NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D6.1.2 Report on user requirements v2

Deliverable Co-ordinator: **Carlos Buil Aranda**

Deliverable Co-ordinating Institution: **Intelligent Software Components (iSOCO)**

Other Authors: **Carlos Buil Aranda, Jose Manuel Gómez Pérez (iSOCO), German Herrero Carcel (ATOS), Claudio Baldassarre (FAO), Yimin Wang (UKARL), Oscar Muñoz (UPM), Walter Waterfeld (Software AG)**

.

Document Identifier:	NEON/2008/D6.1.2/v1.0	Date due:	February 29, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 29, 2008
Project start date:	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

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

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:

Partner 1 - iSOCO

Partner 2 - ATOS

Partner 3 - FAO

Partner 4 - UKARL

Partner 4 - UPM

Partner 4 - Software AG

Change Log

Version	Date	Amended by	Changes
0.1	16-11-2007	Carlos Buil Aranda	Initial requirements template
1.0	03-03-2008	Carlos Buil Aranda	Send to QA

Executive Summary

This document is an update of the previous requirement deliverable [13] . This previous document was delivered in month six during which the use cases were in its initial phase and technical work packages started to define the technologies that were going to be implemented within the NeOn toolkit. Currently is month 24 and a release of the NeOn toolkit is planned and some prototypes of the case studies are also due to this month. The first list of requirements is not updated, it has to be reviewed and this is the work done in this deliverable. Here is presented an updated list of requirements, removing the ones that are not needed anymore or were wrong at the time of writing the first deliverable and new requirements have been added due to the new needs of the use cases.

The sources for updating the NeOn requirements have been obtained from the use cases architecture and ontologies deliverables. In these deliverables a complete description of the use cases goals is presented and the real requirements of them are implicitly or explicitly described there. Once these requirements have been gathered a validation process among the technical work packages has been produced in order to create a real requirements list which can be satisfied during the duration of the project.

Accordingly to the IEEE software requirement specification, an overall description of the requirements has been produced first. This description provides a perspective on the NeOn architecture and introduces the functions that shall be satisfied by it. Next, this information is analyzed in detail producing a complete collection of specific requirements.

Table of Contents

List of tables

List of figures

1. Introduction

This document is an update of the previous requirement deliverable [13] . This previous document was delivered in month six during which the use cases were in its initial phase and technical work packages started to define the technologies that were going to be implemented within the NeOn toolkit. Currently in month 24 and a release of the NeOn toolkit is planned and some prototypes of the case studies are also due to this month. The first list of requirements is not up to date, it has to be reviewed and this is the work done in this deliverable. Presented here is an updated list of requirements, removing the ones that are not needed anymore or were wrong at the time of writing the first deliverable and new requirements have been added due to the new needs of the use cases.

The sources for updating the NeOn requirements have been obtained from the use cases architecture and ontologies deliverables. In these deliverables a complete description of the use cases goals is presented and the real requirements of them are implicitly or explicitly described there. Once these requirements have been gathered a validation process among the technical work packages has been produced in order to create a real requirements list which can be satisfied during the duration of the project.

Accordingly to the IEEE software requirement specification, an overall description of the requirements has been produced first. This description provides a perspective on the NeOn architecture and introduces the functions that shall be satisfied by it. Next, this information is analyzed in detail producing a complete collection of specific requirements.

2. Methodology

In this section we describe the methodology followed for updating the previous requirement list presented in [13] .

2.1. Requirements gathering

2.1.1. Feedback from the use cases

For updating the requirements list we asked for the use cases feedback. The end users of the project will be the end users of the use cases and the most important requirements come from them. Also, the technical work packages have also contributed to this deliverable providing feedback to the use cases requirements list.

The update requirement list has been created in two iterations. During the first iteration feedback from the use cases have been requested. This feedback served us to update the old requirement list with the new requirements from the use cases and also we identified the previous requirements that were not needed anymore by them. During the second iteration we distributed the newly update requirement list among the technical work packages. The technical work packages provided feedback about the feasibility of satisfying these requirements and the list was updated.

2.1.2. Analysis of textual resources

Other textual sources have been taken into account in this deliverable the NeOn technical Annex and [1] were the most important sources for creating [13] . Here we included in this requirement list other key deliverables such:

- [12] in which the architecture of the NeOn toolkit is described. We compared the architecture requirements specified in the two textual sources used for creating [13] with the NeOn architecture and we updated the requirements in this deliverable.
- WP1 deliverables: we updated the existing requirements with the WP1 contributions to the metamodel and the ontological functionalities that the NeOn metamodel offers via the NeOn toolkit

2.2. Requirements Sheet

The requirements sheet template is the main tool used to gather a preliminary collection of architecture requirements. It contains six main fields:

- Requirement id.
- Requirement name.
- Requirement description.
- Relevance of the requirement in the overall NeOn architecture. It can be either critical, average, or low.
- Main conceptual architecture layer where the requirement can be allocated. These layers are three: Infrastructure Services layer, Engineering Components, and GUI Components.
- Type of the requirement extracted, either functional (FR) or non functional (NFR).
- Traceability information. This can be either internal or external. Internal traceability relates a given requirement with other requirements it depends on, whereas external traceability collects information regarding the exact information source where the requirement is detected.

A preliminary version of the template was filled in with information from the textual sources, then it was circulated among partners and the collected data was merged.

This sheet allows classifying requirements hierarchically, from the more general to the more concrete, providing a stratified vision of the requirement information. However, the sheer amount of data collected is too large and the requirements sheet alone does not suffice in order to clearly display information contained within. In this direction, requirement categories have been created, following the IEEE methodology for software requirements specification, pursuing a more precise requirements description. The complete description of the IEEE methodology used for gathering the requirements can be found in [13] .

2.3. Requirements lifecycle

Software requirements specification is not a sequential process with clear and well-defined beginning and end. Quite on the contrary, this is a spiral model where requirements themselves have their own lifecycle. Requirements are:

1. Detected,
2. Gathered and contextualized,
3. Decomposed into smaller requirements,
4. Evolved along with the software product,
5. Sometimes, no longer applicable and disappear from the system specification.

According to this life cycle, partners produced feedback on the different versions of the architecture requirements in order to obtain a more complete and comprehensive list. Specific contributions from the case studies are crucial. They help establishing actual relevance of architectural requirements in terms of their usefulness for application development. In general, user interaction is critical for requirements gathering and evolution as they provide first-hand insight.

This kind of iterations on the requirements list show how they are updated with new recommendations and how new requirements appear. The relevance of the requirements identified will be naturally fitted as NeOn evolves.

This document covers all the lifecycle of the requirements. The requirements lifecycle started in month 6 of the NeOn project and now, in month 24 a revision of these requirements is produced. In this revision the steps 1 to 5 are executed in order to build this final document.

3. NeOn architecture requirements

Next, the NeOn architecture requirements are classified according to the IEEE software requirements specification methodology [3] . Please note that the document structure below may not exactly coincide with the one proposed by this methodology. Quite on the contrary, it has been adjusted to better fit our aim to describe the NeOn architecture requirements, object of this document.

3.1. Overall Description [1]

An overall description of the NeOn architecture requirements is first introduced. Then, specific, detailed requirements will be provided.

3.1.1. NeOn architecture perspective

This section puts the NeOn architecture into perspective with respect to the expected use of networked ontologies. System requirements are sketched within and all kind of interfaces describing the relation of the architecture with users, hardware, software, and communication are introduced.

3.1.1.1. System requirements

Table 1 shows the complete set of system requirements for the NeOn architecture. This table describes the main assumptions upon which the NeOn architecture shall be built. The requirements contained herein, each displayed with their own identifier, are divided in three main categories, as shown in Figure 1: NeOn architecture layers, NeOn services, and NeOn compatibility and extension.

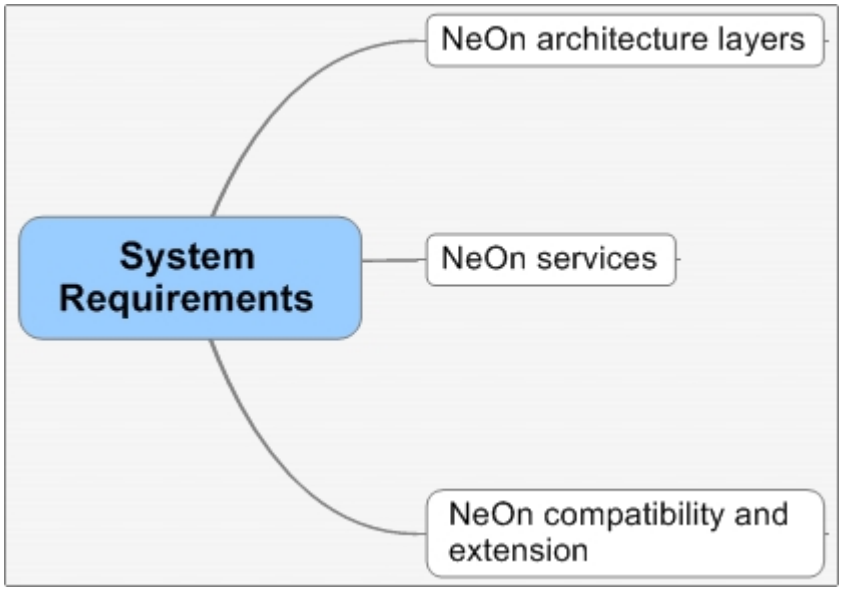

Figure 1: System requirements

NeOn architecture layers

The NeOn architecture is defined in [12] . This document specifies the architecture that the NeOn toolkit has and how the differerent services and plugins are integrated into the platform. The requirements for this architecture were described in [13] and in this section we see how these requirements are satisfied by the NeOn toolkit.

NeOn services

This section describes the list of services provided by NeOn.

[1] This section corresponds to section 2 of the IEEE software requirements specification.

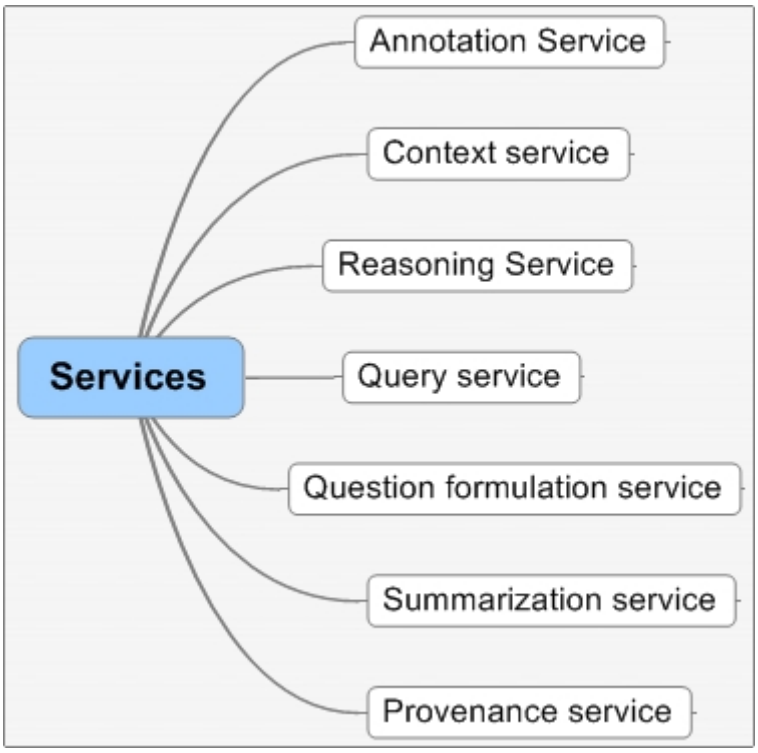

Figure 2: NeOn services

NeOn compatibility and extension

Ontologies and data shall be loaded in a stand-alone server which implements the backend for the reasoning service and the ontology editor. On the other hand, given that both OWL-like languages and Frame languages coexist in the Semantic Web stack the NeOn data model will observe the dual language approach, i.e. both language types will be supported.

Req #	Title	Description	Importance	External traceability
2.1.1.1	Infrastructure services	Information resources will be potentially stored in an Infrastructure services managed by this architecture layer. NeOn shall access these resources transparently to their location	Critical	NeOn TA[2], D6.2.1
2.1.1.2	Engineering components	This middleware layer between the GUI Components and the Infrastructure services shall implement a number of services which allow users to exploit the information stored in the Infrastructure services	Critical	NeOn TA, D6.2.1
2.1.1.3	GUI components	User front ends and high level services shall be supported by this layer	Critical	NeOn TA, D6.2.1
2.1.1.4	Inter-layer communication mechanism	The NeOn architecture layers shall be at the same time service functionality clients and producers for each other. Natural flow of architecture service invocation goes from the higher level layers to the lower level layer. On the other hand, push services, e.g. information refresh may be triggered upwards by lower level layers, in this case, the Infrastructure services layer. The necessary infrastructure, e.g. a service bus, shall be implemented to allow this kind of service invocation	Critical	NeOn TA, D6.2.1
2.1.1.5	Support for integration of heterogeneous information	The following types of distributed information resources shall be integrated in the NeOn knowledge model: - Unstructured content, e.g. text. - Structured knowledge with no clear semantics, e.g.	Critical	D8.3.1, D8.2.1, D7.4.1 (several UC)

[2] Technical Annex

	sources	DBs, catalogues. - Formal knowledge (ontologies and data) - Semantic annotations		
2.1.1.6	Annotation service	The second layer of the NeOn architecture shall include annotation mechanisms and associated services	Critical	D8.3.1, D8.2.1, D7.1.2, D7.4.1 (several UC)
2.1.1.7	Summarization service	The second layer of the NeOn architecture shall include an informational service briefing ontology information	Critical	NeOn R&V
2.1.1.8	Context service	The second layer of the NeOn architecture shall include a service to gather and process context information for later exploitation	Critical	D8.3.1, D8.2.1 (Nomenclator req)
2.1.1.9	Reasoning service	The second layer of the NeOn architecture shall include reasoning mechanisms and associated services.	Critical	D8.3.1, D8.2.1
2.1.1.10	Query service	The second layer of the NeOn architecture shall include query mechanisms and associated services.	Critical	D6.2.1, provided by the plugins
2.1.1.11	Question formulation Service	The NeOn toolkit shall provide a question formulation service that eases query formulation by allowing more natural ways of expressing queries, e.g. natural language	Average	NeOn R&V
2.1.1.12	Provenance service	The NeOn middleware layer shall include a service which keeps track of the precedence of ontologies, i.e. how, when, and by whom ontologies evolve across their lifecycle.	Critical	D8.2.1 (nomenclatore req), 3.4.3 in D7.1.2
2.1.1.13	Re-use and extension of pre-existing technology	The architecture shall be open enough and facilitate the inclusion and extension of state-of-the-art ontology platforms	Average	NeOn TA, D6.2.1
2.1.1.14	Stand-alone server	Ontologies and data shall be loaded in a stand-alone server, based on OntoStudio and OntoBroker, which implements the backend for the reasoning service and the ontology editor.	Critical	D6.2.1 architecture specification
2.1.1.15	Dual language approach	FLogic and OWL shall be supported	Critical	D6.2.1 architecture specification
2.1.1.16	Mappings service	The second layer of the NeOn architecture shall include mapping mechanisms and associated services.	Critical	D6.2.1 architecture specification
2.1.1.17	Integrated NeOn Infrastructure	The NeOn architecture shall integrate the technology developed in the technical work packages	Average	NeOn TA
2.1.1.18	Preconfigured and bundled infrastructure	NeOn infrastructure shall be preconfigured and bundled into sector-specific solutions, e.g. in the case studies	Average	NeOn TA, NeOn case studies
2.1.1.19	Loosely coupled architecture	NeOn architecture will be mainly based on a loosely couple concept	Critical	D6.2.1, architecture specification
2.1.1.19.1	Loosely coupled architecture	NeOn architecture shall allow loose coupling of components	Critical	D6.2.1, architecture specification
2.1.1.19.2	Loosely coupled architecture	NeOn architecture shall allow loose coupling of resources	Critical	D6.2.1, architecture specification

Table 1: System requirements

3.1.1.2. User interface

Table 4 shows the logical characteristics of interfaces between NeOn and its potential users, including GUI configuration issues and customization. Figure 3 shows a classification of user interface requirements with three large categories: those associated to the NeOn editor, requirements for the NeOn browser, and requirements bound to the NeOn GUI. The requirements described herein represent the counterparts of the system functionalities described in section 3.1.2 in terms of GUI capabilities.

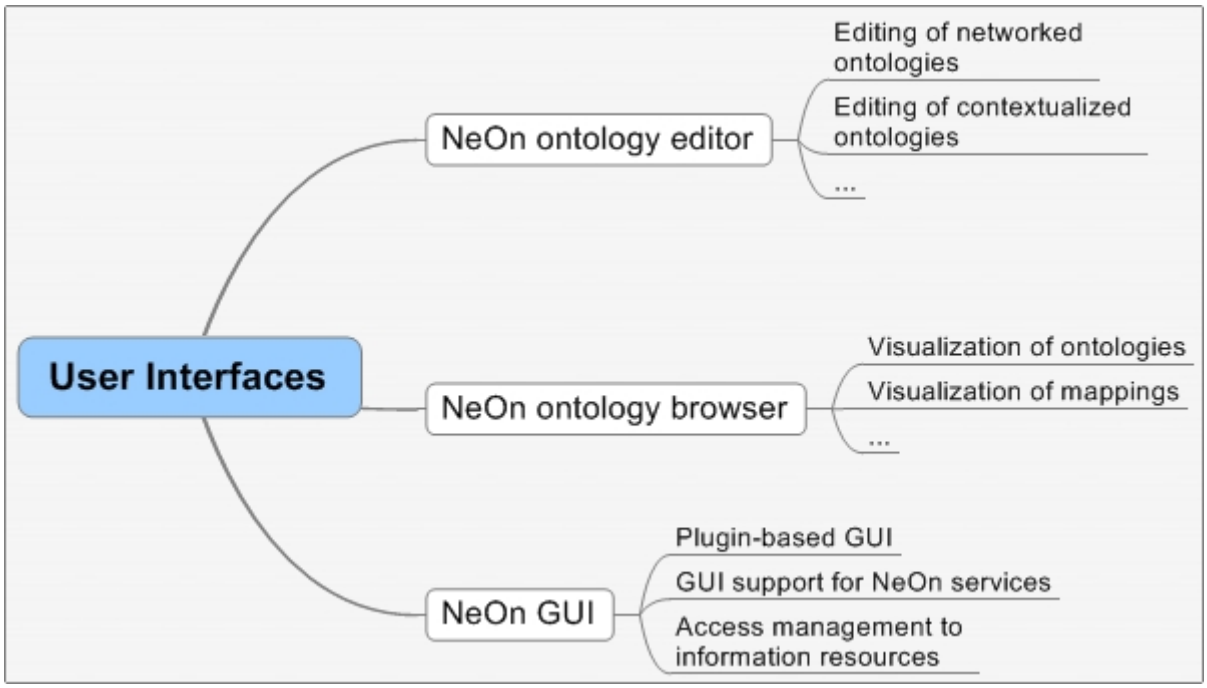

Figure 3: User interfaces requirements

NeOn Ontology Editor

These requirements appeal to the NeOn functionalities related to the editing of networked ontologies in the NeOn framework. The NeOn Editor, based on OntoStudio, which will be appropriately extended for the networked case, shall allow working on several networked ontologies simultaneously. The NeOn Editor shall allow selection and reuse of complete ontologies as well as of their parts. This can be achieved by making the NeOn Editor aware of the modules comprised by a particular ontology, including ontology module editing capabilities. The NeOn Ontology Editor shall allow editing all kind of ontology entities, including ontology mappings.

A fundamental feature, shed by the NeOn pharmaceutical [8] and fishery [7] case studies, is support for contextualized ontologies. The ontologies needed in these scenarios are contextualized, i.e. their properties change according to the particular context of application, and only a subset of these properties is valid in a certain context. Thus, the NeOn Editor shall also allow users to define, select and edit ontology context.

Req #	Title	Description	Importance	External traceability	Internal traceability
2.1.2.1	NeOn Editor	The NeOn toolkit layer shall provide an ontology editor based on OntoStudio	Critical	D8.3.1, D8.2.1, D7.1.2, UC7 in D7.4.1	2.1.1.3 3.2.8
2.1.2.1.1	Edition of ontologies	Change textual attributes of elements of the Ontology	Critical	UC7 in D7.4.1, D8.2.1, new UC in WP7	2.1.1.3 3.2.8
2.1.2.1.2	Edition of multiple ontologies	Run simultaneous editing jobs for multiple Ontologies	Average	4.4.1 in D7.1.1, 4.4.7 in D7.1.1	2.1.1.3 3.2.8

				UC3 in D7.4.1, D8.2.1, D8.3.1, new UC in WP7	
2.1.2.1.3	Multilinguality edition	Add lexicalization in a new language (localization)	Critical	4.4.7 in D7.1.1 UC3 in D7.4.1, 3.4.1 in D7.1.2	2.1.1.3, 3.2.5
2.1.2.1.4	Support for contextualized ontologies	NeOn ontologies shall be contextualized, i.e. their properties depend on the current context	Critical	D8.2.1, D8.3.1 (semantic Nomenclatore req)	2.1.1.3 2.1.1.9
2.1.2.1.4.1	Ontology context editing	The NeOn editor shall allow to select and edit ontology context	Critical	D8.2.1, D8.3.1 (semantic Nomenclatore req)	3.2.7
2.1.2.1.5	Partial ontology selection	Partial selection and reuse of ontologies shall be available	Average	D8.3.1, D8.2.1, UC7 in D7.4.1	2.1.1.3 3.2.8.1.3
2.1.2.1.6	Simplified Ontology Editor	Provide to the users an ontology editor with the most basic operations	Critical	4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3 3.2.8

Table 2: NeOn Ontology Editor Requirements

NeOn Ontology Browser

NeOn shall allow visualizing the structure of networked ontologies, including ontology modularization, and the properties of the ontological entities contained within. Additionally, NeOn shall provide means to graphically represent changes in networked and shared ontologies, as well as ontology lifecycle provenance information, i.e. who and how interacted with a given ontology or with a set of them.

NeOn shall also provide mapping visualization. In this case, effective graphical representation of mapping information is expected to be especially helpful since NeOn will manage ontologies in a potentially complex networked environment. In this direction, NeOn shall allow visualizing all kind of dependencies between networked ontologies.

It is also certainly relevant that context information be accurately displayed. In this direction, NeOn shall provide contextualized views of ontologies. All the different types of visualization described in Table 4 shall be parameterized, and these parameters customizable.

With high probability, the NeOn architecture design will eventually integrate browsing capabilities with the NeOn Ontology Editor.

Req #	Title	Description	Importance	External traceability	Internal traceability
2.1.2.2	NeOn Browser	NeOn shall allow ontology browsing. Might be integrated with the NeOn editor	Critical	D8.3.1, D8.2.1, D6.2.1, D7.4.1, D71.1	2.1.1.3
2.1.2.2.1	Networked ontologies Visualization	NeOn shall allow to browse and visualize networks of ontologies	Critical	D6.2.1, D7.4.1, D71.1 4.4.4 in D7.1.1 UC7 in D7.4.1, D8.2.1	2.1.1.3

2.1.2.2.1.1	Networked ontologies Visualization	Visualization of the Ontology is customizable according the editorial activity; diagram-like, indented tree, node by node are required visualization types	Average	4.4.4 in D7.1.1 2.4.6.22 in D7.1.2	2.1.1.3, 3.2.5
2.1.2.2.1.2	Networked ontologies Visualization	Select one or more languages to visualize lexicalization of Ontology elements	Critical	4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3, 3.2.5
2.1.2.2.1.3	Networked ontologies Visualization	Instantiate a graphic environment for each ontology opened	Medium	4.4.4 in D7.1.1 UC7 in D7.4.1 2.4.6.23 in D7.1.2	2.1.1.3
2.1.2.2.1.4	Networked ontologies Visualization	Support visualization for ontology population from text	Critical	4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3
2.1.2.2.1.5	Networked ontologies Visualization	Print visualization	Average	4.4.4 in D7.1.1 UC7 in D7.4.1 2.4.6.28 in D7.1.2	2.1.1.3
2.1.2.2.1.6	Advanced Ontology Visualization	Hide view of a class	Average	3.4.2 in D7.1.2	2.1.1.3
2.1.2.2.1.7	Advanced Ontology Visualization	Hide relation	Average	3.4.2 in D7.1.2	2.1.1.3
2.1.2.2.1.8	Advanced Ontology Visualization	Hide branch	Average	3.4.2 in D7.1.2	2.1.1.3
2.1.2.2.2.1	Individual Visualization	View related individual data	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.2	Individual Visualization	View related individual data text	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.3	Individual Visualization	View related individual data text highlighted	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.4	Individual Visualization	View multiple related individual data	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.5	Individual Visualization	View trends across different sets of related data (fox example compare sets of fish data building representative curves for each set)	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.6	Individual Visualization	View link to related data	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.7	Individual Visualization	View related data extraction	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.8	Individual Visualization	View related data metadata	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.9	Individual Visualization	View related data quality	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.10	Individual Visualization	View related data provenance	Critical	3.4.3 in	2.1.1.3

				D7.1.2	
2.1.2.2.2.11	Individual Visualization	View related data modification	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.12	Individual Visualization	View related data creation	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.13	Individual Visualization	Group related data by concept and relation	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2.14	Individual Visualization	Re-group related data by concept and relation	Critical	3.4.3 in D7.1.2	2.1.1.3
2.1.2.2.2	Context-dependent Ontology visualization	NeOn shall provide contextualized views of ontologies	Critical	D8.2.1 (Semantic Nomenclator req)	3.2.7
2.1.2.2.3	Visualization of changes in networked ontologies	Visualize Ontology information inherent to editorial work: change log for an element, participating authors, frequency of changes	Critical	D8.3.1 D8.2.1 4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3 3.2.8.1.1.1
2.1.2.2.4	Cross-author change tracking	NeOn shall provide means to represent ontology lifecycle provenance information, who and how interacted with a given ontology	Average	4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3, 2.1.1.12
2.1.2.2.5	Mappings visualization	Visualize mappings list for Ontology pair	Critical	4.4.4 in D7.1.1 UC7 in D7.4.1, D8.2.1, D8.3.1	2.1.1.3 3.2.6
2.1.2.2.5.1	Mappings visualization	Visualize overlapping Ontology elements for Ontology pair	Critical	4.4.4 in D7.1.1 UC7 in D7.4.1	2.1.1.3 3.2.6
2.1.2.2.6	Briefing on ontology entities	Information about ontology entities shall be displayed in a comprehensible way	Average	NeOn R&V NeOn case studies	2.1.1.3 3.2.11
2.1.2.2.7	Visualization of dependencies between networked ontologies	NeOn shall allow to visualize dependencies, i.e. connections and alignments between networked ontologies	Average	4.4.4 in D7.1.1 UC7 in D7.4.1, D8.2.1	2.1.1.3
2.1.2.2.8	Visualization of Ontology modules	NeOn shall provide visualization of ontology modules	Average	D8.3.1, D7.4.1, D7.1.1	2.1.1.3
2.1.2.2.9	Ontology summarization	NeOn shall display information about properties of ontologies in a summarized way.	Average	D6.2.1	2.1.1.3 2.1.1.8
2.1.2.2.9.1	Ontology summarization	Summarization shall contain the main characteristics of the ontologies.	Average	D6.2.1	3.2.15 2.1.1.7
2.1.2.2.9.2	Customizable summarization	NeOn shall allow to customize the parameters to be summarized	Average	D6.2.1	3.2.15 2.1.1.7
2.1.2.2.9.3	Customizable summarization	Visualize Ontology information inherent to editorial work: change log for an element, participating authors, frequency of changes	Average	4.4.4 in D7.1.1 2.4.6.25 in	3.2.15 2.1.1.7

				D7.1.2 D8.2.1 (semantic Nomenclator)	
2.1.2.2.10	Briefing on non-ontological resources	NeOn shall provide information about non-ontological resources being used	Average	D6.2.1	2.1.1.3 3.2.2.2
2.1.2.2.11	Ontology Search	Find an ontology	Critical	3.4.1 in D7.1.2, D8.2.1, D8.3.1	
2.1.2.2.11.1	Ontology Search	Search a loaded Ontology	Critical	UC1 in D7.4.1 (extension)	2.1.1.3
2.1.2.2.11.2	Ontology Search	Search for Ontology elements based on semantic constraints	Critical	UC1 in D7.4.1 (extension)	2.1.1.3
2.1.2.2.11.3	Ontology Search	Search for Ontology elements using string comparison criteria	Critical	UC1 in D7.4.1 (extension)	2.1.1.3
2.1.2.2.12	Workflow visualization	Visualization of Ontology elements masked by Work Flow status	Critical	UC7 in D7.4.1	2.1.1.3

Table 3: Ontology Browser Requirements

NeOn GUI

NeOn GUI requirements specify the rest of the properties that the NeOn user interface must comply with, in terms of facilities for user interface configuration and user access to services. These requirements can be classified under the following main categories:

- Plugin-based GUI
- Support for different types of clients
- GUI support for NeOn services
- Access management to information resources

As stated in the NeOn compatibility and extension requirements, in particular, requirement 2.1.1.19, NeOn extensibility and flexibility is a really important point to be considered under the architecture perspective. This issue is also reflected as part of the user interface requirements. Thus, NeOn shall allow easy configuration of the functionalities available in the NeOn toolkit by means of a plugin-based system. NeOn shall also provide users with reflective functionalities offering information about the plugins present in the interface at any time.
This plugin-based system will be part of the infrastructure that supports the implementation of rich clients to the NeOn functionalities. A rich client contains a large amount of logic within. On the other hand, thin clients, i.e. clients with hardly any logic within, will be based on web technology, merely a dumb interface that collects user requests, invokes the appropriate services, and shows results after execution. The main advantage of this kind of clients is that they can be decoupled from the component actually providing the functionality. Hence, they turn to be certainly useful in distributed systems like NeOn.
Part of the requirements on user interfaces correspond to front-ends of the NeOn services described in 3.1.1.1, i.e. NeOn shall provide GUI support to every service. Additionally, though this requirement is weaker than those described so far NeOn shall provide a user interface that enabled certain types of users with an administrator role

to manage access rights to information resources, including ontological and non-ontological knowledge. On the other hand, information about resource accessibility shall be displayed to regular users.

Req #	Title	Description	Importance	External traceability	Internal traceability
2.1.2.3	NeOn GUI	NeOn shall offer a user-friendly GUI to functionalities and services	Critical	D8.3.1, D8.2.1, D7.4.1, D71.1	2.1.1.3
2.1.2.3.1	Customisation facilities	Visualization of the Ontology is customizable according the editorial activity; diagram-like, indented tree, node by node are required visualization types	Average	4.4.4 in D7.1.1 2.4.6.22 in D7.1.2	2.1.1.3, 2.1.1.9, 3.2.8
2.1.2.3.2	Accessibility to plugins	Plugin load features shall be accessible to the user	Average	D6.2.1	3.1.1
2.1.2.3.2.1	Plugin acknowledgement	NeOn shall provide means to inform users of the different plugins installed.	Average	D6.2.1	3.1.1
2.1.2.3.2.2	Plugin acknowledgement	The NeOn GUI should report which services (plugins) are available and which are enabled.	Critical	D8.2.1	3.1.1
2.1.2.3.3	GUI support for NeOn services	NeOn shall provide user interfaces to every NeOn service	Critical	NeOn R&V	2.1.1.3
2.1.2.3.3.1	GUI support for the annotation service	NeOn shall provide a user interface to the annotation service	Average	D8.3.1, 3.4.1 in D7.1.2	2.1.1.6
2.1.2.3.3.3	GUI support for reasoning and inference	NeOn shall provide a user interface to the reasoning and inference service	Average	D8.3.1, D8.2.1	2.1.1.9
2.1.2.3.3.3.1	GUI support for query formulation	NeOn shall provide a user interface to formulate queries in standard query languages	Average	NeOn R&V	2.1.1.10
2.1.2.3.3.3.2	GUI support for question formulation	NeOn shall provide a user interface to formulate questions	Average	NeOn R&V	2.1.1.11
2.1.2.3.3.3.3	GUI support for answer explanation	NeOn shall sensibly explain question answers in the terms of the formulated question	Average	NeOn R&V	2.1.1.11
2.1.2.3.3.4	GUI support for the summarization service	NeOn shall provide a user interface to the summarization service	Average	NeOn R&V	3.2.15 2.1.1.7
2.1.2.3.3.5	GUI support for the context service	NeOn shall provide a user interface to the context service	Average	D8.3.1, D8.2.1 (Semantic Nomenclator)	2.1.1.8
2.1.2.3.3.6	GUI support for the provenance service	NeOn shall provide a user interface to the provenance service	Average	D8.2.1, D8.3.1 (Semantic Nomenclator), 3.4.3 in D7.1.2	2.1.1.12 3.2.16
2.1.2.3.3.7	GUI support for the NeOn services	GUI support for the NeOn services (all services should have a similar GUI template)	Critical	D8.2.1 (Semantic Nomenclator req)	2.1.1.3
2.1.2.3.4	Access management to information resources	NeOn shall provide a GUI to show and manage access rights to resources	Average	D8.2.1, D8.3.1, D7.1.1	2.1.1.3 3.6.3.1
2.1.2.3.4.1	Access management to ontological resources	NeOn shall provide a GUI to show and manage access permission to ontological resources	Average	D8.3.1, D8.2.1, D7.4.1, D7.1.1	2.1.1.3 3.2.2.14

2.1.2.3.4.2	Access management to non-ontological resources	NeOn shall provide a GUI to show and manage access permission to non-ontological resources	Average	D8.3.1, D8.2.1, D7.4.1, D7.1.1	2.1.1.3 3.2.2.14
2.1.2.3.5	Rich client	NeOn shall provide support for rich clients, i.e. NeOn clients with a large amount of service logic within	Critical	D8.3.1, D8.2.1, D7.4.1, D7.1.1	3.2.19
2.1.2.3.6	Thin client	NeOn shall provide support for thin clients, i.e. NeOn clients with hardly any logic within, like web clients.	Average	D8.3.1, D8.2.1, D7.4.1, D7.1.1	3.2.19

Table 4: NeOn GUI requirements

3.1.1.3. Hardware requirements

This section specifies the logical characteristics of interfaces between the NeOn architecture and the hardware components of the system, including configuration characteristics. It also covers such matters as e.g. what devices are to be supported, how they are to be supported, and protocols. Hardware requirements gathered so far have been structured in Table 5. This table will definitely change as the project evolves. Currently, the minimum processor and memory specifications are provided.

Req #	Title	Description	Importance	External traceability
2.1.3.1	Minimum processor	Pentium IV 2 GHz or better	Critical	NeOn TA
2.1.3.2	Minimum memory	1GB RAM	Critical	NeOn TA

Table 5: Hardware requirements

3.1.1.4. Software requirements

The use of additional software products, e.g. data management systems or operating systems, and interfaces with other applications, e.g. linkage with software libraries, are specified here. As Table 6 shows, NeOn shall be compatible with the main operating systems in the market, including the popular Microsoft family of windows systems and the different distributions of the open source alternative operating system, Linux. Additionally, Java will be the main technology underlying the implementation of NeOn.

Req #	Title	Description	Importance	External traceability
2.1.4.1	Compatibility with several platforms	The NeOn toolkit shall be compatible with several platforms	Critical	NeOn TA
2.1.4.1.1	Operating Systems	NeOn infrastructure shall be compatible with several Operating Systems	Critical	NeOn TA
2.1.4.1.1.1	Windows XP/2000 family	NeOn infrastructure shall be compatible with the Windows XP/2000 family	Average	NeOn TA
2.1.4.1.1.2	Linux family	NeOn infrastructure shall be compatible with the Linux family	Average	NeOn TA
2.1.4.1.1.3	Mac OS X	NeOn infrastructure shall be compatible with Mac OS X	Average	NeOn TA

2.1.4.1.2	Java Platform	NeOn infrastructure shall run on all Java 1.4.2-compatible JVMs	Average	NeOn TA
2.1.4.1.2.1	Java Platform	Deploys using JWS	Medium	3.7 in D7.1.2

Table 6: Software requirements

3.1.1.5. Communication interfaces

Table 7 shows two meta-requirements for communication in the NeOn infrastructure. Communication interfaces are seen under the perspective of the different components which form part of the NeOn platform. Hence, it is certainly important that NeOn shall offer a service-oriented interface that allows legacy components to be included with the smallest impact on the rest of the system. This way, the standard interface methods described as part of the NeOn compatibility and extension policies in section 3.1.1.1 shall be enforced. However, components built from scratch under the NeOn specification shall rather use more direct, higher performance methods for integration. In this direction, the NeOn architecture shall offer an API accessible by means of several client interfaces.

Req #	Title	Description	Importance	External traceability	Internal traceability
2.1.5.1	Service oriented architecture	NeOn shall offer a service oriented interface in order to ease the inclusion of new components into any of the three architecture layers	Critical	D8.2.1	2.1.1.19
2.1.5.2	API –based interface	The NeOn backend will be accessible by means of several client interfaces, e.g. Java, .NET.	Critical	D8.2.1	2.1.1.5

Table 7: Communication interfaces

3.1.2. Functions of the NeOn architecture

This section contains a list of the major functions that the NeOn architecture shall perform, to be detailed in section 3.2.2. Next, we summarize these functions, organized following a classification that groups them in self-contained categories.

Geographically-transparent access to Infrastructure services: NeOn shall allow access to ontological and non ontological information pieces transparently from their location.

Support of heterogeneous knowledge: Different types of information sources will be managed by NeOn, which shall provide the necessary means to include them all into its knowledge model.

Dual language approach: NeOn shall be compliant with the current and future Semantic Web standards. The languages supported are F-Logic and OWL.

Ontology modularization: Large ontologies, where information tends to be dispersed, shall be properly managed by means of a modular structure.

Multilinguality support: As pointed out by the NeOn pharmaceutical and fishery case studies, support for multilingual ontologies shall be provided.

Mapping support: NeOn shall provide means to simplify alignment between entities of two or more globally inconsistent ontologies.

Support for contextualized ontologies: NeOn shall provide the means to parameterize ontologies with respect to their different contexts.

Lifecycle support for ontology development: NeOn shall support knowledge acquisition and ontology development and maintenance throughout their lifecycle, extending standard ontology development facilities to the networked arena.

Reasoning and inference: The NeOn backend shall provide reasoning capabilities on top of the aggregated knowledge contained in a network of ontologies.

Query support: Standard ontology query languages like e.g. SPARQL shall be supported by NeOn, allowing the retrieval of information contained within.

Question formulation: Question formulation facilities shall be built on top of ontology querying that will allow users to build more complex and expressive questions for information retrieval.

Semantic annotation: NeOn shall incorporate semantic annotation functionalities which will provide information characterization in terms of the model described in a set of networked ontologies.

Access management to information resources: Access rights to information resources managed by NeOn need to be administrated as shown by the case studies

User profiling: NeOn shall accumulate information about user profiles during system interaction and provide a customized set of functionalities according to the inferred profile.

Ontology summarization: NeOn shall provide users with the means to summarize ontology state and properties.

Provenance support: Ontology traceability shall be kept by automatically storing relevant information like authors or contributors.

System documentation and help: The different user types of NeOn shall be provided with extensive help and training materials, including tutorials.

Support for different client types: Rich and thin clients shall be supported by the NeOn architecture, as described in section 3.1.1.2.

Service access to the NeOn backend: Access to the NeOn backend shall be provided to NeOn clients.

Plugin support: The NeOn toolkit shall make available its components in a plugin mode. Users shall be able to develop their own plugins and add them to the NeOn toolkit architecture.

3.1.3. Users characteristics

We have focused the contents of this section under the perspective of the properties that NeOn shall satisfy in order to allow ontology developers to easily and quickly get acquainted with the system. First of all, each functionality provided by the system shall be accessible, i.e. the effort required by the user in order to reach them shall be kept certainly low. This requirement is mainly focused on the NeOn toolkit layer of the architecture but shall also be supported by the other two layers.

We plan to achieve this goal by following the conclusions of the user study carried out in [5] where a complete and deep analysis on current ontology engineering tools was made. This user study provided as a result a set of guidelines for the development of successful, human-ontology interaction compliant tools.

On the other hand, NeOn shall be delivered to the end user as a self-contained software package ready for installation. We foresee to provide two different types of installation packages, one containing the open source version of NeOn and a different one for the full fledged commercial version of the software. Both shall be automatically deployable, and an installation wizard will be provided along.

3.1.4. Constraints

Herein we describe the factors that shall constrain the development process of the NeOn infrastructure, including quality assurance criteria. Table 8 contains a detailed collection of these requirements.

Development constraints can be classified under four main categories, also graphically shown in Figure 4:

- **Cost estimation:** Different methods shall be used that allow evaluating the cost of each component of the NeOn infrastructure in a preliminary stage, after implementation design.
- **Testing:** Unit and functional tests shall be used to check the NeOn software.

- **Bug tracking and fixing:** Continuous bug tracking and fixing shall be carried out during the lifecycle of NeOn. The necessary infrastructure shall be provided in order to support this effort.
- **Software licensing:** NeOn or at least part of it shall be licensed as open source software, with the corresponding license, like e.g. (L)GPL.

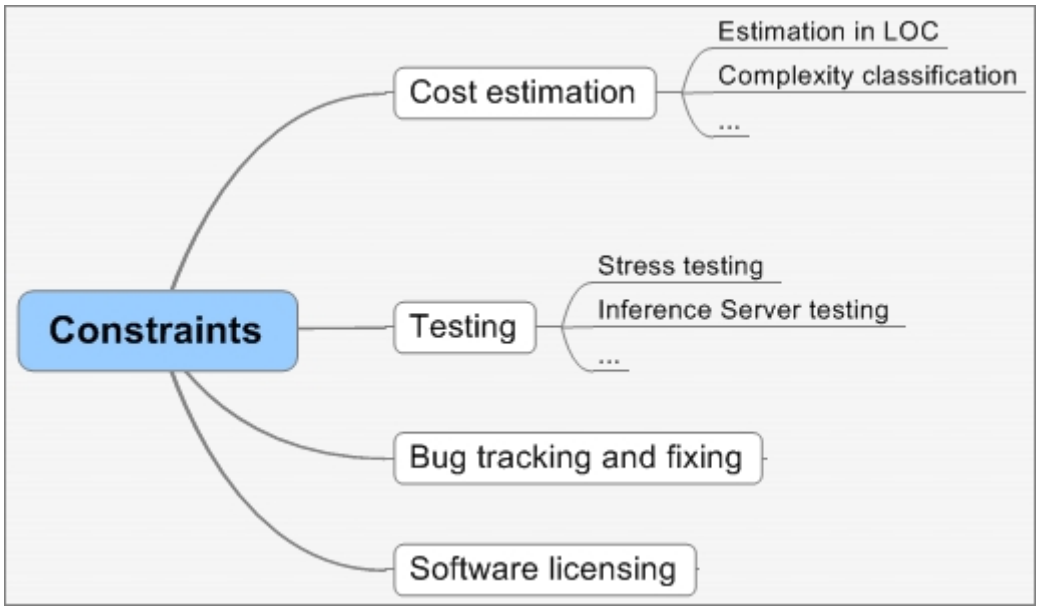

Figure 4: Constraints classification

The methods considered for cost estimation include classification of each component implementation complexity. Values ranging from 1 to 4 will be assigned that quantify this complexity. Additionally, as soon as preliminary prototypes are ready, estimations of the software complexity in terms of lines of codes (LOC) shall be made. Finally, when software maturity allows, a post-mortem analysis will be performed that evaluate the accuracy of predictions by comparing estimated cost with actual cost after implementation.

Tests will be carried out aiming to evaluate two main issues: i) system performance and ii) usability. Performance tests will be focused in measuring the throughput of the reasoning and inference engine of the NeOn backend in different situations. In this direction, a benchmark suite shall be built and executed against large, real-world ontological and non-ontological knowledge repositories. Connectivity shall also be evaluated in order to determine quality of service in terms of throughput and latency with respect to the number of components as well as external clients supported by the architecture.

Usability testing shall eventually reproduce the user study conducted in [5] to evaluate NeOn in terms of human-ontology interaction. The questionnaire created therein will be the perfect tool in order to compare NeOn with the ontology engineering tools evaluated already. Nevertheless, this kind of user study needs a certain degree of stability. Until then, other tests can be performed in order to ensure usability satisfaction. These measures include, but are not limited to, cognitive walkthroughs and interviews with representative user groups throughout the different stages of the NeOn lifecycle, periodic user observation in order to assess GUI intuitiveness and acceptance, and questionnaires that shall help to collect early feedback from test users.

Bug tracking infrastructure shall be available both for the members of the NeOn consortium and end users though different interfaces, focused on the particular needs of each groups of users, will be provided. Apart from the infrastructure, it is also necessary to establish policies for bug fixing. Thus, resources shall be allocated for swift bug fixing.

Finally, open source versions of NeOn shall be licensed in order to achieve maximal dissemination of the framework among the community that shall coexist with commercial versions, as introduced in section 3.2.3. Sites like SourceForge shall be used to distribute open source versions of NeOn.

Req #	Title	Description	Importance	External traceability
2.4.1	Development process	NeOn components shall be developed following a predefined design and implementation method	Critical	NeOn TA

2.4.1.1	Implementation design validation	The implementation design shall be checked against the functional specification of the component	Critical	NeOn TA
2.4.1.2	Implementation changes	Implementation design of the components shall be modified according to eventual changes	Critical	NeOn TA
2.4.1.3	Cost estimation	The cost of each component shall be estimated after implementation design	Critical	NeOn TA
2.4.1.3.1	Estimation in LOC	Implementation cost shall be measured in terms of lines of code	Critical	NeOn TA
2.4.1.3.2	Predicted cost versus real cost analysis	The estimated cost shall be compared against the actual cost after implementation	Critical	NeOn TA
2.4.1.3.2.1	Deviation analysis	Deviation analyses shall be used	Critical	NeOn TA
2.4.1.4	Testing	Unit and functional tests shall be used to check the software	Critical	NeOn TA
2.4.1.4.1	Stress testing	Stress testing using large, real-world ontologies	Critical	NeOn TA
2.4.1.4.2	Inference server testing	A benchmark suite shall be used to guarantee performance of the inference server	Critical	NeOn TA
2.4.1.4.2.1	Server scalability tests	How many concurrent clients the server can handle	Critical	NeOn TA
2.4.1.4.2.2	Client/server I/O performance test	Throughput and latency	Critical	NeOn TA
2.4.1.4.2.3	Reasoning and inference kernel performance testing	The inference kernel shall be intensively tested with large, real-world ontologies	Critical	NeOn TA
2.4.1.4.3	Large-scale, real-world integration scenarios	Industrial partners shall use NeOn technology in their commercial products, e.g. Software AG shall apply it to the EII product using real-world integration scenarios from several customers	Critical	NeOn TA
2.4.1.4.3.1	Transparent use of non-ontological resources	Transparent use of non-ontological resources shall be evaluated	Critical	NeOn TA
2.4.1.4.3.2	Runtime access to external data sources	Runtime access to external data sources via inference on integration ontologies	Critical	NeOn TA
2.4.1.4.4	Usability and user-centred tests	Usability and user-centred tests shall be executed	Critical	NeOn TA
2.4.1.4.4.1	Cognitive walkthroughs and structured interviews	Cognitive walkthroughs and structured interviews with small, focused, representative user groups throughout the different stages of NeOn toolkit/methodology design and development	Critical	NeOn TA
2.4.1.4.4.2	User observation	User observation shall asses intuitiveness and acceptance of user interface features	Critical	D8.2.1

2.4.1.4.4.3	Questionnaires	Users shall fill in questionnaires in order to evaluate their perception on NeOn	Critical	NeOn TA
2.4.1.4.4.4	Involvement of NeOn toolkit	Involvement of NeOn toolkit, methodology and techniques used by ontology developers in acquiring and analyzing user-centred test; also performing usability studies on the developers themselves	Critical	NeOn TA
2.4.1.4.4.5	Involvement of cross-disciplinary teams	Involvement of cross-disciplinary teams (incl. Experts on HCI, psychology of programming, practicing developers, accessibility for impaired users, etc.)	Critical	NeOn TA
2.4.1.5	Infrastructure for continuous bug tracking and fixing	Infrastructure for continuous bug tracking and bug fixing management	Critical	NeOn TA
2.4.1.5.1	Infrastructure for continuous bug tracking and fixing for NeOn partners	Infrastructure for continuous bug tracking and bug fixing management available for the consortium. A bug fixing methodology shall accompany this infrastructure.	Critical	NeOn TA
2.4.1.5.2	Infrastructure for continuous bug tracking and fixing for the public	Infrastructure for continuous bug tracking and bug fixing management available for the public and participating developers	Critical	NeOn TA
2.4.1.5.2.2	Bugzilla	Bugzilla shall be used as bug tracking main infrastructure	Critical	NeOn TA
2.4.2	Software licensing	NeOn or at least part of it shall be licensed as open source software, with the corresponding license, e.g. (L)GPL.	Critical	NeOn TA
2.4.2.1	SourceForge	SourceForge shall be used to distribute the open source version of NeOn	Average	NeOn TA

Table 8: Development constraints

3.2. Specific requirements

This section details all the requirements of the NeOn architecture that shall be implemented. These requirements are aimed towards designers and testers.

3.2.1. External interfaces

This section of the NeOn architecture requirements specification has been focused on how NeOn shall be related with existing software in order to ease integration of this technology into the NeOn framework. With this aim, we approach the integration issue from the plugin paradigm. Figure 5 shows how we have classified this kind of requirements. A complete set of requirements on external interfaces is shown in Table 9. Integration of components has also been discussed in SEKT project (http://www.sekt-project.org/) and some of the components developed in the scope of this project will be integrated in the architecture of NeOn. The techniques used in SEKT will be analysed in order to integrate the different components in the NeOn architecture.

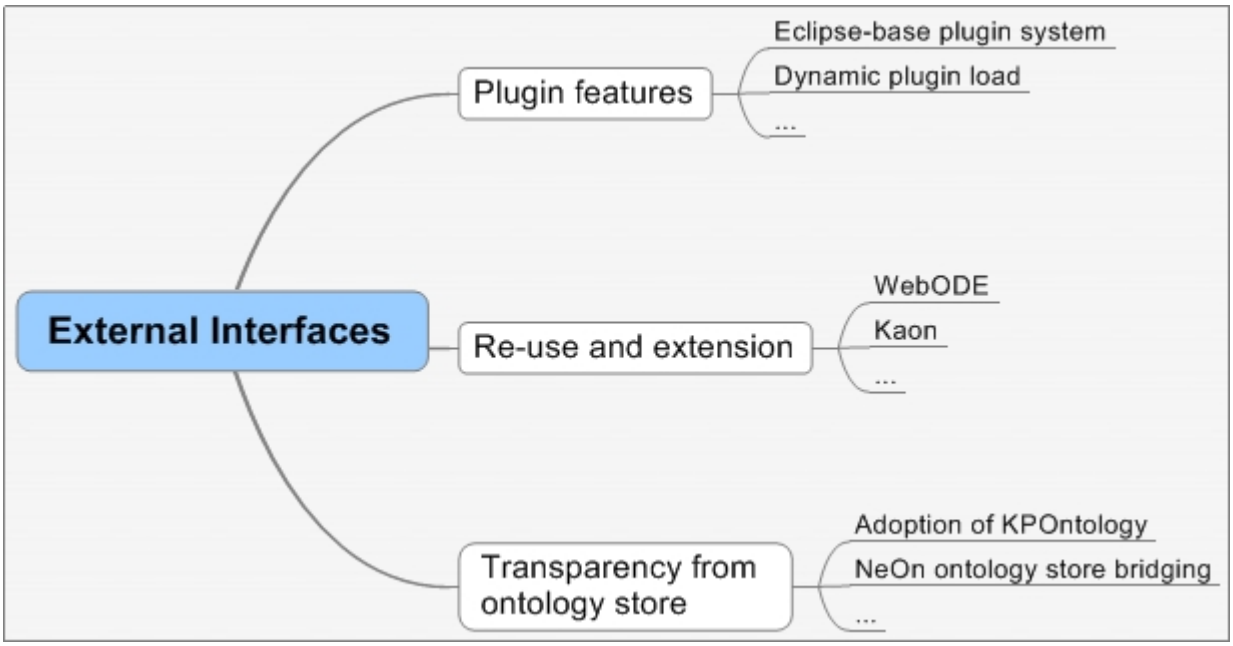

Figure 5: Classification of external interfaces requirements

NeOn shall inherit from Eclipse, on top of which the NeOn infrastructure will be implemented, a simple plugin mechanism that will allow easy integration of new components. This plugin mechanism will be endowed with summarization capabilities that will provide information about the plugin properties as required.

On the other hand, consortium members count with a certainly representative sample of Semantic Web and information integration technology that shall be reused and extended in the context of NeOn. Examples of this technology are e.g. KAON[3], Text2Onto, FOAM, ORAKEL or GATE[4]. Nevertheless, the use of such technology shall not be enforced but take place as a natural consequence of specific user needs. In this direction, the NeOn case studies will have a prominent role.

Directly related to section 3.2.2.1, NeOn shall be transparent from the different types of existing ontology stores. Thus, the main ontology stores shall be supported like e.g. JENA or Sesame.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	External Traceability	Internal Traceability
3.1.1	Plugin Features	Plugin System	NeOn shall be composed by means of plugins in order to allow easy integration of new components	Critical	FR	GUI Components	D6.2.1	2.1.1.2
3.1.1.1	Plugin Features	Eclipse-based plugin system	NeOn shall adopt Eclipse's plugin mechanism	Critical	FR	GUI Components	D6.2.1	2.1.1.3
3.1.1.2	Plugin Features	Dynamic plugin load	Plugin installation shall be a simple process	Critical	NFR	GUI Components	D6.2.1	2.1.1.2

[3] http://kaon2.semanticweb.org/ (Ontology management tool)

[4] http://gate.ac.uk/ (GATE Information extraction library)

3.1.1.3	Plugin Features	Plugin summarization	NeOn shall provide means to summarize the functionalities and characteristics of plugins	Average	FR	GUI Components	D6.2.1	2.1.1.2
3.1.2	Extension of state-of-the-art semantic technology	Re-use and extension	Re-use and extension of existing technology	Critical	FR	System-wide	D6.2.1, several applications are already as a NeOn plugin	2.1.1.2
3.1.2.2	Extension of state-of-the-art semantic technology	KAON2	NeOn shall re-use and extend KAON2	Average	FR	Engineering components	D6.10.1	2.1.1.2
3.1.2.3	Extension of state-of-the-art semantic technology	OntoBroker and OntoStudio	NeOn shall re-use and extend Ontobroker and Ontostudio	Average	FR	Engineering components	D6.10.1	2.1.1.2
3.1.2.4	Extension of state-of-the-art semantic technology	GATE	NeOn shall re-use and extend GATE	Average	FR	Engineering components	D6.10.1	2.1.1.2
3.1.2.5	Extension of state-of-the-art semantic technology	XML-based repository	NeOn shall re-use and extend XML-based repository from SAG	Average	FR	Infrastructure services	D6.10.1	2.1.1.2
3.1.3	Transparency from ontology store	Transparency from ontology store	NeOn ontology repository shall be transparent from actual implementation	Critical	FR	Infrastructure services	D6.2.1	3.2.1.1 3.2.1.2
3.1.3.2	Transparency from ontology store	NeOn ontology store bridging	The NeOn ontology repository shall be bridged to a number of existing technologies	Average	FR	Infrastructure services	D6.2.1	3.2.1.1 3.2.1.2

Table 9: External interfaces

3.2.2. Detailed functions of the NeOn architecture

In this section we detail the main functions, introduced in section 3.1.2, that the NeOn infrastructure shall perform. The next sections show the complete collection of requirements on the functions of the NeOn architecture.

3.2.2.1 Geographically-transparent access to Infrastructure services

NeOn shall allow access to ontological and non ontological information pieces transparently from their location. Load and storage of information resources will be virtualized by the NeOn infrastructure aiming to abstract away the actual location of these resources. In this direction, the NeOn Infrastructure services will internally manage the physical resource where information is actually stored by means of a virtual file system which provides the user with a unique storage space. Figure 6 provides a high-level description of these statements.

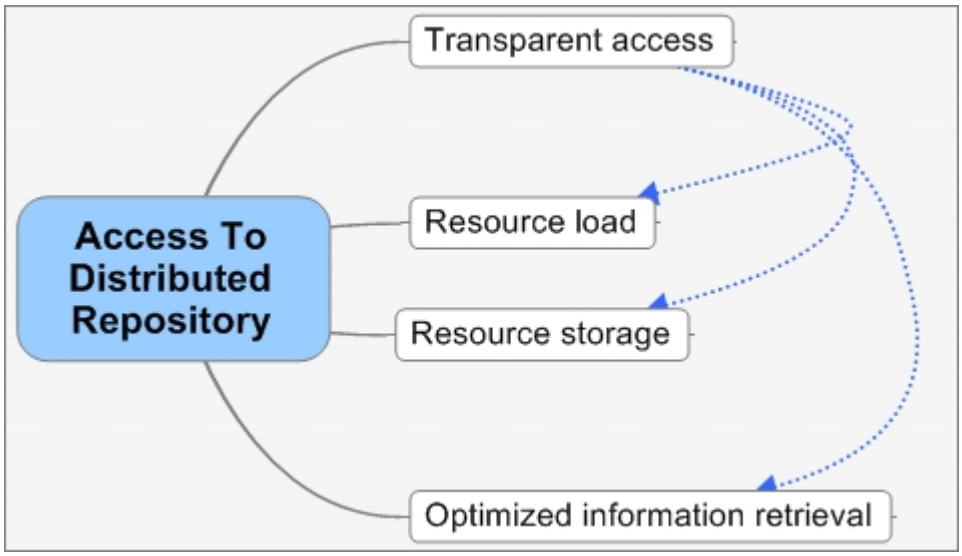

Figure 6: Transparent access to Infrastructure services

Distributed systems like NeOn need to implement optimizations that allow fast recovery of information since it will usually be geographically dispersed. These optimizations will extend the basic mechanism of information retrieval in terms of enhanced performance and flexibility. Foreseen optimizations are:

- **High availability of information resources:** The NeOn Infrastructure services will automatically distribute and maintain replicas of information resources among the nodes of the system. This will allow increasing performance of information retrieval by means of enabling local access to information.
- **Information caching:** Information resources will be cached once they are accessed by the first time. This will facilitate access to frequently used information. At the same time, it will require establishing a cache expiration time to dispose of unused cached information.

In both cases, NeOn shall implement a mechanism that ensures replica and cache consistency. Additionally, the distributed NeOn repository will provide an API to programmatically allow management of the knowledge resources contained within. The requirement list related with the access to the infrastructure services provided by NeOn is shown in Table 10.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.1	Access to Infrastructure services	Transparent access to information resources	NeOn shall allow access to ontological and non ontological information pieces transparently from their location	Critical	FR	Infrastructure services	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.1
3.2.1.1	Access to Infrastructure services	Infrastructure services access API	NeOn Infrastructure services API shall provide access to knowledge contained in the repository	Critical	FR	Infrastructure services	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.1
3.2.1.2	Transparent access to Infrastructure services	Resource transparent storage	NeOn shall provide transparent access to resource storage. The NeOn Infrastructure services shall	Critical	NFR	Infrastructure services	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.1

			internally manage the physical location where the resource is actually stored.					
3.2.1.3	Transparent access to Infrastructure services	Resource transparent load	NeOn shall provide a virtual file system which abstracts away the actual resource location, offering a virtual unique storage space	Critical	NFR	Infrastructure services	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.1
3.2.1.4	Access to Infrastructure services	Optimized information retrieval	NeOn shall implement optimizations to the basic mechanism of information retrieval that allow to increase performance and flexibility	Avera ge	FR	Infrastructure services	NeOn TA	2.1.1.1
3.2.1.4 .1	Access to Infrastructure services	High availability of information resources	The NeOn architecture shall automatically distribute replicas of information resources to facilitate local access to information.	Avera ge	FR	Infrastructure services	NeOn TA	2.1.1.1
3.2.1.4 .1.1	Access to Infrastructure services	Consistency maintenance of replicas	NeOn shall implement a device to ensure consistency of distributed replicas	Avera ge	FR	Infrastructure services	NeOn TA	2.1.1.1
3.2.1.4 .2	Access to Infrastructure services	Caching	The NeOn architecture shall implement a caching mechanism to improve performance	Avera ge	FR	Infrastructure services	NeOn TA	2.1.1.1

Table 10: Access to Infrastructure Services

3.2.2.2 Support of heterogeneous knowledge

Four different types of information sources will be managed by NeOn, i.e. unstructured content like e.g. free text, structured knowledge without clear semantics, formal knowledge i.e. ontologies and data, and data annotations. In summary, we can classify knowledge as ontological and non-ontological knowledge. As shown in Figure 7, NeOn shall provide the necessary means to load and store non-ontological resources into the NeOn knowledge model, including methods that allow conversion from the original format into the NeOn data model. Additionally NeOn shall provide operational transparency, i.e. operations like e.g. query execution shall be executed transparently from the particular type of knowledge and information repository. Table 11 represents the requirements extracted for accessing different types of knowledge.

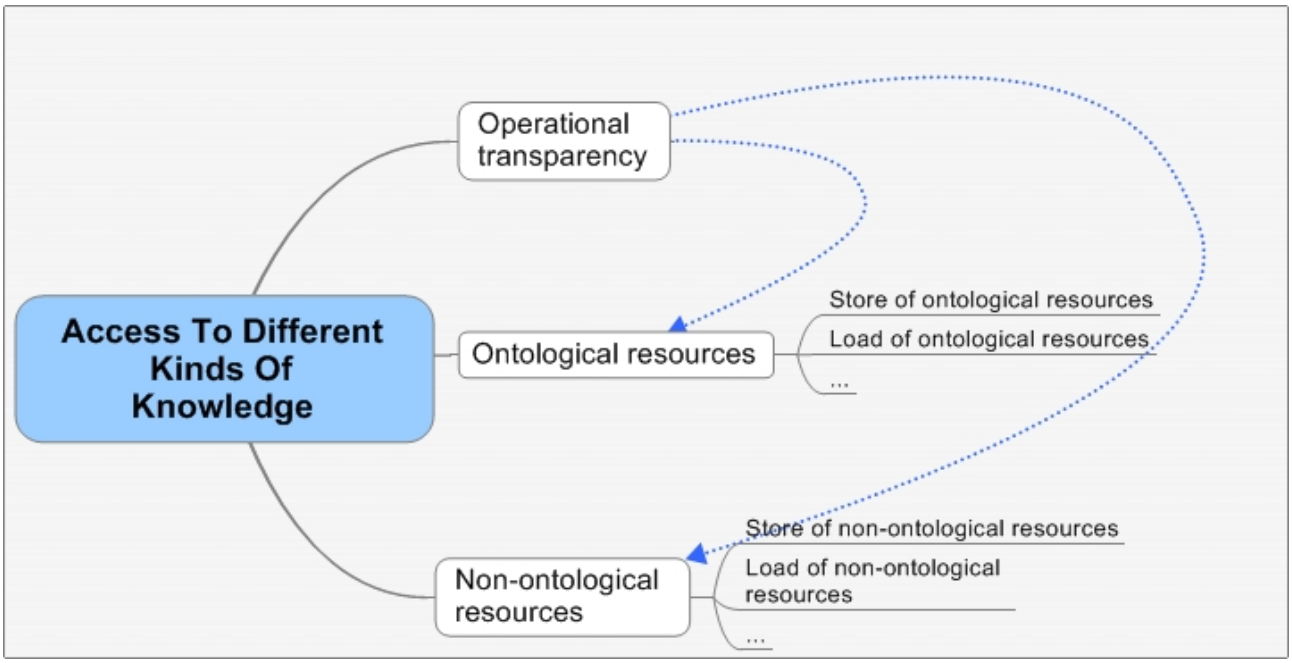

Figure 7: Support of heterogeneous knowledge

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.2	Support of heterogeneous knowledge	Access to the different kinds of knowledge	NeOn shall provide access to the main four different kinds of information i.e. unstructured content, structure knowledge without a clear semantics, formal knowledge (ontologies and data), and semantic annotations	Critical	FR	Infrastructure services	D8.3.1, 3.8 / 3.9 in D7.1.2	2.1.1.5
3.2.2.1	Support of heterogeneous knowledge	Operational transparency	Operations, like the query execution mechanism of NeOn, shall be transparent from the kind of information and repository	Critical	NFR	Engineering components	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.5
3.2.2.2	Reuse of legacy non-ontological resources	Load non-ontological resources	NeOn shall provide facilities to load non-ontological resources in the NeOn knowledge model	Critical	FR	Engineering components Infrastructure services	D8.3.1, 3.8 / 3.9 in D7.1.2	2.1.1.5
3.2.2.2 .1	Reuse of non-ontological resources	Converting non-ontological resources	NeOn shall provide the appropriate conversion methods between the non-ontological resources and the specifics of the NeOn data model	Critical	FR	Engineering components	D8.3.1, 3.8 / 3.9 in D7.1.2,	2.1.1.5

3.2.2.2.2	Reuse of non-ontological resources	Converting MS Access resources	Main Spanish pharmaceutical databases are Access DBs. Tools to create, maintain and populate ontologies from these databases should be provided.	Critical	FR	Engineering Components	D8.2.1 (Semantic Nomenclator req)	2.1.1.5
3.2.2.3	Reuse of non-ontological resources	Store non-ontological resources	NeOn shall provide facilities to store non-ontological resources in the NeOn knowledge model	Critical	FR	Engineering components	D8.3.1 and D7.4.1 Needed in both use cases	2.1.1.5
3.2.2.4	Reuse of existing ontological resources	Load ontological resources	NeOn shall provide means to load ontological resources in the NeOn knowledge model	Critical	FR	Engineering components	D8.3.1 and D7.4.1 Needed in both use cases,	2.1.1.5
3.2.2.5	Reuse of ontological resources	Store ontological resources	NeOn shall provide facilities to store ontological resources in the NeOn knowledge model	Critical	FR	Engineering components	D8.3.1 and D7.4.1 Needed in both use cases,	2.1.1.5
2.2.5.1	Reuse of ontological resources	Store ontological resources	Export in other formats	Critical	FR	Engineering Components	4.4.5 in D7.1.1 UC-4 in D7.4.1	2.1.1.5

Table 11: Support of Heterogeneous Knowledge

3.2.2.3 Dual language approach

NeOn shall support OWL-based and Rule languages.

3.2.2.4 Ontology modularization

Modularization is essential to support collaborative ontology editing and browsing, reuse and selection. Within the networked scenario, ontologies are mainly built in distributed locations, which means naturally, they are modules in a network of ontologies. At the same time, ontologies are difficult to manage especially when they are interconnected in a network. Interdependencies between modules of interconnected ontologies have to be made clear so that these modules can be easily managed (e.g., reused, shared). Ontology selection and reuse will play a major part in the future of the Semantic Web because with the increasing number of ontologies available over the internet, people will more likely use ontologies just like program libraries by selecting and reusing ontology modules [14] .

In Table 12 is shown that NeOn shall support ontology modularization and provide the means to treat modules as first class citizens in the ontological realm. By extension, developers will have the means to manage ontologies to the desired module granularity level, i.e. facilities like e.g. ontology load and store shall also be applicable to ontology modules.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.4	Ontology modularization	Management of ontology modularization	NeOn shall allow to structure ontologies using modules	Critical	FR	Engineering components	D8.3.1 (semantic nomeclator and invoicing use case), 4.3.2 in D7.1.1 UC8 in D7.4.1	
3.2.4.1	Ontology modularization	Treatment of ontology modules as first class citizens	NeOn shall allow users to manage ontologies to the module granularity level, i.e. ontology facilities like ontology load and store, shall be applicable to ontology modules, too.	Critical	FR	GUI Components	D8.3.1 (semantic nomeclator and invoicing use case), 4.3.2 in D7.1.1 UC8 in D7.4.1	

Table 12: Ontology Modularization Requirements

3.2.2.5 Multilinguality support

A fundamental feature, shared by the NeOn pharmaceutical and fishery case studies, is support for multilingual ontologies. It is a key requirement for their respective domains that the models described by these ontologies are available in a series of different languages. These requirements are shown in Table 13.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.5	Multilinguality Support	Multilinguality	NeOn shall support multilingual ontologies	Critical	FR	Engineering components	4.4.7 in D7.1.1 UC3 in D7.4.1, D8.2.1	
3.2.5.1	Multilinguality Support	Add new (multilingual) labels to the ontology elements	LabelTranslator for foreign ontologies (Spain). A Translation service which provides suggestions for the translation of ontology elements	Critical	FR	Engineering Components	D8.2.1 (Semantic Nomenclator req)	

Table 13: Multilinguality Requirements

3.2.2.6 Mapping support

NeOn shall provide means to easy the creation of alignment between entities of two or more globally inconsistent ontologies. In this direction, as in the case of ontology modules, mappings will be treated as first class ontology

citizens (see Figure 8), with their own lifecycle. This will allow proper manipulation, creation, editing, and deletion of mappings.

On the other hand, ontologies evolve and change, and existing mappings between entities belonging to different networked ontologies need to be updated. NeOn shall automatically check mapping consistency throughout their entire lifecycle.

Additionally, mappings can be implicitly present either by means of ontology properties and relations or other mappings. Treatment of mappings as first class ontology entities, together with reflexive properties, are expected to allow reasoning on ontology mappings which will allow to bring out this hidden alignment knowledge. These requirements are listed in Table 14.

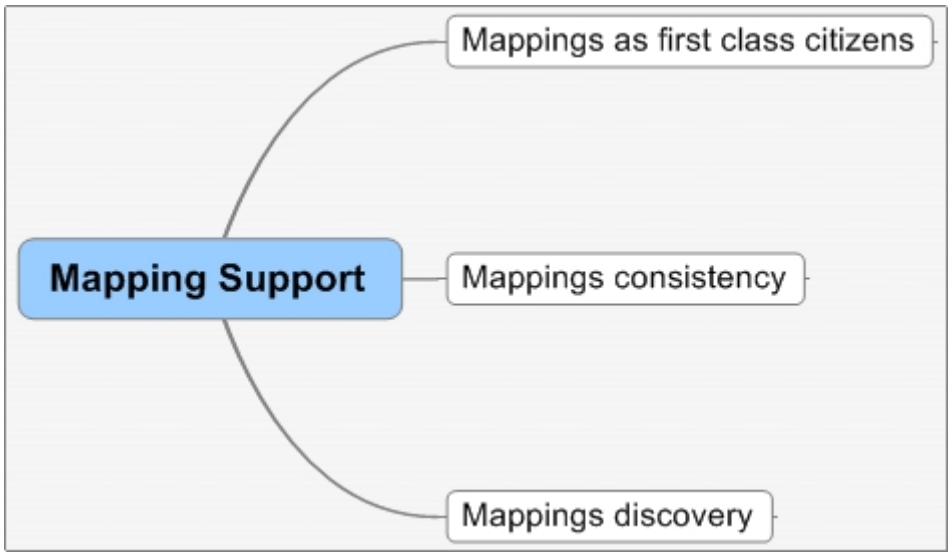

Figure 8: Requirements on mapping support

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.6	Mapping support	Mapping capabilities	NeOn shall provide techniques for simplifying the mapping process between entities of two or more ontologies that are globally inconsistent	Critical	FR	Engineering components	4.4.3 in D7.1.1 UC6 in D7.4.1, D8.2.1	2.1.1.2
3.2.6.1	Mapping support	Treatment of mappings as first class citizens	NeOn shall provide means to treat mappings as ontological entities themselves, with their own lifecycle, in order to allow easy manipulation, creation, editing, and deletion.	Critical	FR	Engineering components	4.4.3 in D7.1.1 UC6 in D7.4.1, D8.2.1	
3.2.6.2	Mapping support	Mapping consistency checking	NeOn shall automatically check the consistency of mappings throughout their lifecycle	Critical	FR	Engineering components	D8.3.1	
3.2.6.3	Mapping support	Discovery of implicit mappings	NeOn shall have reflexive capabilities which allow reasoning on ontology mappings	Critical	FR	Engineering components	D8.2.1 (Semantic Nomenclator req)	2.1.1.10

			with the aim of checking them as well as inferring and automatically applying new mappings					
3.2.6.4	Mapping Support	Manual Mapping	NeOn shall provide to the users tools for creating mappings between ontologies	Critical	FR	Engineering Components	D1.1.1	
3.2.6.5	Mapping Support	Mappings across networked ontologies	Mappings are inherent to the concept of ontology networking. NeOn shall provide the means to support them.	Critical	FR	Engineering components	4.4.3 in D7.1.1 UC6 in D7.4.1, D8.2.1, D8.3.1	

Table 14: Mapping Requirements

3.2.2.7 Support for contextualized ontologies

As introduced in previous sections, context support in networked ontologies is especially critical for NeOn. We define context as the set of all circumstances, properties, and facts within which the ontology has the desired semantics. Thus, (networked) ontologies are dependent on the context in which they are built or in which they are used. Additionally, different contexts can be valid for a particular ontology and all of them must be available when necessary. In Table 15 are shown the requirements that as a consequence, NeOn will provide the means to parameterize ontologies with respect to its different contexts.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.7	Support for contextualized ontologies	Context support in networked ontologies	Ontologies are dependent on the context in which they are built or in which they are used. NeOn shall support context in networked ontologies	Critical	FR	Engineering components	D8.2.1	
3.2.7.1	Support for contextualized ontologies	Multi-context ontologies	Different contexts shall be applied to networked ontologies	Critical	FR	Engineering components	D8.2.1	
3.2.7.2	Support for contextualized ontologies	Support for ontology context parameters	NeOn shall provide means to parameterize ontologies with respect to a given context	Critical	FR	Engineering components	D8.2.1	

Table 15: Contextual Ontologies Requirements

3.2.2.8 Lifecycle support for ontology development

The main goal of NeOn is to support knowledge acquisition and ontology development and maintenance throughout their lifecycle, extending standard ontology development facilities to the arena of networked knowledge, where networking appeals both to the physical distribution of the available resources and to the connections between the entities of different ontologies. Figure 9 summarizes these functions.

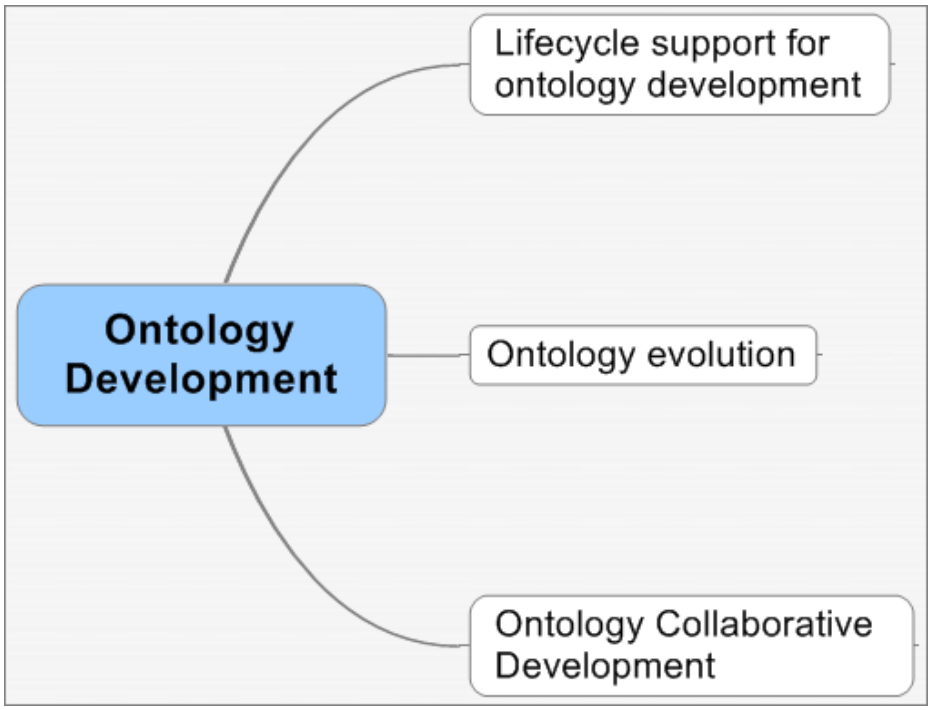

Figure 9: Functions for ontology development

The main functions that need to be supported are the following:

- **Networked ontology editing and building:** NeOn will extend the creation and editing of ontologies to the networked case. This task shall be assisted by means of consistency checkers and reasoners scaled to a network of ontologies.
- **Ontology Evolution:** NeOn will support the modification of ontologies and will keep track of these changes maintaining the consistency of the ontologies.
- **Ontology Collaborative Development:** NeOn will support collaboration between partners in the ontology development task.

Additionally, NeOn shall support collaborative ontology development by means of adopting the CVS metaphor for collaborative work on shared ontological resources. This includes the following functions:

- **Synchronization of ontology concurrent updates:** In a networked, collaborative environment, concurrent updates of shared information are a very real possibility. In such cases, the necessary means to synchronize changes must be provided. CVS-like techniques will be implemented that allow committing changes safely.
- **Consistency checking for ontology concurrent updates:** Safe commitment of ontology updates needs prior detection of conflicts between concurrent changes. Inconsistencies can also appear between the updated portion of the ontology and the rest of the knowledge contained within. Both need to be detected and, when possible, remedial actions shall be automatically applied.
- **Ontology versioning:** The CVS metaphor shall be supported for collaborative ontology development.

Finally, ontology changes shall be propagated across the network of ontologies with which a particular ontology is connected. For consistency reasons, it might be impractical to automatically update dependent ontologies but the change can be evaluated and reported to the ontology developer by means of the corresponding GUI (see section 3.1.1.2). The list of these requirements is shown in Table 16.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.8	Lifecycle support for ontology development	Knowledge Acquisition	NeOn shall support knowledge acquisition and ontology development	Critical	FR	GUI Components Engineering components	D7.4.1, D8.2.1	2.1.1.1, 2.1.1.2, 2.1.1.3, 2.1.1.4

3.2.8.1	Lifecycle support for ontology development	Extended ontology support to the networked paradigm	NeOn shall extend standard ontology development facilities to the distributed, networked case	Critical	FR	GUI Components Engineering components	NeOn R&V	
3.2.8.1 .1	Ontology Evolution	Networked Ontology editing and building	NeOn shall extend ontology creation and editing to the networked case	Critical	FR	GUI Components Engineering components	D7.4.1, D8.2.1, 4.3.1 in D7.1.1	
3.2.8.1 .1.1	Ontology Evolution	Networked Ontology editing and building	Create Ontology reengineering existing non ontological data model	Critical	FR	Engineering Components	4.3.1 in D7.1.1 UC5 in D7.4.1	
3.2.8.1 .1.2	Ontology Evolution	Networked Ontology editing and building	Create ontology for interfacing DB data	Critical	FR	Engineering Components	4.3.1 in D7.1.1 UC5 in D7.4.1, D8.2.1 Semantic Nomenclator	
3.2.8.1 .1.3	Ontology Evolution	Dynamic addition of new ontologies	NeOn technologies together NeOn methodology should provide a workflow which allows users the dynamic addition of new ontologies in the ontology network of the semantic application. NeOn should suggest and support the steps needed in this activity, i.e. suggest new ontologies that can be added --> mappings suggestions --> consistency checking	Critical	FR	Engineering Components	D8.2.1 Semantic Nomenclator req	
3.2.8.1 .1.4	Ontology Evolution	Ontology checking and consistency	NeOn shall assist the ontology editing and building process by providing users with consistency checkers and reasoners	Avera ge	FR	Engineering components	D7.4.1, D8.2.1	
3.2.8.1 .1.5	Ontology Evolution	Ontology checking and consistency	Provide suggestions to the user and email them	Low	FR	Engineering components	3.4.1 in D7.1.2	
3.2.8.1 .1.3.1	Ontology Evolution	Ontology Evaluation	Duplication detection for pair of concepts or instances over sensitive attributes	Critical	FR	Engineering components	4.4.1 in D7.1.1 UC11 in D7.4.1	

3.2.8.1.1.3.2	Ontology Evolution	Ontology Evaluation	String similarity support for pair of literal values attributes	Critical	FR	Engineering components	4.4.1 in D7.1.1 UC11 in D7.4.1	
3.2.8.1.1.3.3	Ontology Evolution	Ontology Evaluation	Support for the user to have two Ontologies at glance to visually compare them	Critical	FR	Engineering components	4.4.1 in D7.1.1 UC11 in D7.4.1	
3.2.8.1.1.3.4	Ontology Evolution	Ontology Evaluation	Generate matrix of values for Ontology structural properties	Critical	FR	Engineering components	4.4.1 in D7.1.1 UC11 in D7.4.1	
3.2.8.1.2	Ontology Evolution	Ontology selection and reuse	NeOn shall provide the means to access one or several ontology repositories for ontology selection. Then, the selected ontology/ies will be imported by means of the ontology load functionality	Critical	FR	GUI Components Infrastructure services	UC1 in D7.4.1, new UC	
3.2.8.1.2.1	Ontology Evolution	Ontology selection and reuse	Create Ontology integrating existing Ontologies	Critical	FR	Engineering Components	New UC	
3.2.8.1.2.2	Ontology Evolution	Ontology selection and reuse	Assisted transformation of existing data model to Ontology model	Critical	FR	Engineering Components	4.3.1 in D7.1.1 UC5 in D7.4.1	
3.2.8.1.3	Ontology Evolution	Selection and reuse of networked ontologies	NeOn shall allow to select and reuse ontologies and ontology entities from a network of ontologies	Critical	FR	GUI Components Engineering components	4.3.1 in D7.1.1 UC5 in D7.4.1, D8.2.1	2.1.2.1.5
3.2.8.1.7	Ontology Evolution	Ontology Restructuring	Test Ontology consistency before committing the changes	Critical	FR	Engineering Components	4.4.2 in D7.1.1 UC10 in D7.4.1	
3.2.8.2	Ontology collaborative development	Ontology collaborative development	NeOn shall allow collaborative ontology development	Critical	FR	GUI Components Engineering components	Semantic Nomenclator, Invoicing and FAO use cases	
3.2.8.2.1	Ontology collaborative development	Adoption of CVS development model	NeOn shall adopt the CVS metaphor for collaborative work on shared ontological resources	Critical	FR	GUI Components Infrastructure services	NeOn TA	
3.2.8.2.1.1	Ontology collaborative development	Synchronization of ontology concurrent updates	NeOn shall allow synchronization of ontologies edited by multiple users	Critical	FR	Infrastructure services	Semantic Nomenclator, Invoicing and FAO use cases	

3.2.8.2.1.2	Ontology collaborative development	Consistency checking for ontology concurrent updates	NeOn shall provide the means to detect the introduction of inconsistencies when a shared ontology is updated. In such case remedial actions should be automatically taken (when possible) and affected users notified	Critical	FR	Engineering components Infrastructure services	D8.2.1 Invoicing use case	
3.2.8.2.1.3	Ontology collaborative development	Ontology versioning	NeOn shall provide support for ontology versioning based on the CVS metaphor	Critical	FR	GUI Components Infrastructure services	D8.2.1 Invoicing use case	
3.2.8.2.1.3.1	Ontology collaborative development	Global awareness of local ontology versioning	NeOn shall ease the adoption of new versions of an ontology in an already existing network of ontologies	Average	FR	GUI Components Infrastructure services	D8.2.1 Invoicing use case	
3.2.8.2.2	Ontology collaborative development	Change propagation in a ontology network	NeOn shall automatically propagate updates in a particular ontology throughout the network of interrelated ontologies	Average	FR	Engineering components Infrastructure services	D8.2.1 Invoicing use case	
3.2.8.2.2.1	Ontology collaborative development	Ontology network consistency maintenance	NeOn shall maintain partial and local consistency among a network of ontologies	Critical	FR	Engineering components Infrastructure services	D8.2.1 Invoicing use case	
3.2.8.2.3	Ontology collaborative development	Social network support	Support for social networks capabilities: wikis… How to use it in specific communities and in the ontology edition	Critical	FR	Engineering Components	D8.2.1	2.1.1.10
3.2.8.2.4	Ontology collaborative development	Argumentation Service	Argumentation support: design decisions	Critical	FR	Engineering Components	D8.2.1	

Table 16: Lifecycle ontology requirements

3.2.2.9 Reasoning and inference

The NeOn backend shall provide reasoning capabilities on top of the aggregated knowledge contained in a network of ontologies. Two different language approaches are developed within the NeOn toolkit and also two rule language approaches are developed. Support for SWRL rules and F-Logic rules is required in the NeOn toolkit.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.9	Reasoning & Inference	Reasoning & Inference functionality	NeOn should provide Reasoning & Inference functionalities.	Critical	FR	Engineering components	D8.2.1	
3.2.9.1	Reasoning & Inference	Unique data model for modelling and reasoning infrastructure	NeOn reasoning and inference operates on the same data model as the NeOn knowledge formation	Critical	NFR	Engineering components	D8.2.1	
3.2.9.2	Reasoning & Inference	Reasoning with up-to-date data	NeOn shall ensure ontology and data are consistent for reasoning and inference	Critical	NFR	Engineering components	D8.2.1	

Table 17: Reasoning and inference requirements

3.2.2.10 Query support

Standard ontology query languages like e.g. SPARQL shall be supported by NeOn, allowing the retrieval of information contained within. The NeOn toolkit should provide functionalities to help users to make queries in those languages.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.10	Query support	Query support	NeOn shall provide the means, e.g. by supporting standard query languages, to query the information contained within	Critical	FR	Engineering components	3.4.2 in D7.1.2	2.1.1.11
3.2.10.1	Query Support	Query for individual	Allow users to query for an individual	Critical	FR	Engineering Components	3.4.2 in D7.1.2	
3.2.10.2	Query Support	Advanced Query	Query using Boolean operators, phrase matching and query refinement	Critical	FR	Engineering Components	3.4.2 in D7.1.2	
3.2.10.3	Query Support	Advanced Query	Generate RSS feed from query results	Low	FR	Engineering Components	3.4.1 in D7.1.2	
3.2.10.4	Query Support	Advanced Query	Email query results	Low	FR	Engineering Components	3.4.1 in D7.1.2	

Table 18: Query Support

3.2.2.11 Question formulation

Question formulation functionality shall be built on top of ontology querying that will allow users to build more complex and expressive questions for information retrieval. Natural language questions shall be supported that after automatic translation into the corresponding ontology query standard will be submitted to the query service and run against the information contained in the system.

Usually, this kind of interfaces show a gap between what the users tried to express in the form of a free text question and the information retrieved by the system as the answer to this question. Thus, it is necessary to explain the results obtained in terms of the question helping the user understand the result set. These requirements are shown in the table below, Table 19.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.11	Question formulation	Question formulation	NeOn shall provide the means that allow users to express queries naturally, e.g. by free test questions	Average	FR	GUI Components	NeOn R&V	2.1.1.12
3.2.11.1	Question formulation	Answer Explanation	NeOn shall explain the results upon question answering in the terms of the formulated question	Critical	FR	Engineering components	NeOn R&V	2.1.1.12

Table 19: Question Formulation

3.2.2.12 Semantic annotation

NeOn shall incorporate semantic annotation functionalities (see Figure 10), which will provide information characterization in terms of the model described in a set of networked ontologies. The different types of knowledge sources available in NeOn apart from ontological knowledge, i.e. unstructured knowledge like e.g. free text, and structured, non-semantic knowledge like e.g. an XML store, will be subject of semantic annotation. Different annotation techniques shall be provided, including data and text mining.

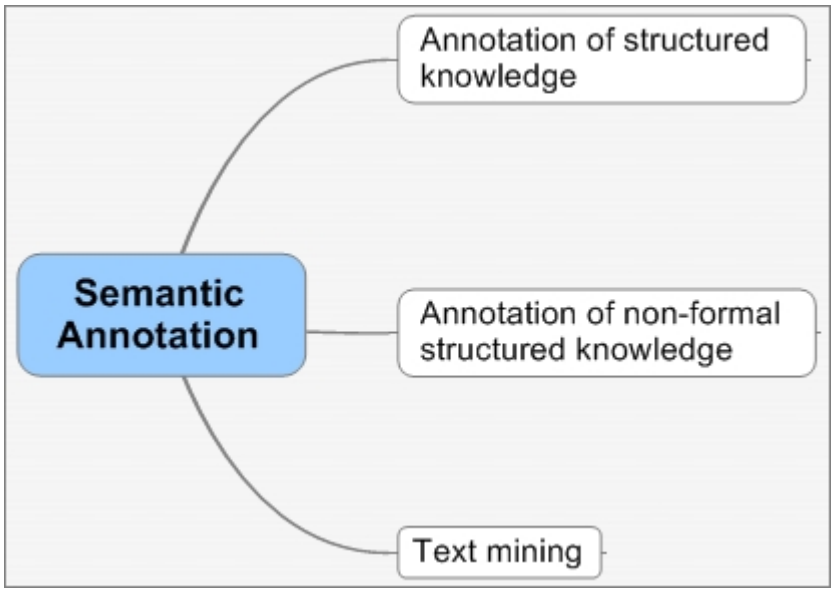

Figure 10: Semantic annotation functionalities

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.12	Semantic annotation	Semantic annotation	NeOn shall allow to annotate information sources	Critical	FR	Engineering components	D8.3.1, 3.4.1 in D7.1.2, 4.4.1 in D7.1.1	2.1.1.6, 2.1.1.5

3.2.12.1	Data annotation	Semantic Annotation of unstructured knowledge	NeOn shall allow to annotate unstructured content, e.g. text	Critical	FR	Engineering components	D8.3.1, D7.1.2, D7.4.1	2.1.1.6, 2.1.1.5
3.2.12.2	Data annotation	Semantic Annotation of non-formal structured knowledge	NeOn shall allow to annotate structured knowledge without clear semantics, e.g. database and XML repositories	Critical	FR	Engineering components	D8.3.1, D7.1.2, D7.4.1	2.1.1.6, 2.1.1.5
3.2.12.3	Data annotation	Rank results based on annotation data	Create rankings from the annotated data	Average	FR	Engineering Components	3.4.1 in D7.1.2	
3.2.12.4	Ontology annotation	Annotation of ontologies	Annotation of the edited Ontology element by mean of free text (with multilingual support), and image association	Average	FR	Engineering Components	4.4.1 in D7.1.1 2.4.1.13 in D7.1.2	
3.2.12.5	Ontology annotation	Ontology Population	Use text corpora to suggest candidate instances and relations between identified ones	Critical	FR	Engineering Components	4.4.2 in D7.1.1 UC10 in D7.4.1	
3.2.12.5.1	Ontology annotation	Ontology Population	Selection of suggested instance to include in the Ontology	Critical	FR	GUI Components	4.4.2 in D7.1.1 UC10 in D7.4.1	
3.2.12.5.2	Ontology annotation	Ontology Population	Some web services could be accessed to extract new information	Average	NFR	Engineering Components	D8.2.1 Semantic Nomenclator req	

Table 20: Semantic Annotation Requirements

3.2.2.13 Access management to information resources

Access rights to information resources managed by NeOn need to be administrated. Case studies have shown that not all kinds of users can have access to all types of information. For example, a particular ontology describing an invoice model must remain stable unless a change in the law or the business dynamics requires it to be updated. Hence, the capability to see, modify, or delete complete or parts of ontologies shall be constrained by the author.

3.2.2.14 User profiling

NeOn shall accumulate information about user profiles during system interaction and provide a customized set of functionalities according to the inferred profile.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.14	User profiling	User profiling	NeOn shall accumulate information about user profiles during system interaction and provide a	Average	FR	Engineering components	D8.2.1	2.1.1.3, 2.1.1.13

			customized set of functionalities according to that profile					

3.2.2.15 Ontology summarization

NeOn shall provide users with the means to summarize ontology state and properties.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.15	Summarization	Ontology briefing	NeOn shall provide users with the means to summarize the state and characteristics of the ontology	Average	FR	Engineering components	NeOn R&V	2.1.1.8

Table 21: Summarization Requirements

3.2.2.16 Provenance support

Ontology characteristics and state provided by the functionality described in section 3.2.2.15 are closely related to ontology provenance information. Ontology traceability shall be kept by automatically storing relevant information like authors or contributors, as well as versioning information. This information will be reported to ontology developers by means of the corresponding GUI.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.16	Provenance Support	Ontology provenance service	NeOn shall keep ontology traceability by automatically storing relevant information like authors or contributors, as well as timeline and versions	Average	FR	Engineering components	3.4.3 in D7.1.2, D8.2.1	2.1.1.13
3.2.16.1	Provenance Support	Trust Service	Provenance trust: Evaluation and rating of the knowledge gathered	Critical	FR	Engineering Components	D8.2.1 Semantic Nomenclator Req.	

Table 22: Provenance Requirements

3.2.2.17 System documentation and help

The different user types of NeOn shall be provided with extensive help and training materials, including tutorials, which will allow improving understanding of the system, either by means of providing information that solves specific problems or by helping the novice ontology developer get acquainted with the system. A help subsystem shall be built that provides contextualized help, in terms of the materials available in each situation and the particular user profile.

3.2.2.18 Support for different client types

Rich and thin clients will be supported by the NeOn architecture, as described in section 3.1.1.2.

3.2.2.19 Service access to the NeOn backend

Access to the NeOn backend shall be provided both to rich and thin clients. A programmatic Java API will be provided to rich clients while a socket-based interface will be available for thin clients.

Req. #	Functionality	Requirement Extracted	Description	Imp.	Type	Layer of architecture	External Traceability	Internal Traceability
3.2.19	Access to Engineering components	Middleware layer accessible by an API	The middleware layer API shall provide programmatic access to the NeOn Engineering components	Critical	FR	Engineering components	D8.3.1	2.1.1.4

Table 23: Functions provided by the NeOn infrastructure

3.2.3. Performance requirements

This section contains a number of factors that need to be taken into account in order to maximize performance during the execution of the functionalities provided by NeOn. Additionally, the key aspects where performance is required are highlighted. Figure 11 shows these factors and aspects.

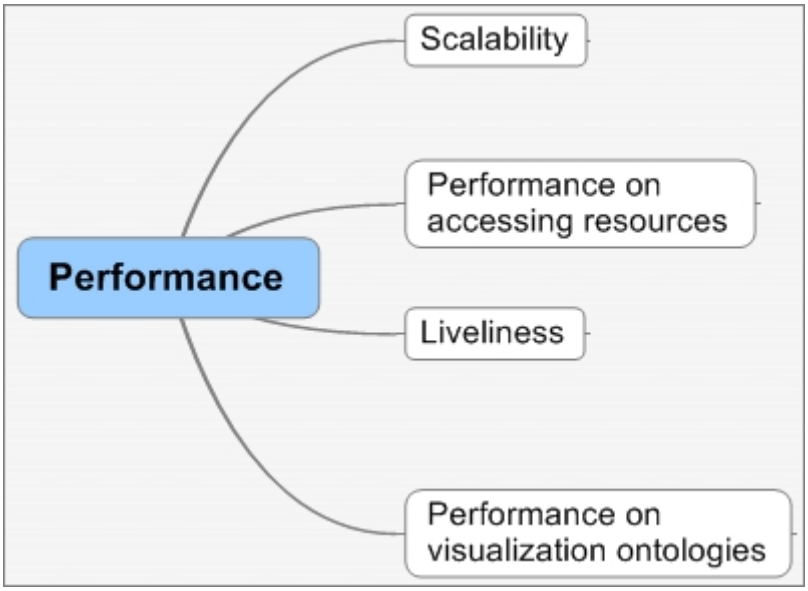

Figure 11: Performance factors and application contexts

Performance is closely related to scalability. If a distributed system like NeOn is not scalable, then, with high probability, it will not be performant either. There are three different ways in which NeOn shall be scalable. Nevertheless, there is a limit up to which a distributed system can be scaled. Above that limit, performance decreases.

- **Load scalability:** NeOn shall allow expansion and contraction of its resource pool without performance decrease. This applies both to ontological and non-ontological resources.
- **Geographic scalability:** NeOn shall maintain usefulness and performance regardless of how apart its resources and components are located.

- **Administrative scalability:** NeOn shall allow increasing concurrent access without performance loss.

Additionally, performance shall not suffer from the heterogeneity of the knowledge resources maintained by NeOn. It shall remain homogenous with respect to the nature of this knowledge, unstructured, structured, or formal. Operational transparency regarding the sources implied in the execution of a NeOn service shall be observed in such way that performance be independent from them.

We have identified a number of circumstances where performance is certainly a critical requirement:

- **Load and storage of ontological and non-ontological resources** shall be timely and efficient.
- **Ontology browsing and visualization:** NeOn visualization techniques shall guarantee quick access to the required ontology elements and easily browse taxonomy and data. By extension, this also applies to networked ontologies.

The execution of NeOn services shall also comply with the following properties:

- **Liveliness:** NeOn services and functions shall guarantee that results of execution are returned in a limited amount of time.
- **Progress awareness:** NeOn GUIs shall keep users informed of the current state of service execution.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	Internal Traceability
3.3.1	Access to distributed Information repositories	Scalability	NeOn shall be scalable	Critical	NFR	Infrastructure services Engineering components	3.2.1 3.4.1.1
3.3.1.1	Access to distributed Information repositories	Load scalability	NeOn shall allow to expand and contract its resource pool with no performance decrease	Critical	NFR	Infrastructure services	3.2.1 3.4.1.1
3.3.1.1 .1	Access to distributed Information repositories	Load scalability for non-ontological resources	NeOn shall allow to expand and contract its pool of non formal resources with no performance decrease	Critical	NFR	Infrastructure services	3.2.1 3.4.1.1
3.3.1.1 .2	Access to distributed Information repositories	Load scalability for ontological resources	NeOn shall allow to expand and contract its ontology pool with no performance decrease	Critical	NFR	Infrastructure services	3.2.1 3.4.1.1
3.3.1.2	Access to distributed Information repositories	Geographic scalability	NeOn shall maintain usefulness and performance, regardless how apart its resources and components are	Critical	NFR	Infrastructure services Engineering components	3.2.1 3.4.1.1

3.3.1.3	Access to distributed Information repositories	Administrative scalability	NeOn shall allow increasing concurrent access without performance loss	Critical	NFR	Infrastructure services	3.2.1 3.4.1.1
3.3.1.4	Access to distributed Information repositories	Scalability limitations	There is a limit up to which we can scale a distributed system, and above that the performance of the system degrades	Critical	NFR	Infrastructure services Engineering components	3.2.1 3.4.1.1
3.3.2	Support of heterogeneous knowledge	Performant access to heterogeneous knowledge	Performance of access to knowledge repository shall be homogenous regardless the nature of this knowledge, i.e. unstructured, structured, or formal	Average	NFR	Infrastructure services Engineering components	3.2.2 3.4.1.2
3.3.2.1	Support of heterogeneous knowledge	Performant operational transparency	Operational transparency shall not imply performance loss	Average	NFR	Infrastructure services Engineering components	3.2.2.1 3.4.1.2
3.3.2.2	Reuse of ontological Resources	Efficient reuse of existing ontological resources	NeOn shall provide efficient support for reuse of existing ontological resources	Critical	NFR	Infrastructure services Distributed component	3.2.2.4 3.4.1.2.1
3.3.2.2.1	Reuse of ontological Resources	Performant load of ontological resources	Loading ontological resources into the NeOn knowledge model shall be timely and efficient	Critical	NFR	Infrastructure services Distributed component	3.2.2.4 3.4.1.2.1
3.3.2.2.2	Reuse of ontological Resources	Performant storage of ontological resources	Storing updated ontological resources in the repository shall be timely and efficient	Critical	NFR	Infrastructure services	3.2.2.5 3.4.1.2.1
3.2.2.3	Reuse of non-ontological resources	Efficient reuse of existing non-ontological resources	NeOn shall provide efficient support for reuse of existing non-ontological resources	Critical	NFR	Infrastructure services Distributed component	3.2.2.2 3.4.1.2.2
3.3.2.3.1	Reuse of non-ontological resources	Performant load of non-ontological resources	Loading non-ontological resources into the NeOn knowledge model shall be timely and	Critical	NFR	Infrastructure services Distributed component	3.2.2.2 3.4.1.2.2

			efficient				
3.3.2.3.1	Reuse of non-ontological resources	Performant storage of ontological resources	Storing updated non-ontological resources in their native repository shall be timely and efficient	Critical	NFR	Infrastructure services	3.2.2.3 3.4.1.2.2
3.3.3	Execution of NeOn services and functionalities	Liveliness	All NeOn services and functions shall guarantee that execution results are returned in a limited amount of time	Critical	NFR	Engineering components	
3.3.3.1	Execution of NeOn services and functionalities	Progress user-awareness	NeOn shall keep users informed of service execution progress	Average	NFR	GUI Components	
3.3.4	Visualization	Fluent ontology browsing and visualization	NeOn visualization techniques shall guarantee quick access to the required ontology elements and easily browse ontology taxonomy and data	Critical	NFR	GUI Components	
3.3.4.1	Visualization	Fluent networked ontology browsing and visualization	NeOn visualization techniques shall guarantee quick access to the required ontology elements and easily browse networked ontologies taxonomy and data	Critical	NFR	GUI Components	

Table 24: Performance requirements

3.2.4. Information repository requirements

Next, requirements on the information repositories maintained by NeOn are described. The whole list of requirements is contained in Table 25. NeOn information repositories will be distributed and heterogeneous. Additionally NeOn shall manage information contained in other information repositories. The most representative of these non-ontological information repositories are relational databases. NeOn shall support widespread database systems like Oracle, MySQL, and Postgres. Special interesting would be to manage XML information repositories built on top of SAG's Tamino.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	Internal Traceability
3.4.1	Information repositories	Information repositories	NeOn shall manage and run on information repositories	Critical	FR	Infrastructure services	2.1.1.1 2.1.1.5 2.1.2.3.4
3.4.1.1	Information repositories	Distributed Information repositories	Information repositories shall be geographically distributed	Critical	FR	Infrastructure services	2.1.1.1 3.2.1
3.4.1.2	Heterogeneous knowledge sources	Heterogeneous knowledge sources	NeOn shall manage heterogeneous knowledge sources	Critical	FR	Infrastructure services	2.1.1.5 3.2.2
3.4.1.2.1	Ontological repositories	Ontological repositories	NeOn shall manage formal knowledge in the form of ontologies	Critical	FR	Infrastructure services	2.1.1.5 2.1.2.3.4.1 3.5.1.5.1 3.5.1.5.2
3.4.1.2.1.1	Ontological repositories	Ontological repository stores	NeOn shall support widespread ontology store, e.g. JENA, Sesame, OWLIM, Kowari	Critical	FR	Infrastructure services	2.1.1.5 3.1.3.2
3.4.1.2.2	Non-ontological repositories	Non-ontological repositories	NeOn shall manage non-formal, legacy information sources	Critical	FR	Infrastructure services	2.1.1.5 2.1.2.3.4.2
3.4.1.2.2.1	Non-ontological repositories	Free text	NeOn shall support free text information sources	Critical	FR	Infrastructure services	2.1.1.5
3.4.1.2.2.2	Non-ontological repositories	Structured knowledge	NeOn shall support structured knowledge with no clear semantics	Critical	FR	Infrastructure services	2.1.1.5

3.4.1.2.2.2.1	Non-ontological repositories	XML knowledge	NeOn shall support Software AG XML information repository	Average	FR	Infrastructure services	2.1.1.5 3.1.2.6
3.4.1.2.2.2.2	Non-ontological repositories	Relational databases	NeOn shall support widespread database systems, e.g. Oracle, MySQL, Postgres	Average	FR	Infrastructure services	
3.4.1.2.2.3	Non-ontological repositories	Data import	Read data from SQL queries, CSV files, XML data, HTML data, multimedia, and various proprietary formats, specifically Microsoft Office documents and Adobe PDF documents	Critical	NFR	Engineering Components	3.8 in D7.1.2
3.4.1.2.2.3.1	Non-ontological repositories	Data import	Retrieve data from Oracle and MySql relational databases, document repositories, ftp sites or web sites	Critical	NFR	Engineering Components	3.8 / 3.9 in D7.1.2
3.4.1.2.2.3.2	Non-ontological repositories	Data discovery	Crawl folder based systems such as ftp and web sites in order to determine and map true extent of contained resources.	Low	NFR	Engineering Components	3.8 / 3.9 in D7.1.2

Table 25: Database requirements

3.2.5. Standards compliance

NeOn shall be compliant with the current and future Semantic Web standards. Table 26 details the complete set of architecture requirements focused on compliance with standards.

Several types of clients based on a number of technologies shall be integrated with NeOn. As advanced in section 3.1.1.5, the NeOn backend will provide an API-based interface to rich clients and a socket-based interface to thin clients. The API-based interface shall support Java clients like e.g. JSP and also COM clients like e.g. ASP, .NET, and VB.

OWL and Rule languages coexist in the Semantic Web arena. Thus, NeOn, as the next generation ontology framework, will take a dual language approach and support both trends, allowing users to exploit the properties of their respective paradigms conveniently. Additionally, other languages shall be supported as required.

Finally, SPARQL shall be adopted as standard query language.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	External Traceability	Internal Traceability
3.5.1.1	Service access to the NeOn backend	Java-based interface to NeOn backend	The NeOn backend shall provide a Java-based interface to rich clients	Critical	FR	Engineering components	NeOn TA	2.1.1.15
3.5.1.2	Service access to the NeOn backend	Socket-based interface to NeOn backend	The NeOn backend shall provide a socket-based interface to thin clients	Average	FR	Engineering components	NeOn TA	2.1.1.15
3.5.1.3	Support for common ontology query languages	Support for common ontology query languages	Query language standards shall be supported by the NeOn query service	Critical	FR	Engineering components	NeOn R&V	3.2.18
3.5.1.3.1	Support for common ontology query languages	Support for common ontology query languages	SPARQL shall be supported by the NeOn query service	Critical	FR	Infrastructure services	NeOn R&V	3.2.18
3.5.1.4	Dual language approach	Semantic Web Standards	A family of Semantic Web Standards shall be used as formal representation languages	Critical	FR	Infrastructure services	NeOn TA	2.1.1.15
3.5.1.4.1	Dual language approach	OWL support	Several OWL flavours	Average	FR	Infrastructure services	NeOn architecture partners	2.1.1.15
3.5.1.4.2	Dual language approach	Support for rule-based languages, e.g. FLogic, DL	Other languages shall be used	Average	FR	Infrastructure services	NeOn TA	2.1.1.15

Table 26: Standards compliance

3.2.6. Software system attributes

Some characteristics that the NeOn architecture shall meet can be included as non-functional requriements in the specification. These requirements include properties like reliability and security.

3.2.6.1 Reliability

Reliability requirements have focused on system availability, including resource and service availability. Since NeOn is a distributed system, special concern deserves fault tolerance issues. Both in the case of resource and service availability, NeOn shall guarantee normal access to the overall system in case some problem occurs in a determined node. Table 27 shows complete information on reliability requirements.

Req #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	External Traceability	Internal Traceability
3.6.1.1	System availability	System availability	NeOn shall guarantee system availability	Critical	NFR	GUI Components Distributed components Distributed repository	[1]	
3.6.1.1.1	Resource availability	Resource fault tolerance	Regardless of whether any node of the Infrastructure services is temporarily unavailable, NeOn shall guarantee normal access to the rest of the system	Critical	NFR	Distributed repository	[1]	3.4.1.1
3.6.1.2.1	Service availability	Service fault tolerance	NeOn services shall be decoupled from each other to guarantee overall system availability in case a particular service is temporarily disabled	Critical	NFR	GUI Components Distributed components	[1]	3.4.1.1

Table 27: Reliability requirements

3.2.6.2 Security

This section specifies the requirements necessary to prevent malicious access to NeOn data or functionality from occurring in the NeOn framework with the aim to avoid system malfunction and unexpected behaviour. Ontology access rights shall allow to specify what users can access the ontologies and data maintained by NeOn and under what conditions. The analogy with a Unix filesystem applies here, i.e. resource owners shall specify what rights the rest of the users will have on them. Additionally, certain information, both contained in ontological and non ontological requirements, shall be constrained to a reduced group of users.Table 28 shows specific requirements on security for NeOn.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	External Traceability	Internal Traceability
3.6.3.1	Access management to Information resources	Access management to Information resources	Access to information resources shall be administrated	Average	FR	Infrastructure services	NeOn R&V Case studies	3.2.1
3.6.3.1.1	Access management to Information resources	Access management to ontological resources	NeOn shall allow to specify access permissions to ontological resources	Average	FR	Infrastructure services	NeOn R&V Case studies	3.2.1 3.4.1.2.1

3.6.3.1.2	Access management to Information resources	Access management to non-ontological resources	NeOn shall allow to specify access permissions to non-ontological resources	Average	FR	Infrastructure services	NeOn R&V Case studies	3.2.1 3.4.1.2.2

Table 28: Security requirements

3.2.6.3 Maintainability

Modular design is a key factor in order to build a complex system like NeOn, which at the same time allows simple and cheap maintenance in terms of effort. As shown in Table 29, loose coupling is NeOn's bet in this regard. Loose coupling of components and resources shall allow increasing software modularity. Thus, the cost of software update, substitution of old components with new versions, and uptake of external resources will be reduced. It will also be limited to the directly attained components, since interfaces with the rest of the system will be kept in most occasions, allowing connected components to be unaware of such change.

Req. #	Functionality	Description	Importance	Type	Layer of architecture	External Traceability	Internal Traceability
3.6.4.1	Loosely Coupling	The loosely coupled design of the architecture shall help keeping maintenance effort low and focused	Critical	FR	Distributed components	NeOn architecture partners	2.1.1.19
3.6.4.1.1	Loosely Coupling	The loosely coupled design of the architecture shall help keeping maintenance effort low and constrained to the local component	Critical	FR	Distributed components	NeOn architecture partners	2.1.1.19.2
3.6.4.1.2	Loosely Coupling	The loosely coupled design of the architecture shall help keeping maintenance effort low and constrained to the local resource	Critical	FR	Distributed components	NeOn architecture partners	2.1.1.19.1

Table 29: Maintainability requirements

3.2.6.4 User documentation

Complex systems like NeOn need to make a special effort and incorporate high quality documentation and help that allow users to fully grasp all the possibilities offered by the software. Besides, as shown in the user study conducted in [5] it is certainly frustrating for users to find particular problems or bogus system behaviours which the system itself does not explain. Thus, it is fundamental for NeOn to provide users with a complete and sound documentation, including reliable tutorials and examples.

It is also interesting for NeOn developers to rely on standard methods that help simplifying the process of building the software. These methods shall include functional specification and implementation design of each system component using standard design tools like e.g. UML.

Table 30 includes a full description of documentation requirements.

Req. #	Functionality	Requirement Extracted	Description	Importance	Type	Layer of architecture	External Traceability
3.7.1	Documentation	Documentation	Industry-strength documentation shall be written for NeOn components, toolkit, etc.	Critical	NFR	GUI Components	NeOn TA
3.7.1.1	Documentation	Functional specification	A functional specification of each component shall be written	Average	NFR	GUI Components	NeOn TA
3.7.1.2	Documentation	Implementation design	An implementation design of each component shall be written	Average	NFR	GUI Components	NeOn TA
3.7.1.3	Documentation	UML compliance	NeOn design shall be UML compliant	Average	NFR	GUI Components	NeOn TA
3.7.2	Support to the modelling process	Support to the process of modelling ontologies	Samples of ontologies and tutorials for the novice users shall be provided	Average	FR	GUI Components	NeOn TA
3.7.2.1	Support to the modelling process	Accessibility	The help features shall be accessible to the user.	Critical	NFR	GUI Components	NeOn R&V
3.7.2.2	Support to the modelling process	Reliability	The tutorials and the sample ontologies shall be reliable so that the user can trust them	Average	NFR	Engineering components	Case studies
3.7.3	Ontology Documentation	Creation of Documentation	Create documentation that describes t he ontology	Critical	FR	Engineering Components	4.3.1 in D7.1.1 UC13 in D7.4.1

Table 30: Requirements on user documentation

4. Conclusion

In this deliverable we reviewed the previous requirement list presented in [13] In the previous requirement list a set of requirements, based mainly on two documents (Neon Technical Annex and [1] an early requirements list was gathered. Now some of these requirements are not needed anymore (because of the evolution of the research done during the project and the evolution of the Semantic Web research field), others are more important and new other requirements have arisen. The main goal of this review of the previous requirement list was to analyze the evolution of the NeOn toolkit requirements based on the use cases requirements and see how these requirements have evolved during these two years of the project.

The main conclusions we obtained from this analysis were three: first, a new set of requirements that were not present before in this list. Now the use cases have evolved and they need new features that were not present before. Requirements like the improvement of the collaboration tasks for ontology engineering by means of adding new requirements are a very good example of the new needs of the use cases. Second, in this deliverable we deleted the requirements that have been discarded by the uses cases. These requirements were not needed anymore or have been discarded also by the NeOn architecture. One example is the bridge to other technologies. Since KAON2 is the only ontology repository that implements the NeOn metamodel for managing networked ontologies it is the de facto repository for the NeOn toolkit. These requirements have been stated in the WP6 deliverables. The third main conclusion we obtained from this deliverable is the requirements that were present in the first requirements deliverable and are still in the update of it. Requirements like the need of a Web based architecture for the use cases are still a key requirement. Finally, an update of the requirements relevance has been produced. The critical requirements for the use cases are the requirements regarding the visualization issues, the access to the NeOn metamodel, the reuse of their data, ontological and non ontological and the Web based access to the NeOn services.

This document shall provide the guidelines for the NeOn architecture in order to focus in the most important software requirements for the use cases. Also, the evolution of these requirements from the first deliverable to this second one shall provide an overview of the Semantic technologies evolution.

Once this deliverable is finished it will be public available for all partners. Since the requirement list is based in the use cases needs the NeOn technological partners should provide technical solutions for solving these needs. The use cases' partners will have to check that the requirements are fulfilled in order to finish the final software products. The requirements list is a dynamic list which has to be continuously reviewed. There will be some requirements that may not be possible to satisfy. If that happens technological partners should provide alternative proposals for the specific problem.

5. References

[1] Sabou M.R, Gómez-Pérez J.M. et al. NeOn Requirements and Vision, 2006

[2] Lamport L, Shostak R, Pease M. The Byzantine generals' problem. ACM Transactions on Programming Languages and Systems, 4(3):382—401. ACM, July 1982

[3] IEEE-SA Standards Board, IEEE Recommended Practice for Software Requirements Specifications, 1998.

[4] Horrocks I, Parsia B, Schneider P, Hendler J. Semantic Web Architecture: Stack or Two Towers? Principles and Practice of Semantic Web Reasoning (PPSWR), 2005.

[5] Dzbor M, Gómez-Pérez J.M, Buil C. Analysis of user needs, requirements, and behaviours with respect to user interfaces for ontology engineering. NeOn Deliverable D4.1.1, 2006.

[6] Project cBio: Advancing biology and medicine with tools and methodologies for the structured organization of knowledge.

[7] Iglesias M, Caracciolo C. Specification of users and user requirements and detailed use cases. NeOn Deliverable D7.1.1, 2006.

[8] Gómez-Pérez J.M, Pariente T, Daviaud C, Herrero G. Analysis of the pharma domain and requirements. NeOn Deliverable D8.1.1, 2006.

[9] Kiryakov A., Popov B., Terziev I., Manov D. and Ognyanoff D. Semantic annotation, indexing, and retrieval. Journal of Web Semantics, 2, Issue 1, 2005.

[10] Hearst M. A., Untangling Text Data Mining. Proceedings of ACL'99: the 37th Annual Meeting of the Association for Computational Linguistics, 1999

[11] Davies J., Studer R., Warren P. (Eds), Semantic Web Technologies: Trends and Research in Ontology-based Systems. John Wiley & Sons. June 2006.

[12] Waterfeld W., Weiten M., Haase P., Cunnigham H., Dzbor M., Oalma R. and Muñoz O., Specification of NeOn reference architecture and NeOn APIs, NeOn Deliverable D6.2.1, 2007.

[13] Gómez-Pérez J.M, Buil C, Muñoz, O.Requirements on NeOn architecture. NeOn Deliverable D6.1.1, 2006.

[14] Mathieu d'Aquin, Marta Sabou, and Enrico Motta. Modularization: a key for the dynamic selection of relevant knowledge components. In Workshop on Modular Ontologies, 2006