself an expert
 - No really familiar
2. Are you familiar with ontology engineering?
 - Yes, I work on ontology engineering
 - I know it but I am not consider myself an expert
 - No really familiar
3. How many ontologies did you build before this tutorial?
 - None
 - Less than 5
 - Less than 10
 - Too many
4. Which is the first thing you do when building ontologies?
 - I go directly to the ontology editors
 - I first try to come up with ontology requirements with domain experts and users
 - I find resources to reuse
 - Other:

Experiences using gOntt:

5. Did you find the use of the gOntt plug-in:
 - Easy?
 - Ok?
 - Too complicated?
 Please explain your answer:
6. Did you find useful the gOntt plug-in for scheduling ontology development projects?
 - Yes
 - No
 Please explain your answer:
7. Do you think a previous training is needed for using and understanding the gOntt plug-in?
 - Yes
 - No
 Please explain your answer:

Questionnaire

8. Do you think the wizard for creating a preliminary plan for your development in the gOntt plug-in is:
 - Easy?
 - Ok?
 - Too complicated?Please explain your answer:
9. Do you think the way for modifying the preliminary plan proposed by the gOntt plug-in is:
 - Easy?
 - Ok?
 - Too complicated?Please explain your answer:
10. Do you think the idea of including methodological guidelines in the gOntt plug-in is:
 - A good and useful idea?
 - Just a nice functionality?
 - A useless idea?Please explain your answer:
11. Do you think the idea of including the possibility of triggering gOntt plug-ins from the gOntt plug-in is:
 - A good and useful idea?
 - Just a nice functionality?
 - A useless idea?Please explain your answer:
12. In your ontology developments, do you think you will use again the gOntt plug-in?
 - Yes, always
 - Yes, sometimes
 - NoPlease explain your answer:
13. In general, do you think the gOntt plug-in is:
 - Very useful?
 - A nice gadget?
 - Not good enough?Please explain your answer:
14. Do you recommend the gOntt plug-in to other colleagues involved in ontology development projects?
 - Yes, always
 - Yes, sometimes
 - NoPlease explain your answer:
15. If you have any other comments or suggestions about the gOntt plug-in, please write them here.

Table 51. Questionnaire about the use of the gOntt plug-in