NeOn-project.org

NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D7.4.3 Prototype System for Managing Fisheries Ontology Lifecycle

Deliverable Co-ordinator: Claudio Baldassarre, Peter Haase

Deliverable Co-ordinating Institution: FAO, UKARL

Other Authors: Claudio Baldassarre (FAO), Peter Haase (UKARL)

Document Identifier:	NEON/2009/D7.4.3/v1.0	Date due:	December 31, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	December 31, 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 11 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: Margherita Sini E-mail address: margherita.sini@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

Change Log

Version	Date	Amended by	Changes
0.8	01-01-2007	Claudio Baldassarre	Version 0.8
0.9	10.12.2008	Peter Haase	Architectural aspects, formatting
1.0	17.12.2008	Claudio Baldassarre	Executive Summary, References

Executive Summary

This deliverable represents an **incremental view** of the work previously performed and reported in D7.4.2; it retains the same title, motivations and scope. The report on the steps already taken and the current situation is done in light of the additional knowledge gathered from the project experience over the past months.

The support to the ontology lifecycle management is achieved by serializing the usage of features either embedded in the core implementation of the toolkit[1], or delivered with additional plugins developed by partners within the project. To have a comprehensive outlook on the available functionalities to date, and assess how they cope with the initial requirements exposed for the lifecycle case study, we report on a list of delivered plugins, the required functionalities mapped to them, and more important, what still remains uncovered to complete the overall support.

[1] At the time of writing this deliverable the toolkit version available and used is Ver.1.2.1

Table of Contents

1 Introduction

1.1 Motivation

WP7 major outcome pivots around the realization of two software environments: one will support the creation and maintenance of a pool of fishery domain ontologies, the other will exploit the information in ontological format to feed service-oriented functionalities, aimed at improving document and data retrieval from Fishery division repository in FAO. The former application is known as “Ontology Lifecycle Support System” (T7.4), and the latter is known as “Fishery Stock Depletion Assessment System, (FSDAS).

The main objective of T7.4 is to create and maintain a set of domain ontologies, and to support their continuous internal and cross-related information update. To achieve this objective some critical mechanisms must be in place:

1. creation and maintenance of data models, modularization and versioning;
2. functionality for checking ontology validity, integrity, and soundness;
3. techniques for ontology learning and population.

The NeOn Toolkit provides the framework where these activities can be performed, as well as the basic functionalities to support the work for ontology creation and modification. What the NeOn Toolkit does not provide but is still needed for the success of T7.4, will be provided in the form of application plugins that integrate with the toolkit architecture.

The additional plugins are intended to support the users in activities and tasks which otherwise require either a long time to execute (e.g. ontology population), or huge manual effort (e.g. automatic ontology mapping), or face to face coordination (e.g. editorial work). This deliverable is hence motivated by the need to describe how the NeOn Toolkit will accommodate the plugins (software integration), provided that they satisfy the requirements, and give the view as a whole of a compact and robust software environment for ontology maintainers.

1.2 Relationship with D7.4.2

This deliverable represents an **incremental view** of the work previously performed and reported in D7.4.2; it retains the same title, motivations and scope. The report on the steps already taken and the current situation is done in light of the additional knowledge gathered from the project experience over the past months.

The description of work relative to T7.4 has not changed since the first definition of functional requirements in D7.1.1, neither has its intended meaning. However a work for detailing the initial specifications has generated an updated version (see D7.1.2), and on whose basis we are able to present the “delta” achieved so far.

We started with analyzing the requirements (see 2.1) for the ontology lifecycle support, considering how they evolved in time (D7.1.1 to D7.1.2). After the delivery of D7.4.2, we have hence worked on the development and refinement of the plugins according to that evolution, to meet and to accomplish T7.4 goals.

We found that despite the effort provided by project partners towards software development, the status of the above mentioned components highlights that some have only partially reached their final shape, and others still do not completely meet FAO's specifications.

This deliverable intends to gather an added value around the following main points:

1. the status of each specific plugin identified as the provider of a subset of required functionalities
2. the status of the overall T7.4 in term of the benefits collected and limitations yet to be overcome
3. the steps taken since the first prototype, level of satisfaction, and expectations for the last year of the project.

1.3 Approach for integration

For the integration of software components we follow the architectural approach of the NeOn reference architecture as described in D6.2.1. In this architecture, components are realized as plugins to core NeOn Toolkit, which provides the base set of functionalities along with extension mechanisms. Additional plugins realize particular functionalities to address certain ontology lifecycle activities.

There are three layers in the NeOn Toolkit architecture, as displayed in Figure 1:

1. Infrastructure services: in this layer, basic services and functionalities are provided for most components.
2. Engineering components: this middle layer consists of tightly coupled components and loosely coupled services to provide major ontology engineering functionality.
3. GUI components: this layer is aimed directly at users. Both engineering components and infrastructure services may have GUI components. A set of predefined core GUI components also exists.

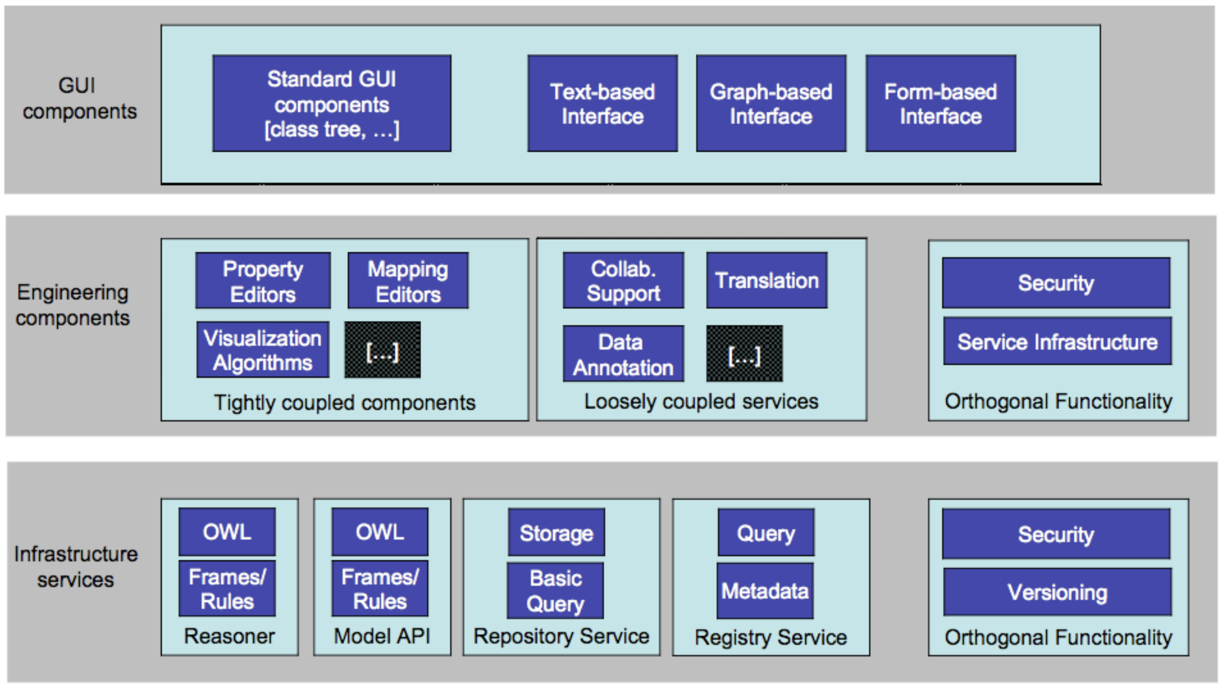

Figure 1: NeOn Architecture

Most plugins integrated as part of this software deliverable are engineering components with GUI components.

2 Requirements analysis

The analysis conducted to realize the second prototype of lifecycle support system has considered the functional requirements as they are described in D7.1.2, and how they evolved compared with the initial attempt in D7.1.1. In the most recent documentation FAO has identified specific categories of requirements with the purpose of exposing a clearer elaboration of the maintenance aspects. The result was intended to drive plugins developers with more substantial indications to focus their effort.

2.1 Analysis outcome

As anticipated in the introduction the main added value in D7.1.2 is the classification of the requirements in to broader categories to associate with a specific plugin responsible to provide them at the best. Unfortunately this approach cannot be easily replicated for each category with respect to all plugins for the following reasons: it is not always possible to map one-to-one category with one plugin; there are plugins which partially share the same purposes, and there are functionalities for which no specific plugin exists but nonetheless can be achieved with some alternative workflows.

Chapter 4 presents extensively the results of verifying which portions of the listed features are covered by its associated plugin(s).

The following table gives an overview of the analysis outcome, compacting a view of the plugins available[2] at the time of writing, and the associated requirement category(ies).

[2] By available we mean plugin that are included in D6.10.2, and hence have passed quality assessment

Plugin Name	Requirement Category[3]
NeOn Toolkit Core	– Ontology implementation – Ontology reuse – Ontology Editing – Quality check – Ontology Visualization – Search
LabelTranslator	– Ontology Editing – Multilinguality
OWLDoc	– Documentation
WorkFlow/Change Management	– Ontology Editing – Workflow – Ontology Visualization
RaDON	– Support to Editing
SPARQL	– Search
Text2Onto	– Support to Ontology Editing – Ontology Visualization
Modules	– Modularization
ODEMapster	– Ontology implementation

Table 1: Plugins associated with the supported lifecycle activities

The analysis is not reduced to only assert associations between plugins and requirements. It also expands in verifying that the features required match with the plugin working behaviour. Hence some tests specific to the scope of ontology lifecycle support have been performed to ensure that this work provides a different value than the review process undertaken in D6.10.2. Chapter 4 gives details on the outcome of this verification phase, as well as on the technical value added by this activity.

IMPORTANT REMINDER: taking into account the constraint on the modelling language expressed by FAO, all the plugins will be guaranteed to work for OWL language otherwise they cannot be considered to be valid regardless of the fact that they perfectly support other ontology formalization languages.

[3] These are the same name used in for the headings in D7.1.2 ANNEX A – Summary of revised requirements for the fisheries ontology lifecycle

3 Prototype analysis

3.1 Introduction

The support for the lifecycle management given from the NeOn Toolkit is achieved by serializing the usage of features either embedded in the core implementation of the toolkit, or delivered with additional plugins developed by partners within the project. To have a comprehensive outlook on the available functionalities to date, and assess how they cope with the initial requirements exposed for the lifecycle case study, we report in this section on the list of delivered plugins, the required functionalities mapped to them, and more important, what still remains uncovered to complete the overall support.

The aim of providing a resume on the actual status is to have a pinpoint view on the reasons why the development has not completely satisfied the requirements, and an idea of the future commitment of partner developers to continue with their work. It should also at the same time provide a punctual feedback to drive the next plugin releases.

In contrast with the concurrent job of plugin quality assessment, the perspective we take here is to verify if the effort produced by developers, meets in fact the guidelines provided with the requirements shown initially in D711, and finally revised in D712.

In other words while a normal plugin review should answer generic questions about:

- plugin integration with the NeOn Toolkit,
- usability,
- documentation support,
- bug free etc

here we try to consider more the point of view of:

- How are specific requirement addressed with one of the provided plugin's features?

3.2 Plugins verification

This section is dedicated to detailing the verification phase which was undertaken to assess the status of the ontology maintenance support. We also aim to draw conclusions that will drive the development of the plugins for developments in the project in the coming year.

The approach adopted was to test the plugins and fill in a template like the one in Table 2

Plugin name	Plugin name

Purpose of the plugin	Reports about the general purpose of the plugin, and how it applies in the scope of T4.3
Functionalities that cover the requirements	Reports about the functionalities tested and covering the requirements expressed in D712
Requirements not yet covered by the plugin	Reports about the functionalities tested but NOT covering the requirements expressed in D712
Reasons for uncovered functionalities	*May be either:* Technical problems Development schedule The Requirement(s) needs further specifications (developers provide details)
Future plans	A description of development plans with respect to the missing functionalities during the upcoming year of the project

Table 2: Template to report the result of plugins assessment

3.2.1 LabelTranslator

3.2.1.1 Purpose of the plugin

The plugin itself proposes to address the multilinguality requirement in ontologies that is now demanded by institutions worldwide most of whom have a vast number of resources available in many different languages. To tackle this requirement LabelTranslator automatically localizes ontologies; takes as input an ontology whose labels are described in a source natural language and obtains the most probable translation into a target natural language of each ontology label. For more detail refer to wiki page[4]

In the scope of the lifecycle management of Fishery ontologies, this plugin supports the ontology engineers to be able to add a new language to an ontology that does not yet contain this feature; in other words localize ontology with respect to a language means to register the human readable information into one or more languages. LabelTranslator supports this activity by mean of a component for assisted term translation from/to a set of languages (see wiki page), and serialize this information according to a linguistic model (LIR), where the lexical information can be accommodated for a future programmatic processing.

3.2.1.2 Provided functionalities that cover the requirements

Adding a new language [D7.1.2 cf. 2.4.2.5]**:** ontology engineers shall be able to add a new language to a monolingual or already multilingual ontology

Selection of languages [D7.1.2 cf. 2.5.9.31.a]**:** select at least two languages (or more, if required), one in view mode, the other in editing mode

Edit multilingual labels [D7.1.2 cf. 2.5.9.31.b]**:** add/edit/delete multilingual labels to individual concepts;

Creation and management of annotations [D7.1.2 cf. 2.5.1.13], multilingual textual annotation will be supported. Some examples are: “scope notes” (as commonly used in thesauri), free-text comments.

3.2.1.3 Requirements not yet covered by the plugin

The original requirements about annotating an ontological resource consider also using multimedia/images to associate to the elements [D7.1.2 cf. 2.5.1.13]; at this stage the LIR model does not allow to express that a binary object is representative of an element or a URL to such a resource

3.2.1.4 Reasons for uncovered functionalities

First, in the requirement specification for multilinguality [D7.1.2 cf. 2.5.9] the need to associate multimedia information to ontology elements is not mentioned. The second reason is a technical motivation: the multimedia information is not directly related with the multilingual information that supports the LIR model.

3.2.1.5 Future plans

Label Translator developers will perform a detailed analysis of the approaches for associating multimedia information at the meta-data level of a Knowledge Based Representation. Then, we will implement the support for this functionality in the next release of the LabelTranslator plugin.

[4] http://www.neon-toolkit.org/wiki/index.php/LabelTranslator

3.2.2 OWLDoc

3.2.2.1 Purpose of the plugin

The OWLDoc plugin adds to the NeOn Toolkit an option to export an OWL-DL ontology as an HTML Documentation. This plugin extracts information from the OWL Ontology and creates an output that contains an organized set of HTML files that provide the documentation about the ontology and all its resources. For more details refer to wiki page[5]

In the scope of lifecycle management of Fishery ontologies, this plugin supports non-ontologist users to access the ontology model content, without forcing on them too many details of a cumbersome graphic interface. Instead they can benefit from a browseable-like documentation in the style of javadoc.

3.2.2.2 Provided functionalities that cover the requirements

Creation of documentation [D7.1.2 cf. 2.4.3.6]**:** The system shall support ontology engineers in the creation of documentation for ontologies, in particular concerning ontology design. Documentation is necessary for both ontology engineers and ontology editors, therefore it shall be possible to use the most appropriate form of documentation for each group.

3.2.2.3 Requirements not yet covered by the plugin

The complete requirement description for documentation creation [D7.1.2 cf. 2.4.2.6] also reports the needs to have an UML-like diagram representing an ontology project as part of the documentation. This last part of the requirements is actually not yet fulfilled by OWLDoc.

3.2.2.4 Reasons for uncovered functionalities

The scope of OWLDoc was intentionally restricted to Javadoc-like documentation. The generation of UML-like diagrams is covered by other plugins, in particular the OntoModel[6] plugin for UML-based ontology modelling.

3.2.2.5 Future plans

No plans of further development are scheduled with respect to the requirement to be covered.

[5] http://www.neon-toolkit.org/wiki/index.php/OWLDoc

[6] http://www.neon-toolkit.org/component/option,com_wrapper/Itemid,128/

3.2.3 Workflow/Change management

3.2.3.1 Purpose of the plugin

The main purpose of the Workflow/Change Management plugin is to track the ontology changes and then manipulate these changes according to the role of a user, where the user could log on / out or register in a preference page. For more detail refer to wiki page[7]

In the scope of lifecycle management of Fishery ontologies, this plugin supports the main activity of monitoring the modifications applied to an ontology model and providing the possibilities of instantiating the workflow designed in [D7.1.2 cf. 2.3].This will allow for a collaboration between various users with different roles and expertise, aiming to keep the data and the model up-to-date.

3.2.3.2 Provided functionalities that cover the requirements

[D7.1.2 cf. 2.5.6] Provide a simple, intuitive interface for each of the user roles in the editorial workflow so that tasks are performed efficiently and effectively. Four specific views are required, based on the user roles and element status:

- **Draft** view: approved information plus changes made by the current editor, with the difference between the two states clearly visible. This view should be available only to the subject expert who made the draft.
- **To be approved** view: approved information plus all the pending elements to be approved by validators, with the difference between the two states clearly visible. This view is for validators.
- **Approved** view: approved information only. For all users, including viewers.
- **To be deleted** view: approved information plus elements that are assigned “to be deleted”, with the difference between the two states clearly visible. This view is for validators and for the editor who made the draft.

Creation and management of metadata [D7.1.2 cf. 2.5.1.12]**:** Metadata is essential to a number of activities, including collaborative editing. The editing environment shall automatically attach and manage the following pieces of metadata:

- date of creation/editing of the element and author (automatically);
- information about author/editors of the ontology elements;
- history of changes.

Visualization of metadata [D7.1.2 cf. 2.5.5.25]: During the process of updating and validating the ontology, it is important that a number of pieces of information be highlighted to the user. In particular: editing history of the ontology element, including its authors and provenance, summary statistics.

3.2.3.3 Requirements not yet covered by the plugin

The complete requirement description for documentation creation and management of metadata [D7.1.2 cf. 2.5.1.12], also specify the support for summary statistics on the ontology; examples of statistic are: editorial actions per ontological element, or during a time period, or per editor etc.

[7] http://www.neon-toolkit.org/wiki/index.php/Workflow_Support

This is not yet provided by the Workflow/Change plugin.

3.2.3.4 Reasons for uncovered functionalities

The required information, apart from the number of classes, properties, individuals and axioms, and about the number of mappings/relations is not the target of Workflow/Change management components.

3.2.3.5 Future plans

No plans to embed the required support in the Workflow/Change management plugin are envisaged, although developers are willing to include some support as part of the development of another plugin component (i.e. Oyster[8]), led by them.

[8] http://www.neon-toolkit.org/component/option,com_wrapper/Itemid,128/

3.2.4 RaDON

3.2.4.1 Purpose of the plugin

The purpose of the RaDON plug-ins is to deal with inconsistency and incoherence occurring in networked ontologies. Specifically, RaDON provides two plugins to deal with a single ontology or an ontology network; for a single ontology the user can perform: incoherence checks and inconsistency checks automatically repair and manually repair. Similar functionalities are featured for an ontology network; the main difference is that they apply to a mapping between two ontologies by assuming the two source ontologies are more reliable than the mapping itself. For more extensive detail refer to the RaDON wiki page[9].

In the scope of the lifecycle management of Fishery ontologies, this plugin assists the user in assessing if the modification s/he is performing is causing potential inconsistency for the ontology.

3.2.4.2 Provided functionalities that cover the requirements

Preview and check consistency **of the newly added elements** [D7.1.2 cf. 2.5.3.19]**:** Before final inclusion in the ontology, it will be possible to visualize the ontology including the newly added elements, and check it for consistency.

3.2.4.3 Requirements not yet covered by the plugin

None.

3.2.4.4 Reasons for uncovered functionalities

Not applicable.

3.2.4.5 Future plans

RaDON will be further developed to handle various forms of ontology networks, instead of only dealing with single ontologies. These capabilities will be particularly relevant for building the next versions of the fishery ontologies.

[9] http://www.neon-toolkit.org/wiki/index.php/RaDON

3.2.5 SPARQL

3.2.5.1 Purpose of the plugin

The SPARQL Plugin allows a user to do conjunctive queries. It receives an OWL ontology and a query expressed using SPARQL syntax as inputs and outputs the answers in a table; more details can be found on the wiki page[10].

In the scope of the lifecycle management of Fishery ontologies, this plugin supports advanced searching feature inside the ontology; the subject for these kinds of searches can be constructed to retrieve elements related by a direct/indirect simple/complex relationship.

3.2.5.2 Provided functionalities that cover the requirements

Structural search [D7.1.2 cf. 2.5.8.30a]**:** It is useful, especially to validators and ontology engineers, to be able to perform searches that exploit structural aspects of the ontology. For example, to identify instances of classes with a common ancestor, to select relations with a given domain and/or range, or to find instances with one or more given properties.

3.2.5.3 Requirements not yet covered by the plugin

Although there is no specific bullet point in D712, that addresses the need to have an interface to the SPARQL syntax, it is assumed by the context, and by the definition of non-ontologist users, that a low level syntax detail should be hid by some form of a more intuitive interaction form. This feature for SPARQL plugin is to date missing.

3.2.5.4 Reasons for uncovered functionalities

SPARQL is not really adequate for an end user oriented query interface, but more suitable for application developers and ontology experts. As such, the SPARQL plugin is not suitable to address this requirement. However, other plugins are being developed for this purpose.

3.2.5.5 Future plans

Current developments include plugins to translate end user queries into structured SPARQL queries. Specifically, developers are working on two plugins:

- ORAKEL: translation of natural language queries into SPARQL
- XXPlore: translation of keyword queries into SPARQL

These plugins are planned to be available by M36 of the project.

[10] http://www.neon-toolkit.org/wiki/index.php/SPARQL

3.2.6 Text2Onto

3.2.6.1 Purpose of the plugin

Text2Onto is an ontology learning framework which has been developed to support the acquisition of ontologies from textual documents; it provides an extensible set of methods for learning new concepts, build a hierarchy and instantiate object properties. The wiki page [11] offers more details about its functioning features.

In the scope of the lifecycle management of Fishery ontologies, this plugin supports the ontology engineers with a fast method for populating the model schemas with instances learned from documental resources that are frequently published about Fishery domain. Engineers might also exploit assistance in the expansion of the conceptual model.

3.2.6.2 Provided functionalities that cover the requirements

Suggest candidate elements [D7.1.2 cf. 2.5.3.17]: On the basis of given textual corpora, the tool shall provide the author with a list of candidate elements suitable for inclusion in the ontology.

Visualization of candidate elements for inclusion in an ontology [D7.1.2 cf. 2.5.5.27]**:** the requirement concerning the editorial support to ontology population shall be provided with an adequate visualization and interface in order to allow ontology editors to select, inspect, approve and include candidate elements in the ontology.

3.2.6.3 Requirements not yet covered by the plugin

The complete requirement description [D7.1.2 cf. 2.5.3.17] about suggesting the users with a candidate elements, also reports about the need: (i) that ontology editors may use support for the creation of instances or relations between instances; (ii) to have property instantiation between individual belonging to different ontologies; (iii) to give the user a way to inspect and select the appropriate candidate [D7.1.2 cf 2.5.3.17].

Support candidate selection [D7.1.2 cf. 2.5.3.18]**:** Editors shall have facilities to inspect and select the candidates suggested by the system. In particular:

- The tool shall show the documents and excerpts supporting the extracted terminology, including document metadata such as title of the document, author, data owner, publication date.

3.2.6.4 Reasons for uncovered functionalities

The acquisition of property assertions –also across multiple ontologies– will be made available as a NeOn Toolkit plugin in early 2009. It will build upon the RELExO plugin which focuses on taxonomy refinement and integration and will be made available as part of D3.8.2[12] (M36).

The explanation facilities of Text2Onto have not been ported to the plugin version and will not be extended in the suggested way, because there is no dedicated developer for Text2Onto right now.

[11] http://www.neon-toolkit.org/wiki/index.php/Text2Onto

[12] Evaluation of methods for contextualized learning of networked ontologies

3.2.6.5 Future plans

In the upcoming months, developers will concentrate on more ontology learning plugins including LeDA, RELExO and RoLExO.

3.2.7 Modules

3.2.7.1 Purpose of the plugin

The Modules plugin provides the functionality to extract modules and allows the manipulation and combination of modules; more specific details on the features provided through this plugin are listed in the dedicated wiki page[13]

In the scope of the lifecycle management of Fishery ontologies, this plugin supports the users with a simplified way to approach ontology schema which are large for some characteristics: e.g. they contain elements classified according to several system codifications, or contain localization in several languages, or yet contain knowledge about diverse domains.

3.2.7.2 Provided functionalities that cover the requirements

[D7.1.2 cf 2.4.4.7] Modules may be manually created by ontology engineers and editors/validators, or created by means of (semi)automatic methods (e.g. entire branches of a hierarchy, subclass skeleton).

3.2.7.3 Requirements not yet covered by the plugin

Some examples of modules found relevant to WP7 use case [D7.1.2 cf. 2.4.4.8]**:** Mechanisms shall be in place to allow ontology engineers to create at least the following (types of) modules:

- **modules by “topic”:** In case of large ontologies, covering more than one domain (e.g. AGROVOC, ASFA) it is useful to be able to select modules on the basis of the domain covered, such as “fisheries”, “aquaculture”, “pests”, etc.
- **modules by language:** Multilingual ontologies may be available in several languages (e.g., fisheries ontologies described in [D7.2.2] contain three or four languages, AGROVOC sixteen) although for common applications not all of them are used. It is then useful to select only the languages used for the specific application at hand.
- **modules by code:** Most fisheries ontologies include one or more classification systems, although not all of them are used at the same time. It is useful to be able to define, select, visualize and utilize only the module of the ontology where the desired classification system is used. Examples of classification systems are: ISO2 [ISO2] and ISO3 [ISO3], ISSCAAP code [ISSCAAP00], the ISSCFV [ISSCFV], ISSCFG [ISSCFG].
- **modules for editorial duties:** Another module specification is related to editorial work. Especially when dealing with large ontologies, it is important to be able to select the ontology elements that are involved in the editorial work of a given editor. This is to avoid editors being given the entire ontology, when smaller, more manageable and focused parts of the ontology can be defined and extracted.

3.2.7.4 Reasons for uncovered functionalities

The modularization plugin is meant to be able to handle the large variety of tasks related to modular ontologies and ontology modularization. It provides a general and flexible framework

[13] http://www.neon-toolkit.org/wiki/index.php/Modules

for these tasks. As such the main reason for most of the "missing mechanisms" above is that these are specific to the FAO case study and would require ad-hoc mechanisms.

- **Modules by topic:** The partitioning plugin is not specific to this purpose and can eventually provide only with weak support. The requirement would have to be specified; a specific process is required to identify elements related to what is called a "domain" or a "topic"
- **Modules by language:** this cannot be done with the current tool as it falls outside the scope of modularization methods in general. An ad-hoc tool to realize this is the quickest solution.
- **Modules by code:** This is very specific to the FAO case study, and is not a modularization scenario. This can eventually be treated very easily using a query engine.
- **Modules for editorial duties:** the problem is unclear as well as a possible applicable solution.

3.2.7.5 Future plans

Several complete modularization use cases will be studied as part of the modularization methodological guideline in WP5 (deliverable D5.4.2[14] M36). This will include scenarios from FAO if appropriate and sufficiently specified.

[14] Revised and extended techniques for contextual visualization of ontologies and ontology networks

3.2.8 ODEMapster

3.2.8.1 Purpose of the plugin

ODEMapster is a processor in charge of carrying out the exploitation of the mappings that a user can draw between ontology elements and some data base elements, with the aim of populating the ontological model with instances coming from the relational repository. The language used to serialize the mapping is called R2O and the actual release of this plugin only supports a subset of it; this release is compatible with OWL/RDF(S) and MySQL type databases. More specific details on the features provided through this plugin are listed in the dedicated wiki page[15]

In the scope of the lifecycle management of Fishery ontologies, this plugin supports the users with an easy way to migrate data contained in a relational data base, thus to populate the ontological schema. Its use is particularly envisaged in scenarios where a massive quantity of data needs to be moved into the ontological schema.

3.2.8.2 Provided functionalities that cover the requirements

Import data from databases [D7.1.2 cf 2.4.1.2.a]: taxonomies, classification schemas and thesauri are commonly stored in relational databases, it shall be possible to connect to a RDBMS, view (and/or import) the relevant tables (logical structure and content) and import the data according to the defined ontological model. Deliverable D7.2.2 [D7.2.2] reports on conversion work of this type.

3.2.8.3 Requirements not yet covered by the plugin

n/a

3.2.8.4 Reasons for uncovered functionalities

n/a

3.2.8.5 Future plans

To avoid the "pre-processing" of the RTMS tables. It is currently not possible to upgrade directly the RTMS database content to ontologies, using ODEMapster. Due to the RTMS structure, it is necessary to pre-process the RTMS tables, before using ODEMapster. Therefore, we will improve the ODEMapster process to take into account the RTMS original tables, avoiding such pre-processing.

[15] http://www.neon-toolkit.org/component/option,com_wrapper/Itemid,128/

3.2.9 Core NeOn Toolkit

3.2.9.1 Purpose of the plugin

The core NeOn Toolkit represents a set of plugin components that belong to the core implementation of the toolkit. They provide the basic functionalities proper of the ontology editor, and come bundled in the binary version that a new user can download from the NeOn Toolkit site[16].

In the scope of the lifecycle management of Fishery ontologies, this plugin-set supports all the functionalities that envisage the creation/modification/deletion of ontology elements, as well as importing/exporting the model from/to different formats, providing the graphical interface to the mentioned features.

3.2.9.2 Provided functionalities that cover the requirements

Support of ontology implementation [D7.1.2 cf. 2.4.1.1]**:** The ontology engineer shall be able to create new ontologies and elements in them

Ontology reuse, reengineering, integration [D7.1.2 cf. 2.4.1.3]: New ontologies may be created on the basis of existing ones, either by transforming the conceptual model of an existing and implemented ontology into a new one (reengineering) or by including the existing ontology into the new on (integration). The ontology engineer shall be able to open and visualize any ontology (at least with view rights) and use it as a basis to create a new one. This implies that the engineer be able to select and copy any ontology element and paste it into the ontology being created (edited). The engineer shall also able to create mappings between the two ontologies.

Edit (single and multiple) ontologies [D7.1.2 cf. 2.4.2.4]: Granted the appropriate rights, ontology engineers shall be able to edit the necessary ontology elements, including mappings between ontologies and relations across ontologies.

Editing ontology elements [D7.1.2 cf. 2.5.1.9]**:** The editing environment shall allow editors to edit one or more ontologies at a time (assuming the appropriate editing rights are granted). Depending on the specific ontologies at hand, editorial duties may be more focused on specific ontology elements (instances, classes, properties, relations).

Compare ontologies [D7.1.2 cf. 2.5.2.15]**:** In order to compare two ontologies, ontology editors need support from the system. This support shall be both visual and by means of statistics (that could also be visually shown side by side) about the two ontologies to compare

Visualization of single ontologies [D7.1.2 cf. 2.5.5.23]**:** Ontology editors shall be able to visualize and browse single ontologies

[16] http://www.neon-toolkit.org/component/option,com_remository/Itemid,103/

Print visualization ontologies [D7.1.2 cf. 2.5.5.28]**:** It shall be possible to print out the chosen visualization(s).

Search [D7.1.2 cf. 2.5.8.30]**:** users shall be able to search within ontologies and legacy systems; search shall be possible on the ontologies being visualized.

Textual search [D7.1.2 cf. 2.5.8.30b]**:** It is useful to be able to search for text across ontologies, independently of where the text appears (labels, properties, annotations, etc.).

3.2.9.3 Requirements not yet covered by the plugin

The complete requirement description [D7.1.2 cf. 2.5.8.30] about searching the ontologies, mentions the possibility to also explorer model in some repository different from the actually loaded ontology collection in the workspace; to date this is not yet delivered.

Thesauri [D7.1.2 cf. 2.4.1.2.c]: Mechanisms shall be in place to support (semi)automatic conversion of thesauri into ontologies (RDFS, OWL).

Runtime access to databases [D7.1.2 cf. 2.4.1.2.b] In many cases it is advisable to keep the data in the relational database and access it through an ontological layer (ontology). The system shall then support ontology engineers in defining the appropriate ontology, mapping it onto the database and accessing the data (i.e., without physically export it from the database). Facilities shall be provided to enable ontology engineers in “adding” and exploiting relations and mappings not present in the database.

Check for “similar concepts” [D7.1.2 cf. 2.5.2.14.b] A simple case of similarity that shall be taken into account applies when two instances appear identical to humans but are not identical to a machine (for example in case of spelling mistakes).

Summary statistics [D7.1.2 cf. 2.5.2.14.b]: a number of summary statistics are useful to control the development of ontology, including:

- depth of the ontology,
- number of ontology elements (classes, instances per class, relations, properties),
- number of mappings and relations between external ontologies,
- distribution of subclasses per top level classes

Export ontologies into other formats for backward compatibility [D7.1.2 cf. 2.5.7.29]: It shall be possible to export ontologies in several output formats in order to facilitate data exchange with legacy systems and uniformity with existing FAO resources. In particular, it shall be possible to export ontologies converting the schema (and included instances) according to relational database design principles, or as SKOS ontologies [SKOS] .

3.2.9.4 Reasons for uncovered functionalities

Export to syntaxes like SKOS or schemata like relational database: first a mapping must be agreed and defined with final users on what properties are skos: broaderTerm for example, or skos: narrowerTerm, or skos: relatedTerm etc. The export to RDBMS schema is spurious since there are so many degrees of freedom.

Thesauri: as for the export functionality this also needs an initial agreement and definition of some form of correspondences between the source and target model format.

Explore models in some external repositories: this requirement needs more specifications on what repositories should be explored, for what purpose. Actually Watson and Oyster support retrieving ontologies based on a search metaphor; these requirements also needs more specifications about other functionalities than what has been delivered with the present NeOn Toolkit plugins.

Runtime access to databases: this is implemented only for F-logic.

Check for "similar concepts" (for example in case of spelling mistakes): Current implementation of a search mechanism would need to be drastically changed.

Summary statistics: the coverage is partial but the work to improve on satisfying this requirement is ongoing

3.2.9.5 Future plans

SKOS export/import will be achieved in 2009

There are no plans for development of runtime access to databases for OWL by Ontoprise

Statistics on ontology is work in progress until May 2009

Enhanced search functionalities are in place on the development agenda for last year of the project.

3.3 Functionalities achievable with alternative workflows

Some of the requirements in D7.1.2 do not have a unique specific plugin devoted to address them; nonetheless users can still resolve their problems if they are prepared to reframe them in a perspective adjusted to the tool they have handy. In the following we provide a list of examples of requirements equivalently satisfied by the NeOn Toolkit support to lifecycle management.

Check for duplicated elements [D7.1.2 cf. 2.5.2.14.a] such as instances having exactly the same pieces of information (e.g., labels, properties values).

A possible solution is to identify duplicates in two ontologies by using one of the features of the Modules plugin.

Visualize overlapping portions of ontologies [D7.1.2 cf. 2.5.5.24.b]: editors should be able to visualize the overlapping between ontologies, i.e., ontology elements present in both ontologies. A clear visualization of overlapping between ontologies is useful as a support to mapping creation, and during editorial workflow.

A possible solution to achieve the visualization is to use one of the features of the Modules plugin

Visualize mapping between ontologies pair [D7.1.2 cf. 2.5.5.24.a]: editors should be able to visualize ontology elements from ontologies plus mappings and relations between them.

A possible solution to achieve this goal is to create and save a third schema that will only contain the mappings between elements of two OWL ontologies.

3.4 Current status of the ontology maintenance support

The goal of this chapter was to identify the development status of the features delivered with the plugins (restricted to OWL language) with respect to the functional requirements exposed in the D7.1.2. This would then give an idea of the amount of work that still needs to be done to cover functionalities required for the actual prototype, and also what amount of effort needs to be dedicated to refine and add advanced functionalities in the scope of supporting the lifecycle of Fishery ontologies.

Summarized in the table below there is a compact view of the plugin assessment outcome; particularly interesting is the status of development with respect to FAO requirements.

- **Plugin name:** the name of current plugin being reported.
- **Requirement category:** the name of one or more categories that the plugin is serving with its features.
- **Reached final stage of development:** a boolean value to synthetically say if this plugin will be further developed with respect to the features required for lifecycle. This evaluation is based on the feedback by the partners about their future developments plans.

Plugin Name	Requirement Category[17]	Reached final stage of development
LabelTranslator	– Ontology Editing – Multilinguality	Not yet
OWLDoc	– Documentation	Yes
WorkFlow/Change Management	– Editing – Workflow – Visualization	Not yet
RaDON	– Support to Editing	Not Yet
SPARQL	– Search	Yes
Text2Onto	– Support to Editing – Visualization	Yes
Modules	– Modularization	Not yet
ODEMapster	– Ontology implementation	Not yet
NeOn Core	– Ontology implementation – Ontology reuse – Editing – Quality check – Visualization – Search	Not yet

Table 3: Compact view of the plugin assessment outcome.

The result shows that 6 plugins reached their final stage of development. It is important to notice that some responsible developers do not consider other efforts to be applied for the components. When this happens it is not equivalent to considering that the identified feature set is entirely satisfied. Two main causes why this is due are: (i) the required feature does not apply to the plugin, (ii) the required feature is part of a set addressed by plugin under development. By eliminating such events some feature sets are reduced and finally considered entirely covered.

Nontheless those features that remains from this migration process, are collected and reclassified both per plugin (whether this exists), and by criticality to the success of T7.4 success. The criticality parameter assumes on of these values low/average/high.

Here in the following we list the mentioned features grouped according to criticality value; the same list can be found in a more schematic format in ANNEX A

Under the criteria ‘criticality for T7.4 objectives success’, we consider the following missing features as **Low**:

[17] These are the same name used in for the headings in D7.1.2 ANNEX A – Summary of revised requirements for the fisheries ontology lifecycle

- Using multimedia file type (e.g. images, audio/video file) to annotate ontological resource
- To have UML-like diagrams associated with the documentation of an ontology project
- To have the ability to extend the search for ontological entities in other repositories other than the ones loaded in the current working space
- To identify similar concepts based on possible misspelling errors
- To have a matrix reporting on structural information about the ontology such as number of classes, property, individual, max width/breadth etc.

Under the criteria 'criticality for T7.4 objectives success', we consider the following missing features as **Medium**:

- To have statistics about the editorial workflow; examples of statistic are: editorial actions per ontological element, or during a time period, or per editor etc.
- To have a graphic interface that supports users not confident with SPARQL syntax, to build a query using elements from the ontology tab; a possible interaction can be drag and dropping concept/property/instance into an area of the window dedicated to compose the subject/predicate/object item of the query
- To have the possibility to export/import an ontology into SKOS or relational data model

Under the criteria ''criticality for T7.4 objectives success', we consider the following missing features as **High**:

- To have in place a mechanism that enables the users to instantiate relationships among individuals in the ontology, and between individuals of different ontologies, when using textual resource processing to gather new knowledge. Moreover the users need to be supported to select the appropriate candidate entity from the list of result extracted from the text corpus, for example displaying the text excerpt related with the element on focus
- To be supported for creating ontology modules responding to specific contexts: domain topic, language, standard code, editorial duties
- To have the access to entities into a relational database without necessarily having to migrate the data into an ontology project, but instantiating links between the ontology model and the DB model

The information we collected from partner developers (see ANNEX A) about the amount of effort they plan to dedicate in response to the requirements not yet covered, can help to draw a conclusive outline of a development expectation:

Out of the total number of functionalities considered highly critical, there are 75% [18] of them for which we received a positive comment from partners willing to commit to tackling the missing points.

- the requirements about modularization will be satisfied, provided that more details on the dynamic of modules definition are submitted. There are no plans to provide support for OWL ontologies to be connected to DB at runtime for operation like read/write from/to DB schema.

Out of the total number of functionalities considered averagely critical, there are 66%[18] of them for which we received a positive comment from partners willing to commit to tackling the missing points

- SPARQL will have a GUI for building the queries (ORAKEL and XXPlore plugin); SKOS export/import will be satisfied provided that an agreement on the SKOS vocabulary is reached between end-users and developers.

Out of the total number of functionalities considered lowly critical, there are 80%[18] of them for which we received a positive comment from partners willing to commit to tackling the missing points.

- enabling LabelTranslator to annotate resource with multimedia file; support search in ontology repositories, provided that more specifications to the requirement are submitted to developers; enhancement of the search functionality, and finally support enrichment of meta information about the ontology (e.g. number of classes, individual, width, breadth etc.).

To conclude, there is a total of 65%[19] of commitment to tackle missing points on behalf of partner developers; for some of them this commitment is agreed providing they have more specifications of the requirement.

[18] This percentage is calculated considering the total number of missing functionalities per criticality -as reported in ANNEX A- and the number of them for which a positive comment on future development commitment has been provided by partners.

[19] This percentage is calculated considering the total number of missing functionalities -as reported in ANNEX A- and the number of them for which a positive comment on future development commitment has been provided by partners.

4 Conclusions

Further steps have been taken since the delivery of the first software prototype with respect to the development and integration work. In the following paragraphs we adopt two main perspectives to report on the evaluation of the enhancements achieved so far: (i) the commitment to deliver the planned plugins, and (ii) the achieved refinement of the features already provided. We also express our degree of satisfaction for the work completed so far and our expectations for the fourth year of the project.

The chapter about the next steps in D7.4.2 reports on the envisaged work to be carried out during the upcoming 10 months of the project; it describes the need to have at least 3 important plugins, supporting likewise crucial activities:

- Editorial workflow support
- Modularization
- Automatic ontology mapping

We have already shown that this software prototype indeed includes the first 2 components. The support for automatic ontology mapping has also improved towards a stable release, but for a minor interface defect, it was mostly impossible to assess it as we did for the other components. We consider this a positive result because it is the materialization of the planning work done in view of supporting this and other tasks in WP7.

D7.4.2 presented several limitations in terms of supporting the users with specific activities; over the months those limitations had been identified, categorized and prioritized to facilitate the realization of the second prototype. At this stage we can consider that the biggest portion of the limitations have been addressed (witnessed by the growth number of plugins), though not completely overtaken. This means that the behaviour experienced when using the plugin is not definitively in line with what a user would expect. We see a closer interaction between developers and FAO users as the means to enable a more detailed and fine grain assessment by feeding back specific comments on the functionalities trials.

Some examples of plugins that provided an improvement over the months are: OWLDoc and ODEMapster, together with the support for OWL offered by the toolkit. A plugin which newly entered this software release is the RaDON component, now dedicated to assess some aspects of correctness of an ontology model.

With respect to the work result delivered, we consider ourselves satisfied but we are also very aware that there are still many components in need of fine tuning to be considered completely acceptable to FAO needs. D7.4.3 can be assumed as a considerable improvement if compared to D7.4.2, and we expect that more can still be achieved if developers work in a closer relationship with FAO end-users. We are also confident of an imminent step ahead when some of the functionalities, now only supporting the F-Logic language, will also be extended to support OWL language.

5 Next Steps

At the end of the previous chapter we presented a rough statistical view showing what we expect will be the commitment of partner developers for the next year of the project with respect to plugins delivery. The value of 65% indicates that out of the total number of functionalities still to be satisfied, there is a high number which is planned to be tackled by plugins providers, and hence a strong probability that the T7.4 can succeed in its objectives.

In the coming months we predict that partners will keep the same commitment, while FAO provides further support when it comes to specify some of the requirements that are not yet clear. For a fine tuning of the plugins behaviour FAO also expects to forge a closer collaboration with plugin developers.

The suggestion, on how to proceed in prioritizing the development of new features for the lifecycle support, must be driven by the identified categories of criticality as reported in ANNEX A.

From our investigation with partners developers, we are already able to list two planned next steps for upcoming project time frame:

- Developments of functionalities so far only available on the F-Logic side (including e.g. mapping support)
- End-user oriented plugins, e.g. to enable search and querying for non-ontology experts.

6 References

[D7.1.1] FSDAS Requirements, http://www.neon-project.org/ACollab/get_file.php?id=475

[D7.1.2] FSDAS Revised Requirements, http://www.neon-project.org/web-content/images/Publications/neon_2008_d7.1.2.pdf

[D7.2.2] Revised and Enhanced Fisheries Ontologies, http://www.neon-project.org/web-content/images/Publications/neon_2007_d7.2.2.pdf

[D7.4.2] Prototype System for Managing Fisheries Ontology Lifecycle, http://www.neon-project.org/web-content/images/Publications/neon_2008_d7.4.2.pdf

[D6.2.1] Specification of NeOn reference architecture and NeOn APIs, http://www.neon-project.org/ACollab/drafting/revisions.php?id=489

[D6.10.2] Updated NeOn Toolkit plugins

[ISO2] International Standard Organization (ISO): ISO 3166 ALPHA-2, 2006.

[ISO3] International Standard Organization (ISO): ISO 3166 ALPHA-3, 2006.

[ISSCAAP00] FAO. International Standard Statistical Classification of Aquatic Animals and Plants (ISSCAAP). Version in use from 2000 available at: ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/AnnexS2listISSCAAP2000.pdf

[ISSCFV] International Standard Statistical Classification of Fishery Vessels (ISSCFV) by Vessel Types, in use until 1995. 1984. ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/annexLII.pdf

[ISSCFG] International Standard Statistical Classification of Fishing Gear (ISSCFG) ftp://ftp.fao.org/FI/DOCUMENT/cwp/handbook/annex/AnnexM1fishinggear.pdf

7 ANNEX A – Features clustered per degree of criticality

Criticality	Feature	Plans for future development	Plugin
Low	Using multimedia file type (e.g. images, audio/video file) to annotate ontological resource	There will be an analysis of possible approaches and an implementation to support annotation with multimedia file	LabelTranslator
	To have UML-like diagrams associated with the documentation of an ontology project	No further development	OWLDoc
	To have be able to extend the search for ontological entities also to ontology in other repositories than the ones loaded in the current working space.	Requirement needs further specification	NeOn Core
	To spot out similar concepts based on possible misspelling errors	Enhanced search functionalities on the development agenda for last year of the project	NeOn Core
	To have a matrix reporting on structural information about the ontology like: number of classes, property, individual, max width/breadth etc.	Statistics on ontology is work in progress until May 2009	NeOn Core
Medium	To have statistics about the editorial workflow; examples of statistic are: editorial actions per ontological element, or during a time period, or per editor etc.	Only partial requirement can be tackle provided some close collaboration with other partners	WorkFlow Management
	To have a graphic interface that supports users not confident with SPARQL syntax, to build a query using elements from the ontology tab; a possible interaction can be drag and dropping concept/property/instance into an area of the window dedicated to compose the subject/predicate/object item of the query	ORAKEL and XXPlore are two plugins planned for 2009;	SPARQL

	To have the possibility to export/import an ontology into SKOS or relational data model	SKOS export/import will be achieved in 2009	NeOn Core
High	To have in place a mechanism that enables the users to instantiate relationships among individuals in the ontology, and between individuals of different ontologies, when using textual resource processing to gather new knowledge.	In early 2009 there will be LeDA, RELExO and RoLExO plugins to support advanced learning features	Text2Onto
	Moreover the users need to be supported to select the appropriate candidate entity from the list of result extracted from the text corpus, for example displaying the text excerpt related with the element on focus.	No further development	Text2Onto
	To be supported for creating ontology modules responding to specific contexts: domain topic, language, standard code, editorial duties	Several complete modularization use cases will be studied including scenarios from FAO if appropriate and sufficiently specified.	Modules
	To have the access to entities into a relational DB without necessary have to migrate the data into an ontology project, but instantiating links between the ontology model and the DB model	No plans for runtime access to databases for OWL	NeOn Core