NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D5.3.2 Revision and Extension of the NeOn Development Process and Ontology Life Cycle

Deliverable Co-ordinator: **Mari Carmen Suárez-Figueroa**

Deliverable Co-ordinating Institution: **UPM**

Other Authors: **Mariano Fernández-López (CEU), Asunción Gómez-Pérez (UPM), Klaas Dellschaft (UKO-LD), Holger Lewen (UKARL), and Martin Dzbor (OU)**

This deliverable presents the revision and extension of the ontology network development process, the ontology network life cycle models and the ontology network life cycle included in deliverable D5.3.1. The main contributions included in this document are (1) the improved version of the NeOn Glossary, (2) the updated of the collection of ontology network life cycle models, (3) the enhancement of the guidelines for scheduling ontology network projects and (4) the proposal of a NeOn plug-in, called gOntt, for helping people to schedule ontology projects.

Document Identifier:	NEON/2009/D5.3.2/v1.0	Date due:	November 30, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	November 30, 2008
Project start date:	March 1, 2006	Version:	V1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

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

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

Work package participants

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

UPM

UKO-LD

UKARL

OU

Change Log

Version	Date	Amended by	Changes
0.1	30-06-2008	Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez	First draft
0.2	30-08-2008	Mari Carmen Suárez-Figueroa	Update of ontology network life cycle models
0.3	15-09-2008	Mari Carmen Suárez-Figueroa	Update of guidelines for scheduling
0.4	15-10-2008	Mari Carmen Suárez-Figueroa	Inclusion of gOntt description
0.5	28-10-2008	Mari Carmen Suárez-Figueroa	General revision and updates
0.6	10-11-2008	Holger Lewen	Inclusion of relations between processes and activities and NeOn plug-ins
	11-11-2008		Inclusion of glossary feedback procedure
0.65	14-11-2008	Mari Carmen Suárez-Figueroa	Update of the NeOn Glossary of Activities
0.7	19-11-2008	Mariano Fernández-López	Internal revision
0.75	20-11-2008	Holger Lewen	New version of contribution taking into account internal revision
	21-11-2008	Klaas Dellschaft	New version of contribution taking into account internal revision
0.8	24-11-2008	Mari Carmen Suárez-Figueroa	Revision of ontology network life cycle models taking into account internal revision
	25-11-2008	Mari Carmen Suárez-Figueroa	Revision of guidelines for scheduling taking into account internal revision
	26-11-2008	Mari Carmen Suárez-Figueroa	Revision of gOntt description taking into account internal revision
0.85	28-11-2008	Mariano Fernández-López Mari Carmen Suárez-Figueroa	General revision and updates
0.9	1-12-2008	Mari Carmen Suárez-Figueroa	General revision and updates
0.91	10-12-2008	Asunción Gómez-Pérez	General revision and updates
0.92	19-12-2008	Martin Dzbor	Review, modification, corrections, proposals for minor additions. Q.A. revision
0.93	23-12-2008	Mari Carmen Suárez-Figueroa	General revision and updates

0.94	6-1-2009	Rosario Plaza	English general revision
1.0	10-12-2009	Mari Carmen Suárez-Figueroa	Updates and final version

Executive Summary

Methodologies should involve the following issues: development process, life cycle models and life cycle, methods, techniques and tools to be used during the ontology building. Thus, within the NeOn project and inside WP5, we are investigating such issues.

In deliverable D5.3.1 [10] we dealt with the three first issues: development process, life cycle models and life cycle, and we presented the following results:

1. The NeOn Glossary of Activities Version 1, which identifies and defines the activities potentially involved in the ontology network construction.

2. The first collection of theoretical ontology network life cycle models, based on those models defined in the Software Engineering field and taking into account the specific features of the ontology network development.

3. Guidelines for obtaining the concrete life cycle for an ontology network mainly based on two decision trees: (a) one for selecting the ontology network life cycle model most appropriate for the concrete case and (b) another for selecting which activities, from the NeOn Glossary of Activities, should be carried out.

4. The identification and description of complex scenarios for building network of ontologies collaboratively with special emphasis in reuse, reengineering and merging ontological and non-ontological resources.

Thus, the main goal of this deliverable is to present a revision and an extension of the methodological issues presented in deliverable D5.3.1 [10].

Concretely, this deliverable deals with the update of the glossary of activities involved in the ontology network development process, the update of the ontology network life cycle models, and the improvements of methodological guidelines for establishing the ontology network life cycle and the schedule for the ontology network development project.

The scope and main contributions of this deliverable are:

1. The update on the NeOn Glossary of Processes and Activities, and a summary of the relationships between such processes and activities and the NeOn plug-ins.
2. A new version of the collection of the ontology network life cycle models, and the relationships between scenarios for building ontology networks and ontology network life cycle models.
3. Methodological guidelines for scheduling ontology network projects.
4. The main functionalities of the gOntt plug-in, which will be the technological support for the scheduling activity.

Table of Contents

List of Tables

List of Figures

1. Introduction

The Semantic Web of the future will be characterized by the use of a very large number of ontologies embedded in ontology networks built collaboratively by distributed teams, where an ontology network or a network of ontologies is defined as a collection of ontologies (called networked ontologies) related together through a variety of different relationships such as mapping, modularization, version, or dependency relationships [4]. So, future Semantic Web applications will be based on networks of contextualized ontologies, which will be in continuous evolution. Such networks could include ontologies that already exist or they could be developed by reusing either other ontologies or non-ontological but knowledge-aware resources (e.g., thesauri, lexicons, text corpora, DBs, UML diagrams, etc.) built by others [2].

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

Methodologies should involve the following issues: development process, life cycle models and life cycle, methods, techniques and tools to be used during the ontology building. In the NeOn project, and in particular in WP5, we are investigating these issues and present them in the following technical reports:

- NeOn Deliverable D5.3.1 [10], which dealt with the first three issues: development process, life cycle models and life cycle.

- NeOn Deliverable D5.4.1 [11], which dealt with methods, techniques and tools for different processes and activities. Concretely, such a deliverable presented methodological guidelines for the ontology specification activity, for the reuse and reengineering of non-ontological resources, and for the reuse of ontological resources (general or common ontologies, domain ontologies, ontology statements, and ontology design patterns).

- NeOn Deliverable D5.3.2, that is this deliverable, is a revision and an extension of the results already presented in D5.3.1 [10].

As stated in deliverable D5.3.1 [10], the degree of maturity of the ontological engineering field is very low if we compare it with the Knowledge Engineering field and, especially, with the Software Engineering field. The long term goal of the Ontology Engineering field is to reach a similar degree of maturity to the degree of Software Engineering field today.

Unlike what happens in the Software Engineering field, the Ontology Engineering field has several distinct characteristics:

1. Prior to NeOn project, there was no activity glossary that could identify and define the activities that potentially could be carried out when single ontologies and ontology networks are developed. None of the best known methodologies (METHONTOLOGY, On-To-Knowledge and DILIGENT) includes a definition for the activities they propose. It is also remarkable that during the last years, new activities were identified when building ontologies, though they have no formal and precise definition for new and old activities. This situation results from a lack of standardization on the ontology engineering field terminology, in contrast with the *IEEE Standard Glossary of Software Engineering Terminology* [1] in the Software Engineering field.

   For this reason, in deliverable D5.3.1 [10] the NeOn Glossary of Activities Version 1, which identifies and defines the activities potentially involved in the ontology network construction, was built.

2. Life cycle models defined in Software Engineering have not been seriously analyzed and no new life cycle models have been proposed to date to cope with the special features of networks of ontologies. METHONTOLOGY proposes a unique type of life cycle model based on evolving prototypes for building single ontologies. On-To-Knowledge proposes an incremental and cyclic ontology life cycle model based on evolving prototypes, and DILIGENT proposes an ontology life cycle model based on evolving prototypes. The ontology engineering field lacked a set of ontology life cycle models, in contrast with the software life cycle models.

   For this reason, in deliverable D5.3.1 [10] the first collection of theoretical ontology network life cycle models was included. Such models are based on those models defined in the Software Engineering field and take into account the specific features of the ontology network development.

3. There were no guidelines that help software developers and ontology practitioners to select a specific life cycle model to create a particular ontology life cycle for their ontology projects.

   For this reason, in deliverable D5.3.1 [10] guidelines for obtaining the concrete life cycle for an ontology network were included. Such guidelines are mainly based on two decision trees: one for selecting the ontology network life cycle model most appropriate for the concrete case and another for selecting which activities, from the NeOn Glossary of Activities, should be carried out.

4. Existing methodologies do not cover more complex scenarios in which reuse and reengineering of ontological and non-ontological resources are needed.

   For this reason, in deliverable D5.3.1 [10] the identification and description of complex scenarios for building network of ontologies collaboratively with special emphasis in reuse, reengineering and merging ontological and non-ontological resources were included.

Taking into account preliminary evaluation results of the notions presented in [10], we plan this deliverable as a revision and an extension of the following issues:

- Improvement of the existing NeOn Glossary of Activities by obtaining the NeOn Glossary of Processes and Activities, including new processes and activities and relationships between processes and activities; and creation of the NeOn Resource Glossary by including other definitions (ontological resource, non-ontological resource, etc.).

- Detailed description of the scenario that involves the reuse of ontology design patterns.

- Update of the collection of ontology network life cycle models, based on preliminary evaluations, in which it was demonstrated that the distinction among the different iterative models (incremental, iterative, evolving prototyping and spiral) is not easy for software developers and ontology practitioners; and taking into account the ideas presented by Larman [6], who basically proposes the existence of two different models: waterfall and iterative-incremental.

- Establishment of the relationships between scenarios for building ontology networks and ontology network life cycle models.

- Enhancement of the existing guidelines for scheduling ontology network projects.

- Proposal of a NeOn plug-in, called gOntt, for supporting end-users in carrying out the scheduling activity for their ontology development.

1.1. Main Contributions

We have included the following results in this deliverable:

- The *NeOn Glossary Version 2*, including the NeOn Glossary of Processes and Activities and the NeOn Resource Glossary.
- The revision and update of the collection of ontology network life cycle models and of the identified scenarios; and the relationship between the identified scenarios in the NeOn methodology for building ontology networks [11] and the collection of ontology network life cycle models.
- The guidelines for scheduling an ontology network project, including the guidelines for obtaining the ontology network life cycle for a concrete ontology network.
- The description of the main requirements and functionalities of the gOntt plug-in.

1.2. Deliverable Structure

The deliverable is structured as follows:

- Chapter 2 presents the progress done with respect to the NeOn Glossary of Activities.
- Chapter 3 provides the revision of the identified scenarios for building ontology networks, the revision of the existing collection of ontology network life cycle models, and the relation between the identified scenarios for building ontology networks collaboratively and the ontology network life cycle models.
- Chapter 4 provides enhanced guidelines for obtaining the concrete life cycle for an ontology network and the scheduling of the ontology network project, described in the style proposed in [11] for methodological guidelines.
- Chapter 5 presents the main requirements and functionalities of the gOntt plug-in.
- Chapter 6 presents the conclusions and future work.

1.3. Relation with the Rest of WPs within Project

The relation between this deliverable and the other parts of WPs in the NeOn project is briefly described below:

- The NeOn Glossary of Activities includes activities related to the research being carried out in WP1 (e.g., ontology evolution, ontology modularization), WP2 (e.g., ontology localization), WP3 (e.g., ontology aligning), and WP4 (e.g., ontology customization).
- The main outcomes produced in this deliverable are being evaluated in task T5.6.
- The gOntt plug-in that helps to schedule an ontology network development will be released in the next version of the NeOn Toolkit.
- FAO, iSOCO, and ATOS as leaders of the use case WPs (WP7 and WP8, respectively) reviewed the NeOn Glossary of Activities from their practical point of view.

- The work carried out in WP7 and WP8 provides a useful feedback for methodological guidelines, and partners for these work-packages used the methodological guidelines from WP5 in their ontology developments.

2. NeOn Glossary of Process and Activities

The ***ontology network development process*** was defined in deliverable D5.3.1 [10] as the process by which user needs are translated into an ontology network. The main goal of the ontology network development process is to identify and define which activities are carried out when ontology networks are collaboratively built.

Within the NeOn consortium, it was decided to build the ***NeOn Glossary of Activities*** for unifying the terminology used by the NeOn partners. The idea was to achieve consensus on the identification and definition of the activities involved in developing ontology networks.

The consensus reaching process followed within the NeOn consortium for the identification and definition of the activities involved in the ontology network development process was presented in [10]. Such a work was conceived due to the 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*.

Thus, the first step towards the standardization of the terminology used in the Ontology Engineering field was presented in [10, 12, 13]. Such a step lies in achieving consensus on the definitions for the activities involved in the development process for ontology networks, within the NeOn consortium, and building the NeOn Glossary of Activities. From our understanding, any standardization agency such as ISO or W3C does not deal with the unification of Ontology Engineering terms. Only some ISO technical committees such as ISO/TC37/SC4 are working on the contribution of the ontologies for unifying linguistic resources.

The *NeOn Glossary of Activities Version 1* presented in deliverable D5.3.1 [10] is exclusively focused on the processes and activities involved in the ontology network development process, and it is published in the NeOn website[1].

Thus, our future aim is to standardize the NeOn Glossary of Activities, since the identification and definition of the activities involved in the development of ontology networks should be a result agreed within the Ontology Engineering field. The idea is to propose to standardization committees, such as the technical committee ISO/TC37, the standardization of the NeOn Glossary. Terminology standards help to avoid confusion by harmonizing terms, in our case activities involved in the development of ontology networks. The future standard NeOn Glossary of Activities is intended to serve as useful reference for those in the Ontology Engineering field and for those who come into contact with ontologies.

ISO TC 37 (Terminology and other language resources) has set up a task force on ontologies to clarify the terminology of this area and to propose a strategy for TC 37 in this area. It is worth mentioning that we are being invited to participate in such task force.

For this reason, we decided to promote an initiative to obtain feedback from people outside NeOn project in order to have richer definitions and a more complete glossary. Additionally, we decided to include other kinds of definitions and not only definitions for activities and processes. Thus, in Section 2.1, we explain the procedure for getting feedback about the NeOn Glossary Version 1 from the ontology engineering community, using the argumentation tool “Cicero” [3]. And in Section 2.2, we enhance the existing glossary with more activity/process definitions and with other kinds of definitions (data, metadata, etc.), which will be included in the NeOn Resource Glossary. The NeOn Glossary will include the NeOn Glossary of Activities and the NeOn Resource Glossary.

Another important issue with respect to the methodology for building ontology networks is to know the technological support we have for the different activities and processes identified in the NeOn

[1] http://www.neon-project.org/web-content/images/Publications/neonglossaryofactivities.pdf

Glossary. In this regard, we relate the activities and processes included in the NeOn Glossary with NeOn plug-ins in Section 2.3.

2.1. Getting Feedback from the Ontology Engineering Community

Important goals of having the NeOn Glossary of Processes and Activities are to have as complete as possible agreed collection of the important terms in the field of ontology engineering. Because the terminology in a dynamic field like ontology engineering is continually evolving, it is essential to establish a workflow for collecting the feedback about existing terms and the suggestion of new terms for maintaining the NeOn Glossary.

For this purpose, we set up a discussion project[2] in the Cicero argumentation tool. The discussion project is being used for collecting the feedback from the ontology engineering community. People who are interested in the glossary can find all current definitions on Wiki pages in Cicero[3]. Furthermore, a quick start guide[4] explains how to use the tool and how to get access and contribute to it. More details about Cicero and its discussion methodology are available in [3, 4].

Using Cicero for getting feedback has several advantages:

- With Cicero, all discussions are centrally collected and accessible for everyone.
- Cicero applies a specialized methodology for structuring discussions. For this methodology it was shown that Cicero facilitates efficient discussions and accelerates convergence to a solution (see [3, 9]).
- The captured discussions are part of the design rationale and 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. This can then be used for easily linking glossary definitions and relevant discussions with each other and thus for enhancing the documentation of the glossary.

During the feedback process, we can distinguish between feedback with regard 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. It is explained below how Cicero can be used in either of these cases:

1. **Proposing the inclusion of new activities** If we want to discuss whether a new activity should be included into the glossary, we have to describe the issue first (e.g. "We need a better coverage of activities related to ontology maintenance ..."). Before creating a new issue, we should check whether a relevant issue does not already exist. In either case, we can then reply to the issue and submit concrete solution proposals. In the example above, a solution proposal may be the actual description of a missing activity. The solution proposal may then be discussed together with other people by giving supporting or objecting arguments (e.g. an example of an ontology maintenance workflow may be given where this activity is relevant).

2. **Changing the description of an already existing activity** In this case, we may start by looking for the discussion where the activity was originally proposed. Once found, we can then extend the existing discussion and propose a new solution for describing the activity. If no relevant discussion can be found, we may also create a new issue (e.g. "In the glossary,

[2] http://cicero.uni-koblenz.de/wiki/index.php/Prj:NeOn_Glossary_of_Processes_and_Activities

[3] http://cicero.uni-koblenz.de/wiki/index.php/Category:Glossary_Definition

[4] http://cicero.uni-koblenz.de/wiki/index.php/Help:Quick_Start

many very similar activities exist that should be marked as synonyms of each other") to which again solution proposals and arguments can be submitted.

After setting up the discussion project in Cicero, a call for participation in providing feedback about the NeOn Glossary was sent to several mailing lists (including public-owl-dev, protege-owl, web-semantica-ayuda, semweb-spain, sti, super, and soa4all). Furthermore, a story about this initiative was published on the NeOn website[5] and flyers with the call for participation were distributed at ISWC 2008. For the future, it is planned to collect the feedback from the discussion project and use it for publishing an updated version of the glossary.

However, this initiative has had no very success. It seems that the call for participation has been not very productive. Reviewing the Cicero page, we realized that just a few people registered in the page and few people provided feedback on the current issues.

2.2. NeOn Glossary Version 2

In this deliverable we introduce the **NeOn Glossary** that includes, as Figure 1 shows, the *NeOn Glossary of Processes and Activities Version 2*, as an important and independent subset that makes use of the terminology defined in the *NeOn Resource Glossary*.

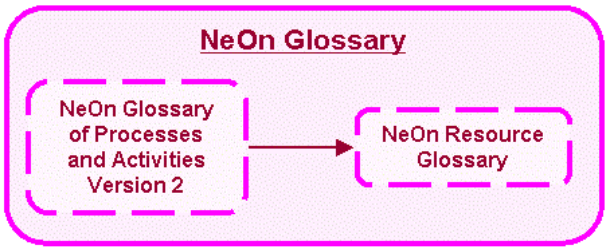


Figure 1. NeOn Glossary Composition

The ***NeOn Glossary of Process and Activities Version 2*** includes all the definitions from the NeOn Glossary of Activities Version 1 [10] and a set of new or modified definitions. In summary, the NeOn Glossary of Process and Activities Version 2 include 63 definitions.

The following new or modified definitions were originated from the methodological work carried out in WP5, concretely from the top-down work done in D5.4.1 [11].

- *Ontological Resource Reuse* is defined as the process of using available ontological resources (ontologies, modules, statements, or ontology design patterns) in the solution of different problems (e.g., the development of different ontology-based applications, the activity of ontology aligning (as background knowledge), etc.).

  We can distinguish the following types of ontological resource reuse: ontology reuse, ontology module reuse, ontology statement reuse, and ontology design pattern reuse, as Figure 2 shows.

  - *Ontology Reuse* is redefined as the process of using available ontologies in the solution of different problems.
  - *Ontology Module Reuse* is defined as the process of using available ontology modules in the solution of different problems.
  - *Ontology Statement Reuse* is defined as the process of using available ontology statements in the solution of different problems.

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

- *Ontology Design Pattern Reuse* is defined as the activity of using available ontology design patterns in the solution of different modeling problems during the development of new ontologies.

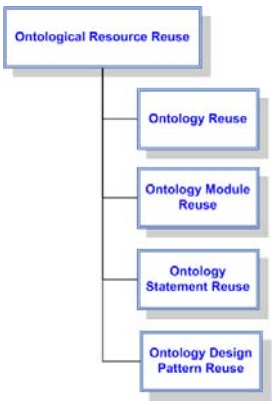


Figure 2. Ontological Resource Reuse

Non-Ontological Resource Reengineering was defined in deliverable D5.3.1 [10] as the process of retrieving and transforming an existing non-ontological resource (e.g., data bases, controlled vocabularies, etc.) into an ontology. This process could be compared with the ontology learning activity, but in such an activity the knowledge is only transformed in conceptual structures, while in the process of reengineering non-ontological resources the sources can be transformed into conceptual structures and instance data. Such a process can be divided in the following activities: non-ontological resource reverse engineering, non-ontological resource transformation and ontology forward engineering (defined in deliverable D5.3.1 [10]), shown in Figure 3, where:

- *Non-Ontological Resource Transformation* is defined as the activity of generating an ontological model at different levels of abstraction from the non-ontological resource.
- *Non-Ontological Resource Reverse Engineering* is defined as the activity of analyzing a non-ontological resource to identify its underlying components and creating a representation of the resource at higher levels of abstraction.
- Ontology Forward Engineering was defined in deliverable D5.3.1 [10].

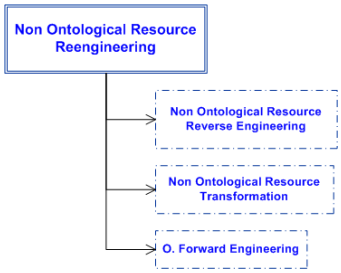


Figure 3. Non-Ontological Resource Reengineering

Figure 4 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”.

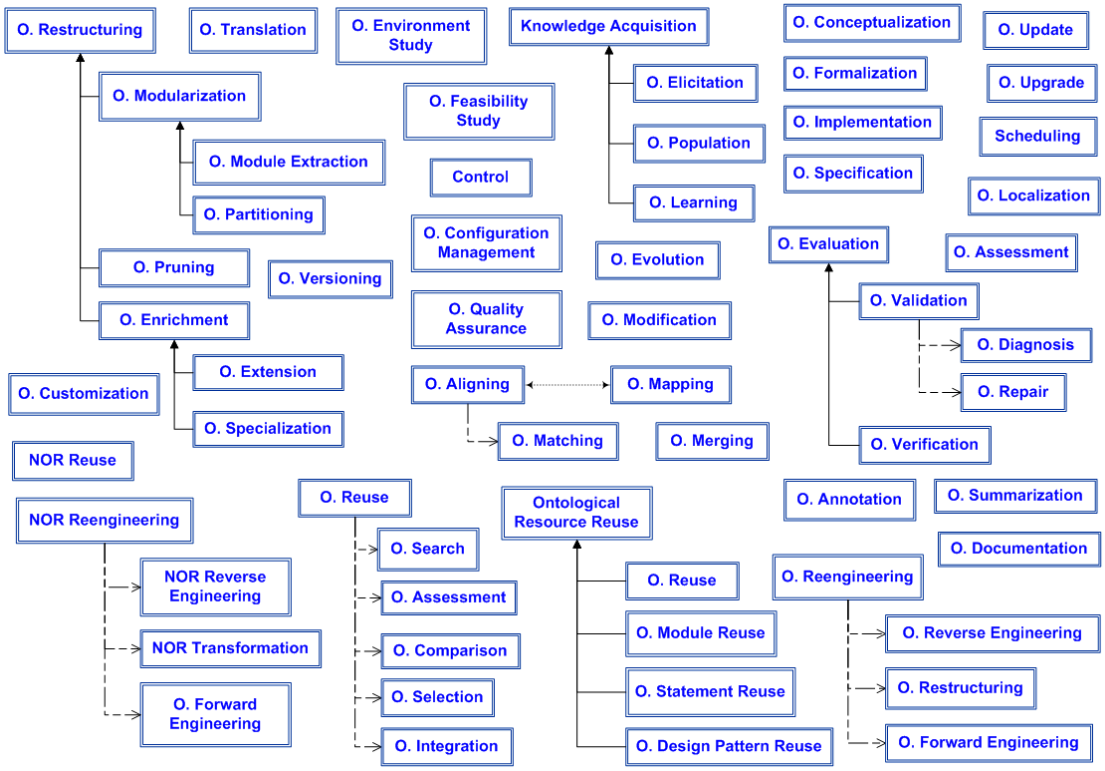


Figure 4. NeOn Glossary of Processes and Activities

The *NeOn Resource Glossary* was created to fix the problem of using different terminology for what is essentially the same purpose, and to promote good ontology design (and labelling) practices. This glossary includes the following twelve definitions in an alphabetical order, and is located in a private wiki page internal to the NeOn project. This glossary is not intended to be exhaustive, because it is still in progress within the NeOn project.

- *Concept* is defined as an abstract idea or a mental symbol, typically associated with a corresponding representation in a particular language.
- *Entity* is defined as something that has a distinct, separate existence, though it need not be a material existence. This term refers to a superset for concepts, individuals, objects, etc.
- *Individual* is defined as a specific, unique member of a set of all objects of the same type, which has been represented by a given concept; thus individuals are representations of specific but different realizations of a given abstract concept.
- *Metadata* is defined as an association of an entity (usually a label or textual explanation) to another entity (e.g., concept X can be annotated using metadata explaining its authorship).
- *Module* is defined as a specialization of concept '"ontology"', module can be seen as "a role for a particular ontology" – when module is self-contained we talk about ontology X, when X becomes a part of a larger whole, we may talk about module X in Y.
- *Modular ontology* is defined as an ontology designed by integrating at least two other ontologies in the role of modules ("ontology X importing another ontology Y may be seen as a modular ontology, for example").
- *Networked ontologies* are defined as a set of ontologies among which certain allowed properties are defined using the official NeOn Metamodel ("e.g., '"previousVersionOf"', '"importsModule"', etc.).

- *Non-ontological resource* is defined as a knowledge aware resource whose semantics has not been formalized yet by means of an ontology. Elements in this set are glossaries, dictionaries, lexicons, classification schemes and taxonomies, and thesauri.
- *Object* is defined as a physical entity, something existing in the real world, within the grasp of senses.
- *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 network (or network of ontologies)* is defined as the set of ontologies participating in some of the allowed "networking" properties.
- *Property* is defined as a specific entity that is usually associated with individuals; thus a property can be characterized in terms of individuals or types it applies to and in terms of allowed values.

2.3. Relation between Processes and Activities and NeOn Plug-ins

Methodologies can be seen as guidelines to establish and promote best practices. Using those leads to a more consistent output. Still, many developers have not adopted the use of methodologies. We believe that this is mainly due to the overhead that the use of methodologies causes. Especially at the beginning, users have to go back to the definition of the methodology frequently to check whether they follow it correctly and see which the next steps are. When methodologies are applied more frequently, they are internalized and the lookups are not needed anymore. But even then developers might forget certain steps and not follow the methodology as intended.

One solution to the overhead may be to guide the user directly in the engineering tool (e.g. the NeOn Toolkit) using tool-support for the different activities involved in the methodology. In order to cater to different requirements, activity wizards adapted to the specific needs of each user can be used. Profiles can be used to distinguish between the most common user types. Each user should be able to configure the tool-support as needed, since sometimes using a wizard/tool can cause unnecessary slowdown. Imagine, for example, a person doing label translation in the ontology and being fluent in both source and target languages. This person should not have to use the label translator plug-in and a wizard associated with this activity, since the correct translations are already known.

To clearly define the “building blocks” of the NeOn Methodology that are mainly processes and activities, a glossary of activities was created in deliverable D5.3.1 [10]. This glossary provides definitions of the terms used and thereby provides a common vocabulary. We will map existing NeOn plug-ins to activities defined in the NeOn Glossary.

Table 1 presents a mapping between processes and activities involved in the NeOn methodology and the tools that support them. It is important to note that due to the nature of the activity definition, there might never be a plug-in which covers all aspects of an activity completely. Certain tasks might only be performed by humans. Also, some plug-ins feature functionalities that can be used to assist users in multiple steps of the methodology, covering more than one activity. When reading the table, it is evident that there are some activities for which there is no plug-in, and other activities for which there are multiple plug-ins. If this is the case, plug-ins mainly focus on different aspects or techniques used to carry out the activity.

The main focus of Table 1 is to give the reader a quick way to look up which NeOn plug-in could be used or could be a help to carry out a certain process or activity.

Table 1. Mapping from Processes and Activities to Plug-ins that Support them

Process or Activity Name	NeOn Plug-in Name
Ontology Aligning	❑ AlignmentServer ❑ Plug-in for Ontology Mapping
Ontology Annotation	❑ SAFE-Plug-in
Ontology Assessment	
Ontology Comparison	
Ontology Conceptualization	❑ OntoModel ❑ Plug-in for Ontology Schema Modeling ❑ Plug-in for Textual Flogic Editor
Ontology Configuration Management	
Control	
Ontology Customization	
Ontology Diagnosis	❑ RaDON
Ontology Documentation	❑ Cicero ❑ OWLDoc ❑ Oyster ❑ Wikifactory
Ontology Elicitation	
Ontology Enrichment	❑ LeDA ❑ Plug-in for Rule Modeling ❑ Plug-in for Rule Debugging ❑ Watson
Ontology Environment Study	❑
Ontology Evaluation	❑ RaDON
Ontology Evolution	❑ Change Capturing
Ontology Extension	
Ontology Feasibility Study	
Ontology Formalization	❑ OntoModel ❑ Plug-in for Ontology Schema Modeling ❑ Plug-in for Textual Flogic Editor
Ontology Forward Engineering	
Ontology Implementation	❑ OntoModel
Ontology Integration	
Knowledge Acquisition for Ontologies	
Ontology Learning	❑ LeDA ❑ Text2Onto
Ontology Localization	❑ LabelTranslator
Ontology Matching	
Ontology Merging	❑ Modules

Process or Activity Name	NeOn Plug-in Name
Ontology Modification	
Ontology Modularization	❑ Modules
Ontology Module Extraction	❑ Modules
Ontology Partitioning	❑ Modules
Ontology Population	❑ ODEMapster Wizard
Ontology Pruning	❑ Modules
Ontology Quality Assurance	
Non-Ontological Resource Reengineering	❑ EcoreImport
Ontology Reengineering	
Ontology Restructuring	
Ontology Repair	❑ RaDON
Non-Ontological Resource Reuse	
Ontological Resource Reuse	
Ontology Reuse	❑ Oyster ❑ Watson
Ontology Module Reuse	
Ontology Statement Reuse	❑ Watson
Ontology Design Pattern Reuse	
Non-Ontological Resource Reverse Engineering	
Ontology Reverse Engineering	
Scheduling	❑ gOntt
Ontology Search	
Ontology Selection	
Ontology Specialization	
Ontology Specification	
Ontology Summarization	
Non-Ontological Resource Transformation	
Ontology Translation	
Ontology Update	❑ Change Capturing
Ontology Upgrade	❑ Change Capturing
Ontology Validation	❑ RaDON
Ontology Verification	❑ RaDON
Ontology Versioning	❑ Change Capturing ❑ OWLDiff

3. Scenarios for Building Ontology Networks and Life Cycle Models

Based on the analysis of the NeOn use cases, on the different studies carried out to revise the state of the art of ontology development and on the building of ontologies in different international and national projects (Esperonto, Knowledge Web, SEEMP, SEKT, etc.), we have detected that there are different ways or paths to build ontologies and ontology networks.

For this reason within the NeOn project, we identified different scenarios for building ontology networks, emphasizing the reuse and reengineering. To date we have identified 9 scenarios, and eight of them were described in deliverable D5.3.1 [10].

Additionally, in deliverable D5.3.1 [10] we defined an ***ontology network life cycle model*** as the framework, selected by each organization, on which to map the activities identified and defined in the NeOn Glossary of Activities to produce the ontology network life cycle.

Since there is not a unique model valid for all the ontology development projects, in deliverable D5.3.1 [10] we proposed a collection of theoretical ontology network life cycle models based on the models commonly used in Software Engineering and taking into account the specific features of the ontology network development. These ontology network life cycle models vary from trivial and simple models to difficult and complex ones.

In this section we describe in detail the scenario not described in deliverable D5.3.1 [10], we update the collection of models, and we establish the relation between scenarios and models.

3.1. Scenarios for Building Ontology Networks

The NeOn methodology [11] has identified a set of nine flexible scenarios for collaboratively building ontologies and ontology networks, with special emphasis on reusing and reengineering knowledge-aware resources (ontological and non-ontological).

Figure 5 presents the set of the 9 most plausible scenarios for building ontologies and ontology networks. The 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 Activities [12, 13]. Figure 5 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.

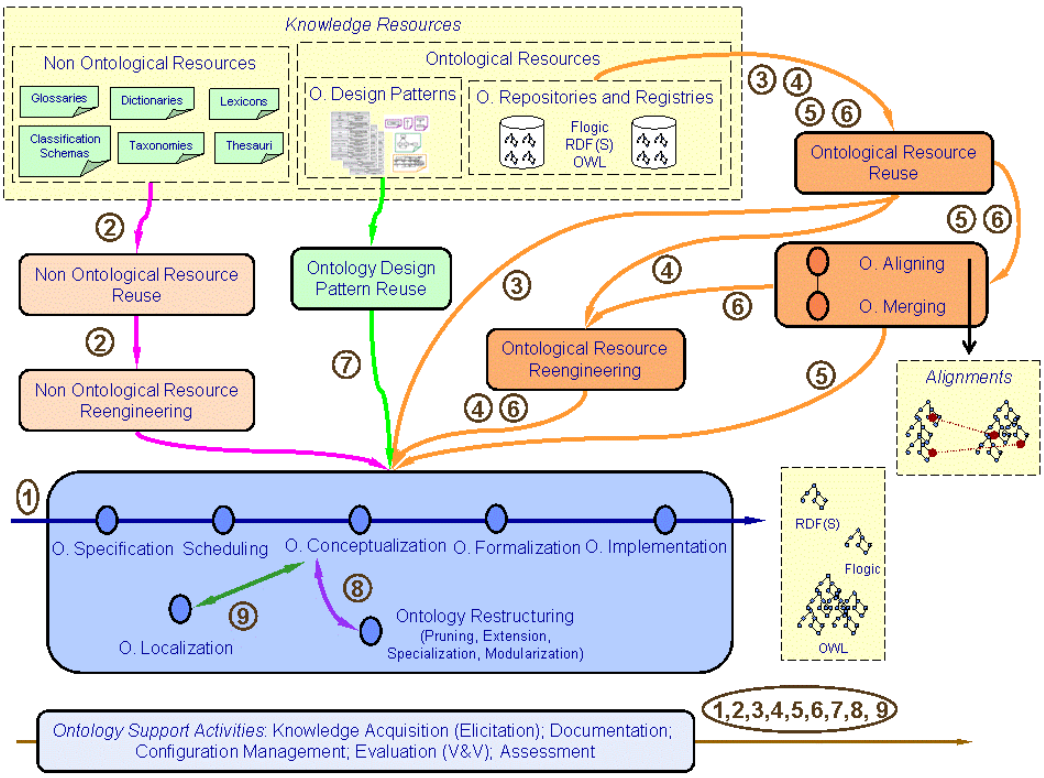


Figure 5. Scenarios for Building Ontologies and Ontology Networks

The nine identified scenarios are named in the following way:

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

Knowledge acquisition, documentation, configuration management, evaluation and assessment should be carried out all along the ontology development, that is, in any scenario used for developing the ontology network.

From this set of scenarios, we can say that scenario 1 is the most typical for building ontologies and ontology networks without reusing existing knowledge resources. Moreover, the identified scenarios within the NeOn methodology are flexible because their combination is allowed within the development of ontologies and ontology networks. It is worth 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. Indeed, as Figure 5 shows, the results of any other scenario should be integrated in the corresponding activity of scenario 1.

As mentioned above, in this section, we include the description of scenario 7 (reusing ontology design patterns) that was not described in previous deliverables. Such a description includes the

following: assumptions for the scenario, prerequisite resources, sequence of tasks to be carried out, and outcomes for the scenario.

- **Assumptions**: It is supposed that software developers and ontology practitioners want to reuse ontology design patterns (ODPs) for different purposes: to reduce modelling difficulties, to speed up the modelling process, or to check the adequacy of modelling decisions.
- **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 as well be available. Ontology design patterns repositories or catalogues are made available to users. These catalogues contain templates that fully describe design patterns.
- **Sequences of tasks**: In this scenario, software developers and ontology practitioners are working on the development of an ontology and encounter problems regarding the way in which certain knowledge should be modelled. This could happen during the ontology conceptualization activity, the ontology formalization activity, or during the ontology implementation activity. Since they are conscious of the existence of ontology design pattern repositories, they access them for finding modelling solutions. The tasks to be carried out during the *ontology design pattern reuse activity by expert users* are the following ones:
    1. To search possible ODPS to be used in a repository or catalogue of ontology design patterns. This implies to carry out a careful analysis of the different information sections included in the ontology design patterns templates in order to find possible ODPs for solving the modelling problem.
    2. To select the ontology design pattern that better matches the modelling problem. This implies to contrast the analyzed ontology design pattern templates against the real use case to find out the overlap level.
    3. To adapt the ontology design pattern 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.
    4. To integrate the ontology design pattern into the corresponding model (conceptualization, formalization or implementation).
- **Outcomes**: The principal output of this reuse activity is an ontology design pattern integrated into the ontology network being developed.

Figure 6 depicts schematically the tasks to be performed during the reuse of ontology design patterns.

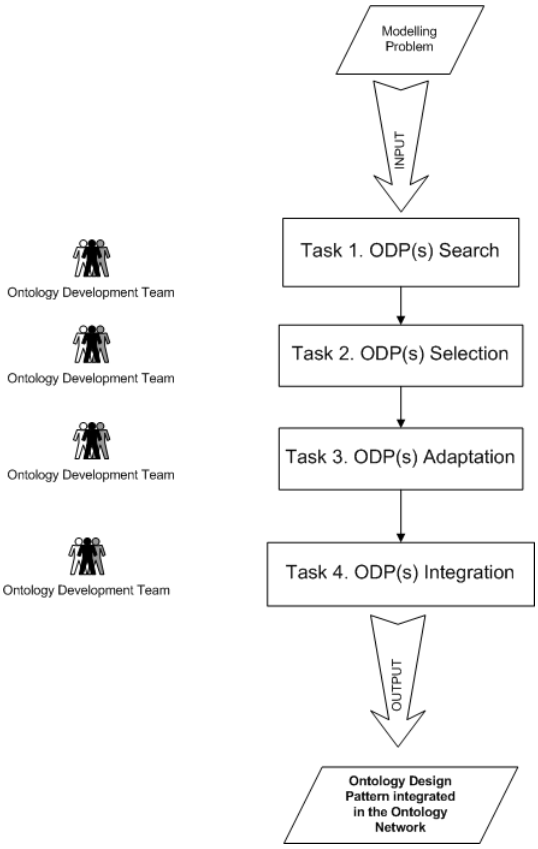


Figure 6. Tasks in the Reuse of Ontology Design Patterns

3.2. Collection of Ontology Network Life Cycle Models

In deliverable D5.3.1 [10] we defined **ontology network life cycle model** as a model to define how to develop an ontology network project (that is, how to develop and maintain an ontology network); in other words, how to organize the activities of the NeOn Glossary of Activities into phases or stages.

In general, life cycle models 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.

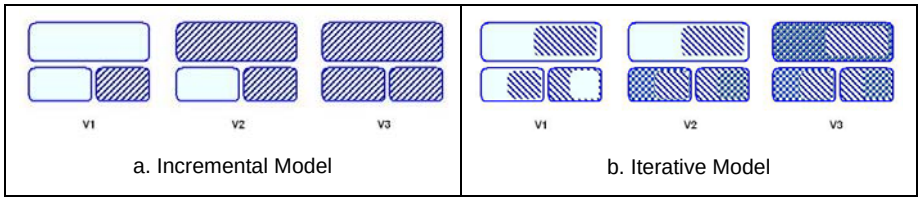
As we have already mentioned there is not a unique life cycle model valid for all ontology development projects and each life cycle model is appropriate for a concrete project, depending on several features. Therefore, to propose a unique life cycle model for all ontology network developments is not very realistic. For this reason, we proposed in deliverable D5.3.1 [10] a collection of ontology network life cycle models (version 1), based on the life cycle models described and used in Software Engineering, taking into account the specific features of the ontology network development.

In deliverable D5.3.1 [10], the proposed collection of models included the following ontology network life cycle models (ONLCMs):

- *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.

Because of the importance of reusing and reengineering knowledge resources and merging ontologies, five significantly different versions of the waterfall ontology network life cycle model have been defined and proposed: (1) *five-phase waterfall (initiation, requirement definition, design, implementation and maintenance phases)*, (2) *six-phase waterfall* that extends the previous one with a new phase in which the reuse of already implemented ontological resources is considered, (3) *six-phase + merging phase waterfall*, (4) *seven-phase waterfall* in which the six-phase model is taken as general basis and a new phase, the reengineering one, is included after the reuse phase, and (5) *seven-phase + merging phase*.

- *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 (in a concentric way). Figure 7.a shows how an ontology network grows using this model (the striped parts in the figure designate already developed parts).
- *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. Figure 7.b shows how the ontology network is developed following this model (the striped parts designate the developed parts).



a. Incremental Model b. Iterative Model

Figure 7. 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, 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 preliminary evaluations of the uses of the different proposed models in the collection presented in deliverable D5.3.1 [10], it was obvious that software developers and ontology practitioners 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 referred to which model should be chosen when the requirements are not well understood, fully captured, and when the development should carry out by parts.

Thus, (1) taking into account these preliminary evaluation results in which it was demonstrated that the distinction among the different iterative models (incremental, iterative, evolving prototyping and spiral) is not easy for software developers and ontology practitioners; (2) taking into account the ideas presented by Larman [6], who proposed the existence of two different models: waterfall and iterative-incremental, and (3) taking into account the experience we had in ontology development in different research projects (Esperonto[6], Knowledge Web[7], SEEMP[8], etc.) in which the main

[6] http://esperonto.net

models used were those based on waterfall ideas and on iterative ones, we decided to use these ideas valid in software engineering and apply them in ontology engineering modifying the existing proposed collection of ONLCMs in deliverable D5.3.1 [10] by means of committing to only two different ONLCMs:

- *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 [6], we propose here a unique iterative-incremental ONLCM. This approach is simpler, and thus, it will be easier to introduce in the ontology development by software developers and ontology practitioners.

In this section we propose the revised collection of ONLCMs, which includes the enhanced waterfall model and the new iterative-incremental model, in line with the rationale described above.

3.2.1. Waterfall Ontology Network Life Cycle Models

The main characteristic of the proposed waterfall life cycle model family 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 proposed waterfall ontology network life cycle model is that the requirements are completely known, without ambiguities and unchangeable at the beginning of the ontology network development. That is, the set of requirements is closed.

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 be 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 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 technical perspective) and the assessment (from user and need perspectives) of the different phase outputs, project and configuration management and documentation.

Taking into account the importance of knowledge resources reuse and reengineering and ontology merging, we define and propose the following five different versions of the waterfall ontology network life cycle model.

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

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

- ***Four-phase waterfall ontology network life cycle model***.

  This proposed 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 proposed model is shown in Figure 8, 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) [11], including the requisites 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 cannot 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 later 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, in this case. Additionally, in this phase the generation of new versions for the ontology network should be also carried out.

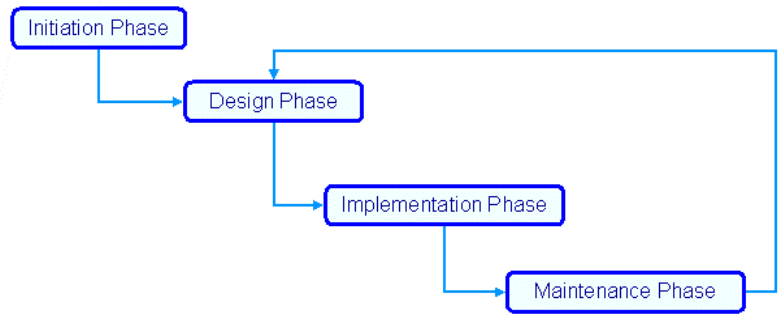


Figure 8. Four-Phase Waterfall Ontology Network Life Cycle Model

- ***Five-phase waterfall ontology network life cycle model***.

  This model, shown in Figure 9, extends the four-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 modeling 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 four-phase model.

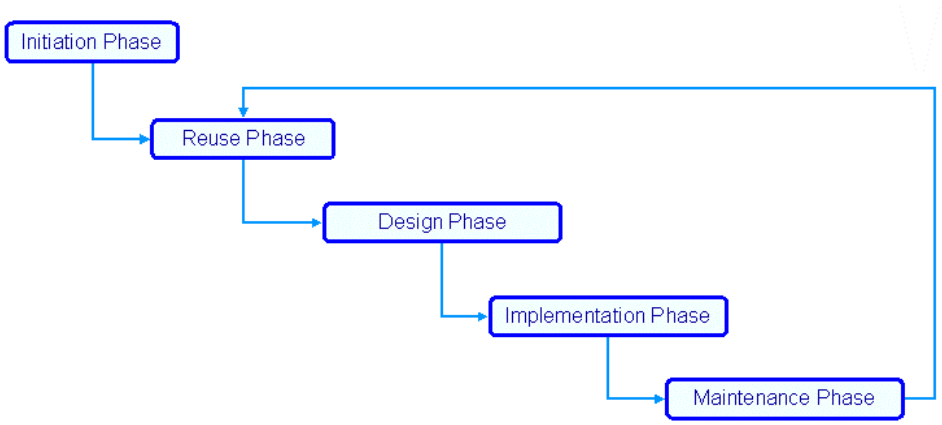


Figure 9. Five-Phase Waterfall Ontology Network Life Cycle Model

- ***Five-phase + merging phase waterfall ontology network life cycle model***.

Figure 10 shows the proposed model, which is a special case of the five-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 five-phase model.

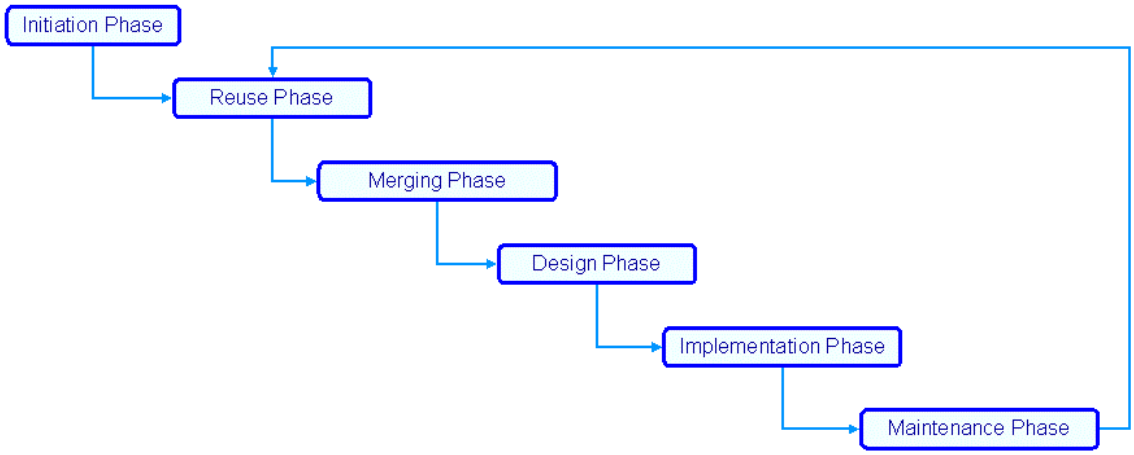


Figure 10. Five-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

- ***Six-phase waterfall ontology network life cycle model***.

In this model, shown in Figure 11, the five-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 aware 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; and the ontological resources can or cannot be reengineered which is decided by the ontology development team.

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

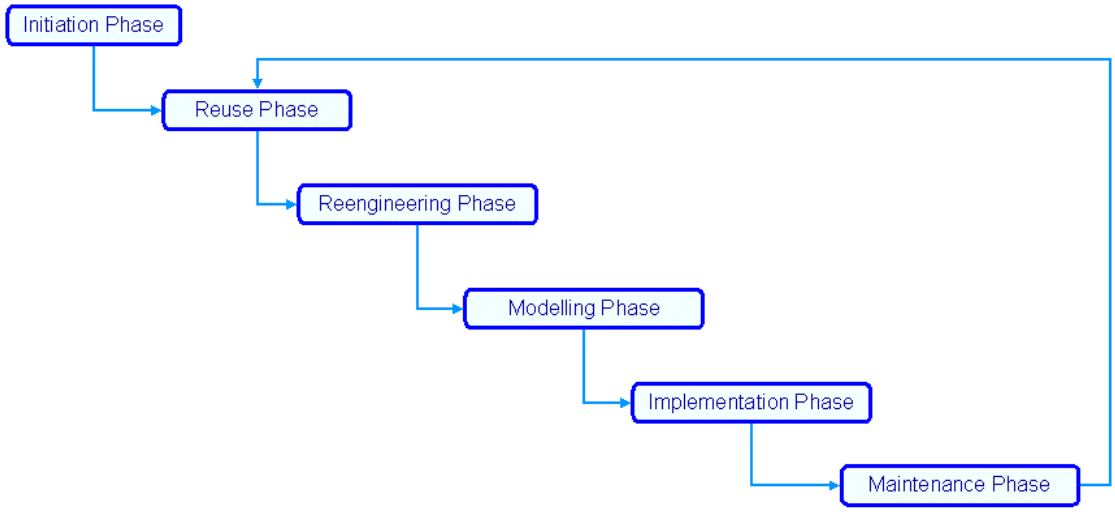


Figure 11. Six-Phase Waterfall Ontology Network Life Cycle Model

- ***Six-phase + merging phase waterfall ontology network life cycle model***.

This model, extended from the six-phase model and shown in Figure 12, includes the *Merging Phase* after the reengineering of knowledge resources (ontological and not ontological). For the other phases the purposes and outcomes are the same as those presented in the seven-phase model.

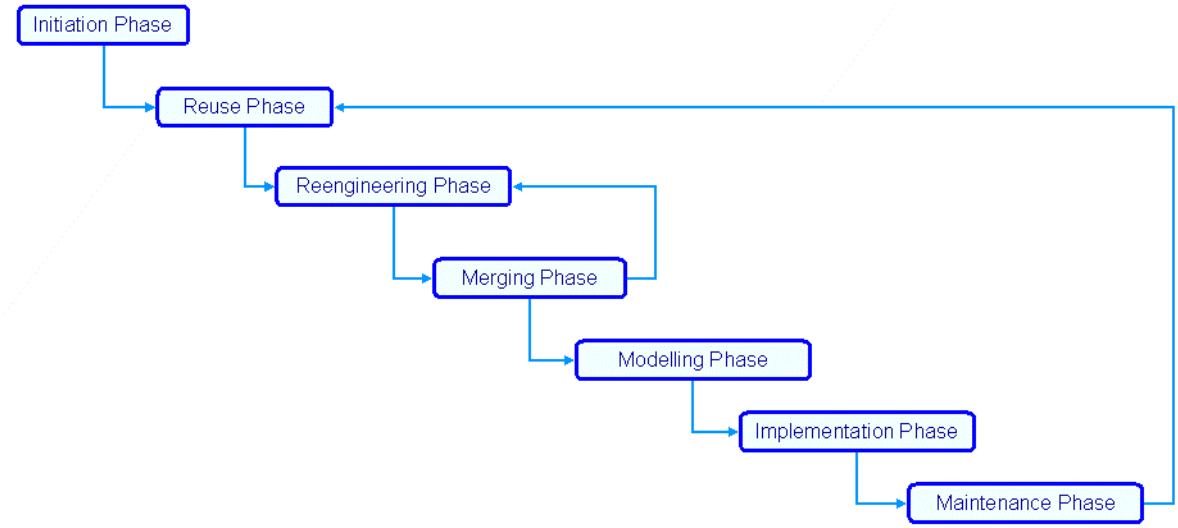


Figure 12. Six-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

3.2.2. Iterative-Incremental Ontology Network Life Cycle Models

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 using any type of waterfall model from those presented in Section 3.2.1, as shown schematically in Figure 13.

This model could be used in the following situations:

- In ontology projects with large groups of developers and including complex developments.
- In ontology project in which requirements are not completely known or can change during the ontology development.

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 in the successive improvement and extension of the ontology network by means of performing multiple iterations with cyclic feedback and adaptation. 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 existing partial ontology network could be enhanced.

The main benefit of this model is to identify and alleviate the possible risks as soon as possible. This model focuses on a set of basic requirements; from these a subset is chosen and considered in the development of the ontology network. The partial result is reviewed, and the initial set of requirements is increased and/or modified in the next iteration until the complete ontology network is developed.

It is worth mentioning that at the beginning of the ontology network project, to perform a complete requirement specification or a simple and incomplete requirement specification has an influence on the number of iterations during the ontology project.

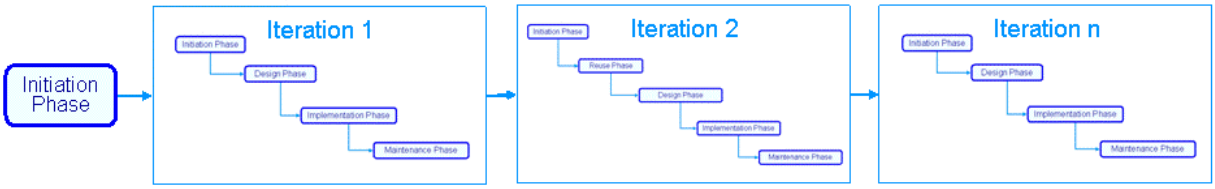


Figure 13. Schematic Vision of the Iterative-Incremental Model

Figure 13 shows the schematic vision of the iterative-incremental model where the first initiation phase 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 3.2.1. 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 perform in the next iterations.
- In the initiation phase of each iteration, the revision of the ontology network requirements and of the global plan should be carried out. Additionally, a detailed plan for the particular iteration should be performed.

3.3. Relation between Scenarios and Life Cycle Models

The set of nine flexible scenarios for collaboratively building ontologies and ontology networks, presented in Section 3.1 and previously in D5.3.1 [10], and the proposed ontology network life cycle models (waterfall and iterative-incremental), presented in Section 3.2, are intrinsically related because both (scenarios and life cycle models) have been created taking into account the importance of reusing and reengineering knowledge-aware resources (ontological and non-ontological).

This implicit relation facilitates the selection of a concrete ontology network life cycle model when a particular ontology network project is being scheduled.

As we explained in Section 3.2.1, the waterfall model has five significantly different versions and such versions have been created incrementally (that is, four-phase is the basis for five-phase, five-phase is the basis for six-phase, etc.), as Figure 14 shows.

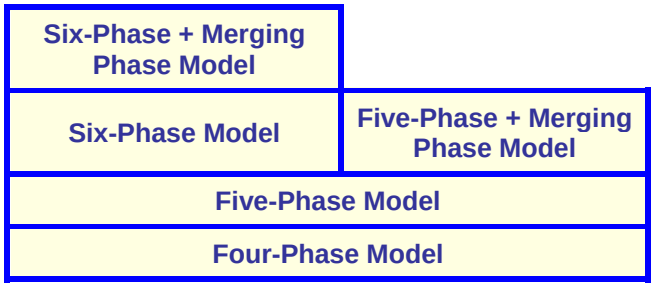


Figure 14. Pyramid of the Versions of the Waterfall Ontology Network Life Cycle Model

Table 2 summarizes the relationships between scenarios for building ontology networks and ontology network life cycle models. Scenario 1 is related to all the ontology networks life cycle models, because as explained in Section 3.1 it is an obligatory one. If a scenario is related to a concrete model version, then such a scenario is also related to the model version created on top of it. Table 2 does not show scenarios 8 and 9, because such scenarios are not directly related to any model, but to all of them. Such scenarios can be optionally related to all the life cycle models, concretely in the modeling phase.

Table 2. Relation between Scenarios and Life Cycle Models

	Four-Phase Model	*Five-Phase Model*	*Five-Phase + Merging Phase Model*	*Six-Phase Model*	*Six-Phase + Merging Phase Model*
Scenario 1	X	X	X	X	X
Scenario 2				X	X
Scenario 3		X	X	X	X
Scenario 4				X	X
Scenario 5			X		X
Scenario 6					X
Scenario 7		X	X	X	X

As explained in Section 3.2.2, 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 2 are also valid for this model.

4. Guidelines for Scheduling: Obtaining the Ontology Network Life Cycle

Scheduling [12, 13] 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 software developers and ontology practitioners 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 to guide the ontology network development, including processes and activities, their order, and time and human resources restrictions.

To establish the concrete schedule for the ontology network development, three 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) How much resources (human and time) are needed for the development of the ontology network?

The first two 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 third question is related to the inclusion of time and human resources restrictions, and its response would result in the concrete schedule for the ontology network development.

To respond to the two first questions, an initial collection of ontology network life cycle models and some guidelines for selecting the most appropriate model and choosing the processes and activities were presented in [10, 14]. Once software developers and ontology practitioners have taken these decisions, they obtain (by mapping the selected ontology network life cycle model and the selected processes and activities, and then placing in order such processes and activities) the particular life cycle for their ontology network development. That is, the ordered sequence of processes and activities to be carried out during the life of the ontology network.

After that, software developers and ontology practitioners can answer the third question by including in the ontology network life cycle information about time and people to obtain the concrete scheduling. The information about how many people should be involved in the ontology network development can be obtained using the ONTOCOM model [7]. This is a cost estimation model, whose goal is to predict the costs (expressed in person months) arising in typical ontology engineering processes.

Taking into account the update of the ontology network life cycle models, presented in Section 3.2, and the results from preliminary experiments (to be presented in deliverable D5.6.2) in which the initial guidelines presented in [10, 14] were used, in this chapter we update the scheduling guidelines. We also describe them by including the corresponding filling card and a workflow that represents the guidelines, following the same style we used in [10] for describing activity or process guidelines.

In the context of the NeOn methodology for building ontology network, we propose the filling card, presented in Figure 15, for the scheduling activity; the filling card includes the definition, goal,

inputs and outputs, who carry out the activity and when the activity should be carried out. This concrete filling card follows the filling card template presented in deliverable D5.4.1 [11].

Scheduling

Definition

Scheduling 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.

Goal

The scheduling activity states a concrete programming or scheduling to guide the ontology network development, including processes and activities, their order, and time and human resources restrictions and assignments.

Input

Ontology Requirements Specification Document (ORSD).

Output

Schedule for the ontology network development.

Who

Software developers and ontology practitioners, who form the ontology development team (ODT), in collaboration with users and domain experts.

When

This activity must be carried out after the ontology requirements specification activity.

Figure 15. Scheduling Filling Card

As stated in Figure 15, the scheduling activity must be carried out after the ontology requirements specification activity. It is worth to mention that it is advisory to carry out a quick search for knowledge-aware resources using as input the ontology requirement specification document before to carry out the scheduling activity.

The tasks for carrying out the scheduling activity can be seen in Figure 16, and are explained in detail below.

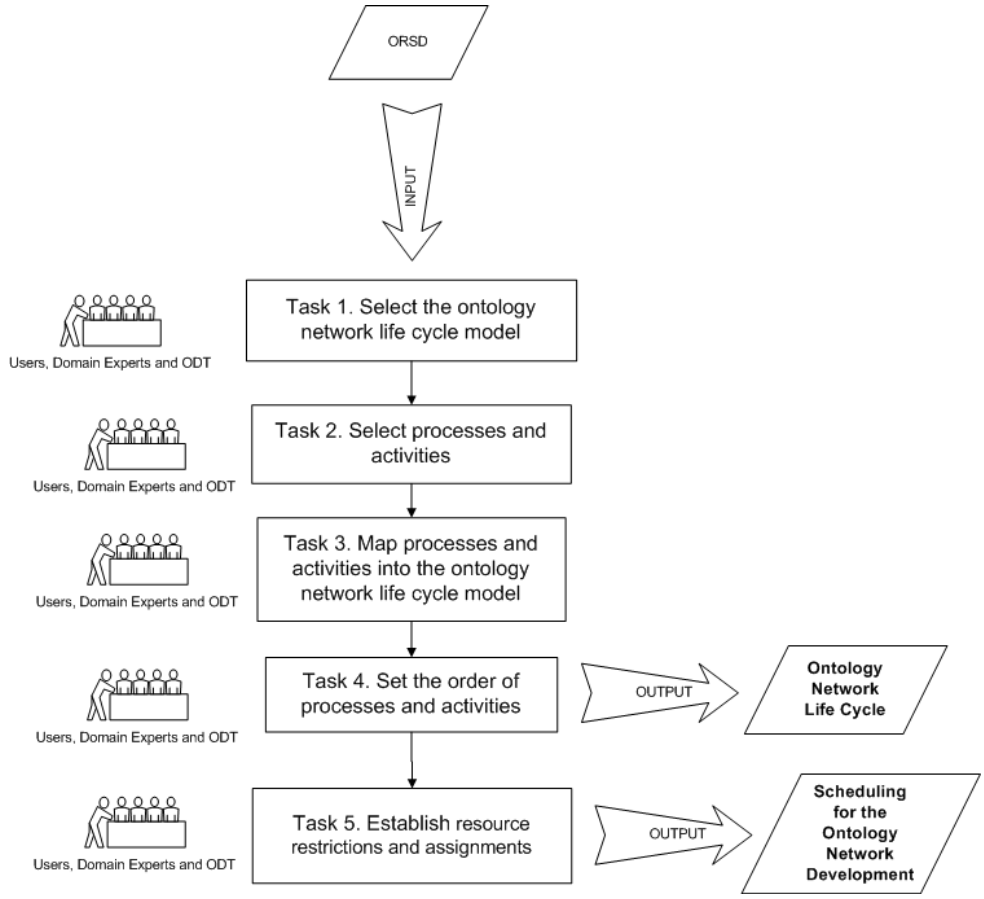


Figure 16. Tasks for Scheduling

Task 1. Select the ontology network life cycle model (ONLCM).

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 the ontology requirement specification document (ORSD) and the set of potential knowledge-aware resources to be used during the development. The actors involved in this task use the set of natural language questions related to the ontology network requirements proposed here.

The main question to be answered in this task is: “Which ontology network life cycle model should be chosen?”. Such choice should be based on software organization culture, previous experience of the team of software developers and ontology practitioners, application area, and comprehension and volatility of the ontology network requirements. Furthermore, general ontology network development requirements should be taken into account.

To help software developers and ontology practitioners to decide the most appropriate model among the ones in the collection presented in Section 3.2, the following set of natural language questions are proposed.

- Are the ontology network requirements assumed to be fully known at the beginning of the ontology network development?
    - If the answer is yes, then we can propose to choose the waterfall ontology network life cycle model.
- Is it possible that the ontology network requirements are not fully known at the beginning of the ontology network development and/or that they may change during the development?

- If the answer is yes, then we can propose to select the iterative-incremental ontology network life cycle model.

➢ Have the ontology network requirements assigned different priorities?

- If the answer is yes, then we can propose to select the iterative-incremental ontology network life cycle model.

Based on the answers to these questions, the waterfall model or the iterative-incremental are selected. However, taking into account the different model versions presented in Section 3.2, we propose the following set of natural language questions to select a particular model version.

If the iterative-incremental model is selected, it is necessary to decide among the different versions of waterfall model by means of answering these natural language questions in each iteration.

❑ Have you planned or it is said in your ontology requirements to use any existing ontological resource in your ontology network development?

- If the answer is yes, then the five-phase waterfall model should be selected.

❑ Have you planned or it is said in your ontology requirements to use ontology design patterns in your ontology network development?

- If the answer is yes, then the five-phase waterfall model should be selected.

❑ Have you planned or it is said in your ontology requirements to use and merge a set of existing ontological resources in your ontology network development?

- If the answer is yes, then the five-phase + merging phase model should be selected.

❑ Have you planned or it is said in your ontology requirements to use any non-ontological resource such as thesauri, data bases, etc. in your ontology network development?

- If the answer is yes, then the six-phase model should be selected.

❑ Have you planned or it is said in your ontology requirements to use and modify any existing ontological resource in your ontology network development?

- If the answer is yes, then the six-phase model should be selected.

❑ Have you planned or it is said in your ontology requirements to use, merge, and modify a set of existing ontological resources in your ontology network development?

- If the answer is yes, then the six-phase + merging phase model should be selected.

If software developers and ontology practitioners answer in the affirmative to several questions of those proposed here, they would obtain several candidate models, and thus the final model version to be selected should be the most specific one based on the pyramid in Figure 14 and discussed in Section 3.3.

If software developers and ontology practitioners answer in the negative to all the aforementioned questions, the final model version to be selected should be the four phase waterfall model.

Task 2. Select processes and activities.

The goal of this task is to select the set of processes and activities to be performed during the ontology network development. Users, domain experts and the ontology development team carry out this task taking as input the ontology requirement specification document (ORSD) and the set

of potential knowledge-aware resources to be used during the development. The actors involved in this task use as technique (1) a set of natural language questions proposed here, whose answers refer to a particular set of scenarios for building ontology networks, and (2) Table 3, which corresponds scenarios with processes and activities.

The natural language questions to be used in this task in order to obtain which scenarios should be followed during the ontology network development are the following ones:

- Have you planned to use any non-ontological resource such as thesauri, data bases, etc. in your ontology network development?
    - If so, then your ontology network development should follow scenario 2.
- Have you planned to use any existing ontological resource in your ontology network development?
    - If so, then your ontology network development should follow scenario 3.
- Have you planned to use and modify any existing ontological resource in your ontology network development?
    - If so, then your ontology network development should follow scenario 4.
- Have you planned to use and merge a set of existing ontological resources in your ontology network development?
    - If so, then your ontology network development should follow scenario 5.
- Have you planned to use, merge, and modify a set of existing ontological resources in your ontology network development?
    - If so, then your ontology network development should follow scenario 6.
- Have you planned to use ontology design patterns in your ontology network development?
    - If so, then your ontology network development should follow scenario 7.
- Have you planned to restructure your ontology network?
    - If so, then your ontology network development should follow scenario 8.
- Have you planned to develop your ontology network in different natural languages?
    - If so, then your ontology network development should follow scenario 9.

After answering the previous set of questions, the actors involved in this task obtain the set of scenarios applicable in the ontology development. Using such a set of scenarios, Table 3 and the table of 'Required-If Applicable' activities presented in deliverable D5.3.1 [10], software developers and ontology practitioners obtain the set of processes and activities selected for the ontology development.

Table 3 relates scenarios with processes and activities in the following way: (1) a process or an activity can be carry out in a particular scenario (S1); (2) a process or an activity can be carry out in a list of scenarios (S4, S6, S8); and (3) a process or an activity can be carry out in a set of scenarios (S1-S9).

Table 3. Correspondence between Scenarios and Processes and Activities

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 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 Matching	S5-S6
Ontology Merging	S5-S6
Ontology Modification	S1
Ontology Modularization	S4, S6, S8
Ontology Module Extraction	S4, S6, S8
Ontology Partitioning	S4, S6, S8
Ontology Population	S1-S9
Ontology Pruning	S4, S6, S8

Process or Activity Name	*Related Scenarios*
Ontology Quality Assurance	**S1-S9**
Non-Ontological Resource Reengineering	**S2**
Non-Ontological Resource Reverse Engineering	**S2**
Non-Ontological Resource Transformation	**S2**
Ontology Reengineering	**S4, S6**
Ontology Restructuring	**S4, S6, S8**
Ontology Repair	**S1-S9**
Non-Ontological Resource Reuse	**S2**
Ontology Reuse	**S3-S6**
Ontology Module Reuse	**S3-S6**
Ontology Statement Reuse	**S3-S6**
Ontology Design Pattern Reuse	**S7**
Ontology Reverse Engineering	**S4, S6**
Scheduling	**S1**
Ontology Search	**S3-S6**
Ontology Selection	**S3-S6**
Ontology Specialization	**S4, S6, S8**
Ontology Specification	**S1**
Ontology Summarization	**S1-S9**
Ontology Translation	**S1**
Ontology Update	**S1**
Ontology Upgrade	**S1**
Ontology Validation	**S1-S9**
Ontology Verification	**S1-S9**
Ontology Versioning	**S1**

Task 3. Map processes and activities into the ontology network life cycle model.

The goal of this task is to obtain the mapping between the selected processes and activities and the selected ontology network life cycle model. Users, domain experts and the ontology development team carry out this task taking as input the selected ontology network life cycle model and the selected set of processes and activities. Selected processes and activities should be carried out in a phase or stage of the selected ONLCM to fulfil the purpose and outcomes of that phase.

Software developers and ontology practitioners can use in this task Table 4 that matches the process and activity outputs against the purpose and outcomes of each phase or stage of the ONLCM, taking into account the following existing phases in our repository of ONLCMs: merging phase, reuse phase, design phase, reengineering phase, initiation phase, implementation phase, and maintenance phase.

Table 4 relates processes and activities to phases in the following way: (1) a process or an activity should be carried out in a particular phase of the selected model; or (2) a process or an activity should be carried out in all the phases of the selected model.

Table 4. Correspondence between ONLCM Phases and Processes and Activities

Process or Activity	Corresponding Phase
Ontology Aligning	**Merging Phase**
Ontology Annotation	**All Phases**
Ontology Assessment	**All Phases**
Ontology Comparison	**Reuse Phase**
Ontology Conceptualization	**Design Phase**
Ontology Configuration Management	**All Phases**
Control	**All Phases**
Ontology Customization	**Reengineering Phase**
Ontology Diagnosis	**All Phases**
Ontology Documentation	**All Phases**
Ontology Elicitation	**All Phases**
Ontology Enrichment	**Reengineering Phase**
Ontology Environment Study	**Initiation Phase**
Ontology Evaluation	**All Phases**
Ontology Evolution	**Design Phase**
Ontology Extension	**Reengineering Phase**
Ontology Feasibility Study	**Initiation Phase**
Ontology Formalization	**Design Phase**
Ontology Forward Engineering	**Reengineering Phase**
Ontology Implementation	**Implementation Phase**
Ontology Integration	**Design Phase**
Knowledge Acquisition for Ontologies	**All Phases**
Ontology Learning	**All Phases**
Ontology Localization	**Design Phase**
Ontology Matching	**Merging Phase**
Ontology Merging	**Merging Phase**

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

Task 4. Set the order of processes and activities.

After obtaining the mapping between the selected processes and activities and the selected ONLCM, software developers and ontology practitioners should place in order the selected processes and activities, obtaining in this way the ontology network life cycle.

The order in which processes and activities will be performed are determined by three major factors:

a) The selected ONLCM will dictate an initial ordering of processes and activities.

b) Processes and activities may be mapped for parallel execution rather than for serial execution.

c) The entry and exit criteria of associated processes and activities might impact on the ordering. 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. The initial ordering may always be amended, changed and updated.

Task 5. Establish resource restrictions and assignments. After obtaining the ontology network life cycle, software developers and ontology practitioners should include information about temporal scheduling and human resource assignments. The output of such task can be represented as a Gantt diagram similar to the one shown in Figure 17. As general motivation here, we decided to reuse Gantt charts, which are *de facto standard* in software project management. The main idea was not to create new representations for schedules in ontology engineering, but to adapt the existing ones (such as Gantt diagrams) to ontology developers.

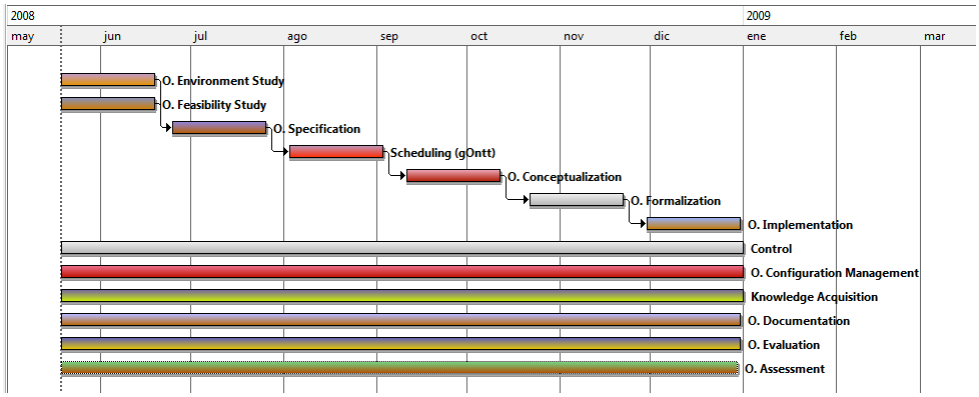


Figure 17. Example of Gantt Diagram

5. gOntt: NeOn plug-in for the Scheduling Activity

gOntt is the NeOn plug-in that will support the scheduling activity having the following main functionalities:

- To help software developers and ontology practitioners to decide which ontology network life cycle model is the most appropriate for their ontology network.
- To help software developers and ontology practitioners to decide which concrete process and activities should be carried out in the ontology network development and in which order they should be performed.
- To help software developers and ontology practitioners to include human resources and time restrictions in the ontology network life cycle.
- To inform software developers and ontology practitioners about how to carry out a particular process or activity, including methodological guidelines and a reference of which concrete NeOn plug-in should be used for each process and activity.

In summary, the **gOntt plug-in should help to schedule an ontology network development. Additionally, the gOntt plug-in should be a NeOn meta-tool with the goal of providing methodological guidelines for each process and activity and information about the existing NeOn plug-ins for each process and activity**

In this chapter, we include the main users and functionalities of the gOntt plug-in.

The gOntt plug-in should allow three different roles or type of users:

- ***gOntt Administrator***
- ***Ontology Project Manager***
- ***Ontology Development Team***

Additionally, gOntt should have two different views:

- *Administrative View* for being used by gOntt administrator.
- *Gantt View* for being used by ontology project manager and ontology development team.

The main requirements for gOntt plug-in can be divided into 4 groups: general requirements, requirements for gOntt administrator, requirements for ontology project manager, and requirements for ontology development team.

The list of requirements for gOntt plug-in were obtained (1) from previous experiences in scheduling in software engineering projects, (2) from interviews with experts in scheduling ontology projects, and (3) based on the scheduling guidelines presented in Chapter 4.

General Requirements:

- To create, modify, and delete gOntt projects, that is, both (1) default templates for scheduling that provide initial scheduling and (2) particular schedules for ontology projects.
- To save all gOntt projects in files with .got extension, this must be based on xml standard.
- To save and open .got files.
- To create and include new phases composed by processes and activities in a gOntt project.
- To create and include processes and activities (both from the NeOn Glossary or new activities not included in the glossary) in a gOntt project.
- To have different visualizations for phases, processes and activities.

- To have two different visualizations of gOntt projects: graphical view and textual view, both with the same information (activities, processes, resource assignments, order, restrictions, etc.).
    a. Graphical view and textual view should be updated at the same time.
    b. Begin and End dates should appear in the views.
    c. Graphical view should show a Gantt diagram.
    d. Phases, processes and activities should appear in the list view with indentation (that means a set of processes and activities are inside a phase; and a set of activities are inside of a process).
- To delete processes, activities and phases from a gOntt project.
- To modify process, activity and phases names in a gOntt project.
- To modify process, activity and phases order in a gOntt project.
- To create, modify and delete connections between activities, between processes, and between activities and processes.
    a. If a connection exists between two activities or two processes or between activities and processes, this means dependency, the second one cannot start until the first one finishes (sequential order).
    b. Optionally, other types of relations such as two activities/processes starting or finishing at the same time.
- To include and modify duration and starting date for processes, activities and phases.
- To have the possibility of associating plug-ins to processes and activities. Processes and activities can have more than one plug-in associated.
- To have the possibility to hide phases and see only processes and activities

Specific Requirements for gOntt Administrator:

- To include new processes and activities coming from the NeOn Glossary into gOntt plug-in.
- To include methodological guidelines for activities and processes.
- To create schedule templates to be used in the guided manner by the ontology project manager.
- To modify and delete existing schedule templates.

Specific Requirements for Ontology Project Management:

- To create ontology network project schedules from scratch, by means of introducing processes, activities, phases and their relationships and restrictions.
- To create ontology network project schedules in a guided way, by using templates that provides initial schedule. The guided way should be based on 3 wizard menus:
    - The first menu based on natural language questions to select the ontology life cycle model.
    - The second menu based on natural language questions to select processes and activities.
    - The third menu, which is optional, based on natural language questions to decide between possible optional processes and activities.
- To modify existing ontology network project schedules.

Specific Requirements for Ontology Development Team:

- To use the ontology network project schedule.
- This kind of user can not edit ontology network project schedules.

We are currently implementing the gOntt plug-in using the aforementioned requirements. We have planned the first version of gOntt plug-in for month 36.

Figure 18 and Figure 19 show respectively the appearance of gOntt plug-in in two different example situations: (1) the schedule for the reuse phase in an ontology development project, and (2) the inclusion of a new activity in an existing schedule.

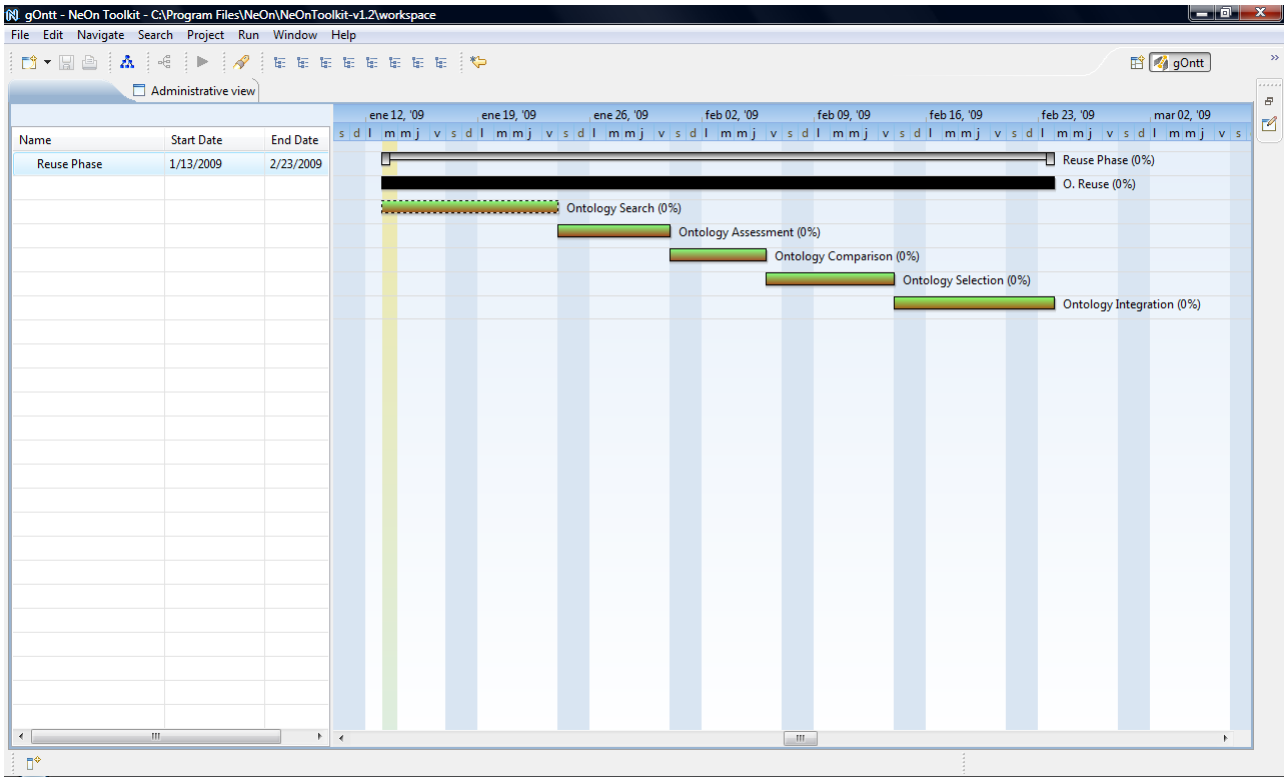


Figure 18. Screenshot of gOntt Plug-in (1)

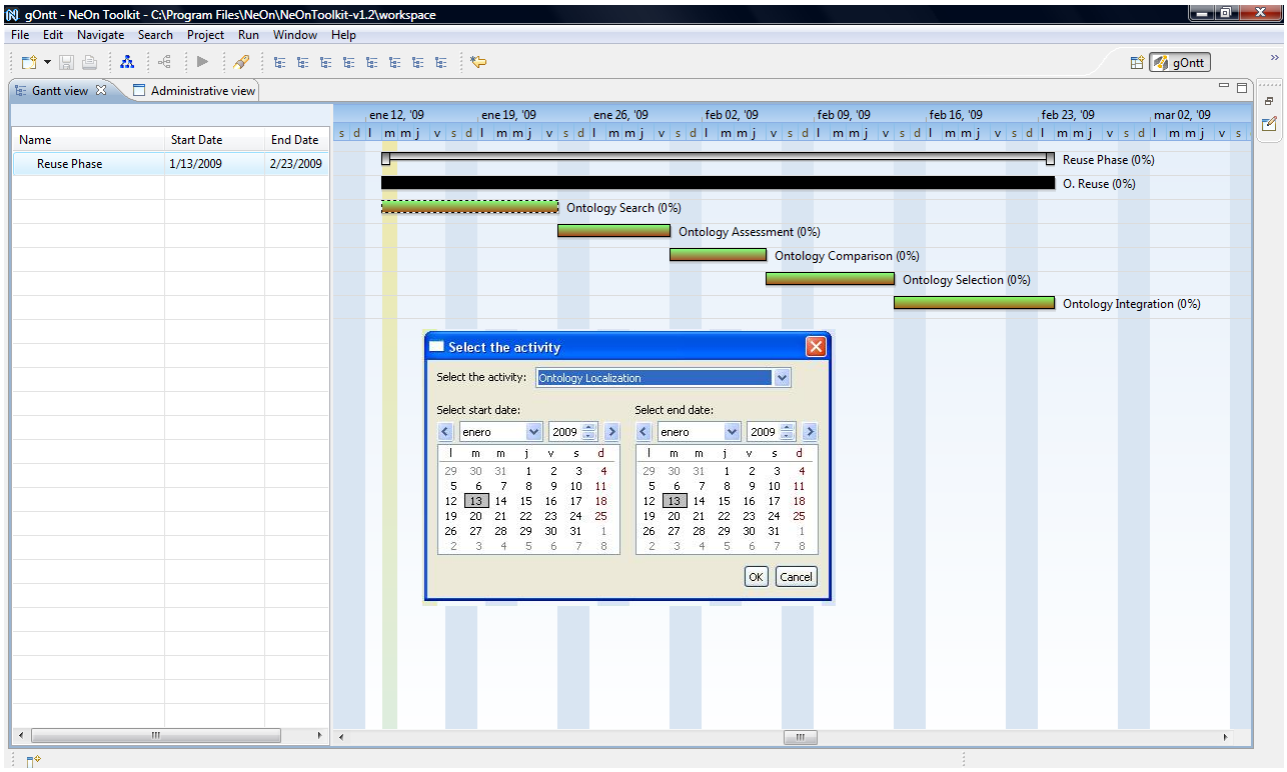


Figure 19. Screenshot of gOntt Plug-in (2)

6. Conclusions and Future Work

After analysing the state of the art in D5.3.1 [10], we can say that the degree of maturity of the Ontology Engineering field is very low if we compare it 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 degree of maturity similar to the one that the Software Engineering has today. Deliverable D5.3.1 [10] presented an advance in this sense by means of the following contributions:

- First version of the *NeOn Glossary of Activities*, which identifies and defines the activities potentially involved in the ontology network construction, as a first step for solving the lack of a standard glossary in the Ontology Engineering field.
- Identification and description of *eight scenarios for building network of ontologies* collaboratively with special emphasis in reuse, reengineering and merging ontological and non-ontological resources.
- The first collection of several *theoretical ontology network life cycle models*, based on those defined in the Software Engineering field and taking into account the specific features of the ontology network development.
- *Methodological guidelines for scheduling ontology network project and for obtaining the concrete life cycle for an ontology network*.

In this deliverable we improve the results presented in deliverable D5.3.1 [10] in the following way:

- Improvement of the existing NeOn Glossary of Activities by obtaining the NeOn Glossary of Processes and Activities, including new processes and activities and relationships between processes and activities; and creation of the NeOn Resource Glossary by including other definitions (ontological resource, non-ontological resource, etc.).

  This improved version is available in Cicero[9] in order to obtain feedback from the ontology engineering community outside NeOn project.

  As current work we have planned to propose to standardization committees, such as the technical committee ISO/TC37, the standardization of the NeOn Glossary, after collecting the feedback provided by ontology engineering community.

- Revision and update of the identified scenarios for building ontology networks, including the detailed description of the scenario that involves the reuse of ontology design patterns.

- Update of the collection of ontology network life cycle models, based on (1) preliminary evaluations with the first collection presented in D5.3.1 [10], in which it was demonstrated that the distinction among the different iterative models (incremental, iterative, evolving prototyping and spiral) is not easy for software developers and ontology practitioners; (2) experiences in different ontology developments within different projects; and (3) ideas presented by Larman [6], who basically proposes the existence of two different models: waterfall and iterative-incremental.

  The revised collection presented in this deliverable includes the enhanced waterfall model and the new iterative-incremental model.

[9] http://cicero.uni-koblenz.de/wiki/index.php/Prj:NeOn_Glossary_of_Processes_and_Activities

As future work we have planned to evaluate this current collection of ontology network life cycle models in real use cases.

- Establishment of the relationships between scenarios for building ontology networks and ontology network life cycle models. These relationships facilitate the selection of a concrete ontology network life cycle model when a particular ontology network project is being scheduled.

- Enhancement of the existing guidelines obtaining the concrete life cycle for an ontology network and the scheduling of the ontology network project, described in the style proposed in [11] for methodological guidelines.

  Such improved guidelines are based on (1) the update of the ontology network life cycle models; and (2) a set of natural language questions related to ontology requirements.

  Also as future work we have planned to evaluate the guidelines for scheduling ontology projects in real use cases.

- Proposal of a NeOn plug-in called gOntt for supporting the scheduling activity, based on the methodological guidelines presented in this deliverable.

  As current work we are developing the first version of the gOntt plug-in that will be evaluated in real use cases.

  As future work we have plan to include in the gOntt plug-in the possibility of (1) establishing human resources restrictions and establishments and (2) including the history of the development, i.e., percentage of process or activity that has been done.

References

1. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990 (Revision and redesignation of IEEE Std 792-1983).

2. NeOn Consortium. NeOn: Lifecycle Support for Networked Ontologies. *NeOn Technical Annex*. 2006.

3. K. Dellschaft, H. Engelbrecht, J. Monte Barreto, S. Rutenbeck, S. Staab. *Cicero: Tracking Design Rationale in Collaborative Ontology Engineering*. Proceedings of the ESWC 2008 Demo Session. Available at: http://www.uni-koblenz.de/~klaasd/Downloads/papers/Dellschaft2008CTD.pdf.

4. K. Dellschaft, A. Gangemi, J. Gómez, H. Lewen, V. Presutti, M. Sini. *NeOn Deliverable D2.3.1 Practical Methods to Support Collaborative Ontology Design*. February 2008. Available at: http://www.neon-project.org/.

5. P. Haase, S. Rudolph, Y. Wang, S. Brockmans, R. Palma, and J. Euzenat, M. d'Aquin. *NeOn Deliverable D1.1.1 Networked Ontology Model*. November 2006. Available at: http://www.neon-project.org/.

6. 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.

7. E. Paslaru-Bontas, C. Tempich, Y. Sure. *ONTOCOM: A Cost Estimation Model for Ontology Engineering*. Proceedings of the 5th International Semantic Web Conference (ISWC 2006). Volume 4273. Lecture Notes in Computer Science (LNCS), pp. 625--639. Springer-Verlag Berlin Heidelberg, 2006.

8. S. Pfleeger. *Software Engineering: Theory and Practice*. 2nd edition. Prentice Hall 2001. ISBN: 0-13-029049-1.

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

10. 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.

11. M. C. Suárez-Figueroa, K. Dellschaft, E. Montiel-Ponsoda, B. Villazon-Terrazas, Z. Yufei, G. Aguado de Cea, A. García, M. Fernández-López, A. Gómez-Pérez, M. Espinoza, M. Sabou. *NeOn D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks*. NeOn project. http://www.neon-project.org. February 2008.

12. M. C. Suárez-Figueroa, A. Gómez-Pérez. *Towards a Glossary of Activities in the Ontology Engineering Field*. 6th Language Resources and Evaluation Conference (LREC 2008). Marrakech (Morocco). May 2008.

13. M. C. Suárez-Figueroa, A. Gómez-Pérez. *First Attempt towards a Standard Glossary of Ontology Engineering Terminology*. 8th International Conference on Terminology and Knowledge Engineering (TKE2008). 18-21 August 2008. Copenhagen.

14. 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.