NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D1.3.2 Change management to support collaborative workflows

Deliverable Co-ordinator: **Raul Palma**

Deliverable Co-ordinating Institution: **UPM**

Other Authors: **Peter Haase; Qiu Ji**

In this deliverable we present the implementation of the methods and strategies for propagating networked ontologies that we proposed in NeOn deliverable D1.3.1 in order to support the collaborative ontology development based on editorial workflows. Our implementation realises a major component for case study task T7.4 that we used as a test case. We also present the experiments we conducted to evaluate our approach and the results of the evaluations. In particular, the implementation was tested by the FAO case study partner in the development of fisheries ontologies produced within WP7 that will underpin the Fisheries Stock Depletion Assessment System (FSDAS).

Document Identifier:	NEON/2008/D1.3.2/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 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.ump.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 SL–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 – 665 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 Kolenz-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. Marino 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 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:

- Universidad Politécnica de Madrid (UPM)
- University of Karlsruhe (UKARL)
- Ontoprise (ONTO)

Change Log

Version	Date	Amended by	Changes
0.1	01-09-2008	Raul Palma	Creation
0.2	05-09-2008	Raul Palma	TOC
0.3	10-09-2008	Raul Palma	Experiment design
0.4	01-10-2008	Raul Palma	Introduction
0.5	05-10-2008	Raul Palma	Implementation section
0.6	31-10-2008	Raul Palma	Execution of experiment
0.7	05-11-2008	Raul Palma	Analysis of completeness
0.8	15-11-2008	Raul Palma	Analysis of experiments results
0.9	15-11-2008	Raul Palma	conclusions
0.95	15-12-2008	Peter Haase	proof reading, minor corrections
1.0	15-01-2009	Peter Haase, Raul Palma	corrections after review

Executive Summary

In this deliverable we present the implementation of the methods and strategies for propagating networked ontologies that we proposed in NeOn deliverable D1.3.1 ([PHWd07]) to support the collaborative ontology development based on editorial workflows. Collaborative ontology development is one of the most significant problems in ontological engineering. It is especially important when developing large scale and/or multidisciplinary ontologies. Our implementation provides a complete infrastructure that addresses this problem by modelling the process followed by organisations to coordinate the collaborative ontology development. Additionally, our solution fits the requirement from case study workpackages and in particular, it realises a major component for case study task T7.4.

Providing a complete solution to support the collaborative ontology development process involves on the one hand conceptual models that provide the foundations to represent the required information for the solution (presented in D1.3.1) and on the other hand several components that support related aspects including the following:

- Change Management Components
- Workflow Management Components
- Ontology Editing and Visualization Components
- Distributed Registry

We first present our prototype framework to support the collaborative ontology development by providing a detailed description of each of its components. The prototype consists of a set of plugins for the NeOn Toolkit. Then, we describe the experiments we conducted in collaboration with FAO case study partner to evaluate our framework. In particular, the implementation was tested by FAO ontology editors for the development of fisheries ontologies produced within WP7 that will underpin the Fisheries Stock Depletion Assessment System (FSDAS). Finally, we present the results of the evaluation and analyse our next steps accordingly.

In a nutshell, this deliverable reports the first comprehensive software prototype as part of the NeOn Toolkit that implements the methods and strategies presented in D1.3.1 to support the collaborative ontology development process. The planned next step is to improve the prototype based on the evaluation results.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The growing use and application of ontologies in the last years has led to an increased interest of researchers in the development of ontologies, either from scratch or by reusing existing ones. This situation, however, demands also a bigger effort in the maintenance and management of ontologies. Ontology development and maintenance activities are addressed by many different methodologies (e.g. Methontology[FPJ97], On-To-Knowledge[SSSS01], DILIGENT[Tem06], etc.). However, most of them only consider the development of ontologies by single users or a small group of ontology engineers placed in the same location. More important is that even though they address the methodological aspects, in general they focus less on the process followed by organisations to coordinate the collaborative ontology development. In practice ontologies may be distributed, and a whole team of ontology engineers with different roles may collaborate in the development and maintenance, usually following a well defined process. Examples of such collaborative development processes can be found in international institutions like the United Nations Food and Agriculture Organisation (FAO), who are developing and maintaining large ontologies in the fishery domain [MGGPISK07]. Other similar examples are those of the Gene Ontology (GO) project[1], which addresses the need for consistent descriptions of gene products in different databases, the caGrid project[2], which aims at providing a virtual informatics infrastructure that connects data, research tools, scientists, and organizations, etc.

Consequently, in this collaborative organisational setting, existing approaches are not enough to support all ontology development and maintenance needs. Furthermore, although recently some proposals and tools have been designed specifically to support collaborative ontology development (e.g. client-server mode in Protégé along with the PROMPT and change-management plugins), they generally only address parts of the overall problem (see NeOn Deliverable 1.3.1). Most of the existing advanced ontology tools (e.g. Protégé core system, SWOOP, etc.) support only the single-user scenario, where there is just one user involved in the development and later modification of the ontologies. With such tools, a typical scenario of collaborative ontology development would look as follows: An editor changes an ontology using his ontology editor system and then sends (e.g. using email or uploading it to an ontology repository) his locally changed ontology to other users (i.e. to add more changes using their own Protégé system, or review current changes). Even in the scenario where all users are editing the same ontology stored in a central server (e.g. using client-server mode in Protégé), the coordination of the actions of the editors (e.g. when editors want their changes to be reviewed or what kind of actions they can perform) is not yet fully supported.

As we can see from the previous discussion, in this type of collaborative scenario, change management is central. Dealing with the ontology changes involves the execution of many related tasks identified in the context of the ontology evolution process. For instance, among these tasks are the capturing and formal representation of ontology changes, the verification of the ontology consistency after the changes are performed and the propagation of those changes to the ontology related entities[3]. Hence, we need appropriate procedures (and corresponding infrastructure) to control and support the management of ontology changes.

[1] `http://www.geneontology.org/`

[2] `http://www.cagrid.org/`

[3] For the particular case of propagation of changes between ontologies and related semantic metadata(annotations) we refer the reader to NeOn Deliverable 1.5.2 [MAP$^+$08]

This procedure can be modelled as a collaborative workflow, which according to [GLP+07], is a special case of epistemic workflow characterized by the ultimate goal of designing networked ontologies and by specific relations among designers, ontology elements, and collaborative tasks. The need for such workflows has also been acknowledged in the past by other related works (e.g. [TN07]). An example of such workflow is that followed by the FAO (described in [MGGPISK07]), which we take as a use case in our work, in order to derive a generic set of required activities to support it.

1.1 Motivation

One of the goals of the FAO use case partner is that fisheries ontologies produced within WP7 will underpin the Fisheries Stock Depletion Assessment System (FSDAS).

However, for such a dynamic domain like fisheries that is continuously evolving, we will need to provide the appropriate support for a successful implementation and service delivery of the FSDAS. In particular, it will be crucial to support *ontology editing* and *maintenance* activities in order to incorporate and continuously reflect *changes* in the domain in the related ontologies.

The full lifecycle of the fisheries ontologies is introduced in [MGKS+07]. It consists of six major steps: First, ontology engineers organize and structure the domain information (i.e. from the Fisheries FIGIS databases, Fisheries fact sheets and other information system and documents) into meaningful models at the knowledge level (*conceptualize*). In the next step, ontology engineers perform the knowledge acquisition activities with various manual or (semi)automatic methods various methods to transform unstructured, semi-structured and/or structured data sources into ontology instances (*population*). The third step is the iteration of conceptualization and population processes until getting a populated ontology that satisfies all requirements and it is considered stable. Once achieved, in step four, the ontology will enter into the test and maintenance environment, implemented through the editorial workflow in step five. The editorial workflow will allow ontology editors to consult, validate and modify the ontology keeping track of all changes in a controlled and coherent manner. Finally, once ontology editors in charge of validation consider the ontology final, they are authorized to release it on the Internet and make it available to end users and systems.

In this context, we will need to support a distributed team of ontology editors that is working collaboratively in the development of one ontology following a well defined process and maintaining a log history of its changes. However, the situation can become even more complex if the management of ontologies and related changes is also distributed e.g. distributed copies of the same ontology are edited locally and changes are propagated between copies in such a way that all copies are kept synchronized.

1.2 Overview of the deliverable

In the remainder of this deliverable we present our solution for the **management of collaborative ontology development in a distributed scenario by means of an editorial workflow** based on the methods and strategies that we proposed in NeOn deliverable D1.3.1. In particular, based on the proposed models for the representation of the workflow and ontology changes and the proposed strategies for the management of changes in distributed environments we present the implementation of the proposed solution. The remainder of this deliverable is organised as follows: In section 2 we present a brief overview of our proposed models and the implementation that provides the technological support to the models and methods presented in NeOn Deliverable 1.3.1. Section 3 introduces the way we evaluated our work: section 3.1 describes how we tested the completeness of our representation model and section 3.2 describes the experiments setting that was conducted at FAO. We present and analyze the results of the experiment and we conclude in section 4.

Chapter 2

An editorial workflow approach for collaborative ontology development

In this section we present our solution to support the collaborative ontology development and describe how it tackles the requirements identified in NeOn deliverable 1.3.1. We first present an overview of the revised conceptual models[1] that were originally presented in D1.3.1 that provide the foundations to represent the required information in our solution and then we present the implementation support. Finally, we describe different collaborative scenarios supported by our solution.

2.1 Conceptual Models Overview

2.1.0.1 Change Representation

A core element in our approach is the representation of changes (c.f. *change management requirement*). In D1.3.1 we presented our proposal for the representation of changes which integrates many of the features of the existing approaches (e.g. [Sto04], [Kle04]) in a consistent layered manner. In this deliverable we highlight only the most relevant parts of our representation of changes: We refine and extend existing work and propose a layered approach for the representation of changes that consists of a generic ontology that models generic operations in a taxonomy of changes that are expected to be supported by any ontology language and that can be specialized for specific ontology languages (e.g. OWL) while still providing a common, independent model for the representation of ontology changes. It comprises three levels for the classification of changes: Atomic (i.e. the smallest and indivisible operation that can be performed in a specific ontology model), Entity (i.e. basic operations that can be performed over ontology elements usually from an ontology editor) and Composite (i.e. group of changes applied together that constitute a logical entity). It also provides the link to capture the argumentation of changes and it relies and uses some of the knowledge defined in our early work, the Ontology Metadata Vocabulary (OMV) [HP05] to refer to ontologies and users. OMV is a metadata schema that captures relevant information about ontologies such as provenance, availability, statistics, etc. Besides the main class Ontology, OMV also models additional classes and properties required to support the reuse of ontologies, such as Organisation, Person, LicenseModel, OntologyLanguage and OntologyTask among others. Our change ontology has been implemented as an OMV extension because it models specific ontology metadata (i.e. ontology changes).

Furthermore, the change ontology provides the means to support not only the tracking of changes but also the information that identifies the original and the current version of the ontology after applying the changes (*versioning requirement*). This is not a trivial issue: even though ontologies are in general identified by an URI, in practice it is not enough to identify a particular ontology version (i.e. different versions of the same ontology have the same URI). Hence, the management of ontology versions requires a clear definition of the ontology identification. In our solution, we rely on the identification of ontologies that we presented in [HP05],

[1]Our conceptual models are available in OWL at `http://omv.ontoware.org`

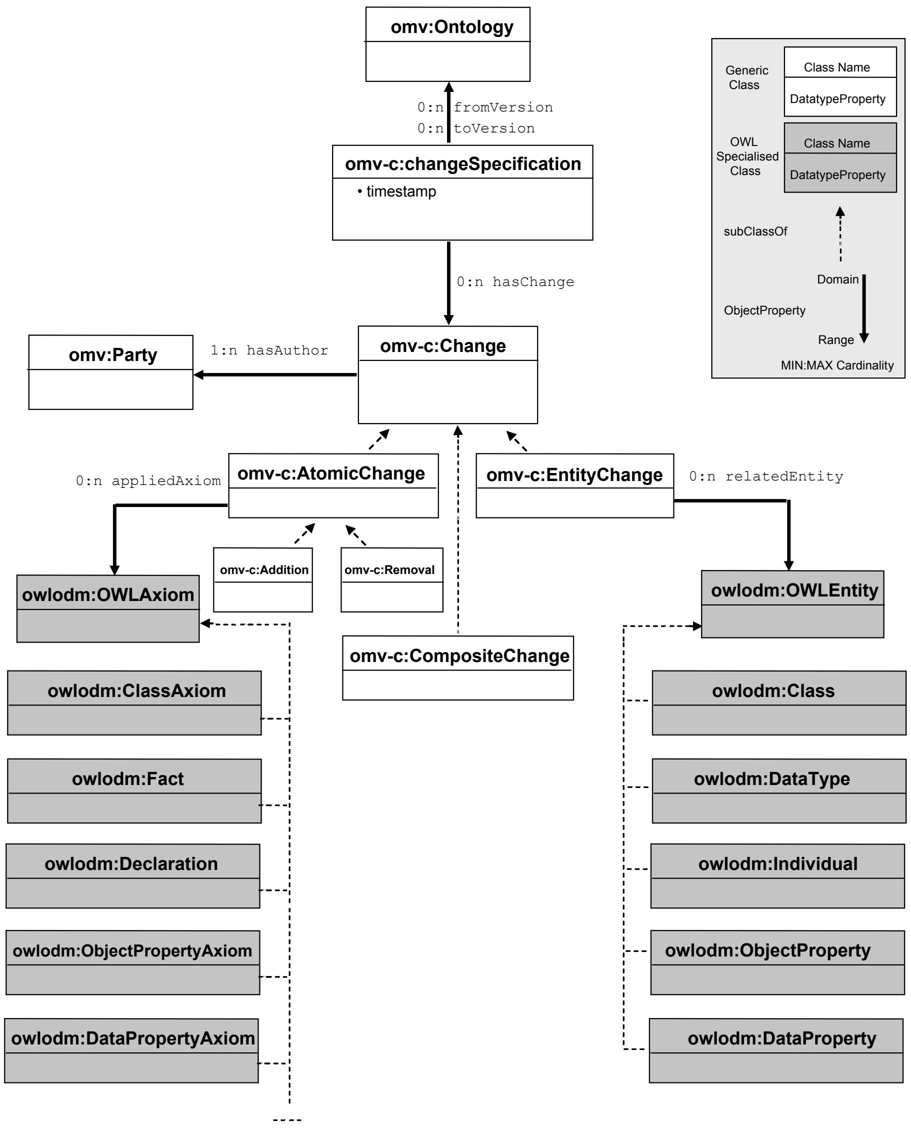


Figure 2.1: Main Classes and Properties of Change Ontology for OWL 2

which consists of a tripartite identifier: the ontology URI, the ontology version (if present), and the ontology location.

Finally, to keep track of the actual sequence of changes (i.e. the order in which changes were performed), our ontology relies on two elements: each change is linked to its predecessor via the "hasPreviousChange" object property and a "Log" class provides the pointer to the last change in the ontology history.

OWL Change Ontology Extension

The main classes and properties of the change ontology for OWL 2 are illustrated in Figure 2.1

The taxonomy of entity-level changes has been extended to model the particular changes for the OWL 2 ontology language based on the OWL 2 metamodel described in NeOn Deliverable 1.1.2 [HBP^{+}07]. Hence, the extended taxonomy includes changes for OWL elements such as objectProperties (e.g. add/remove EquivalentObjectProperties, functionalObjectProperty, etc.) or dataProperties (e.g. add/remove disjointDataProperties, functionalDataProperty, etc.) among others. Furthermore, the atomic-level changes are associated to the corresponding OWL axioms as described below. Note that the composite-level changes were not

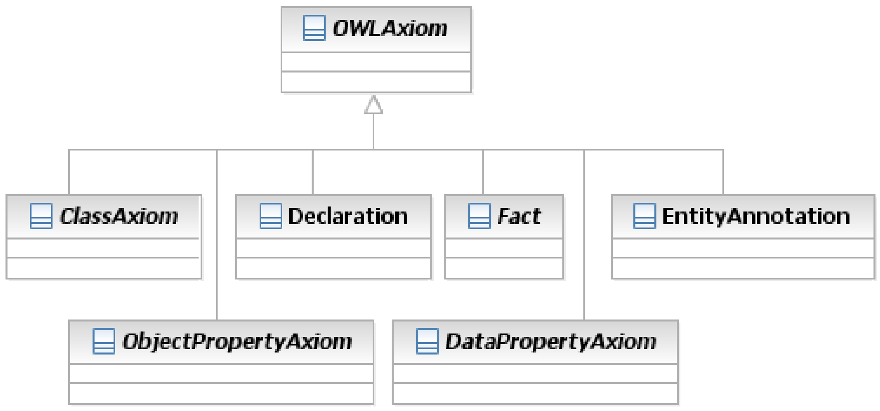


Figure 2.2: OWL 2 Axioms

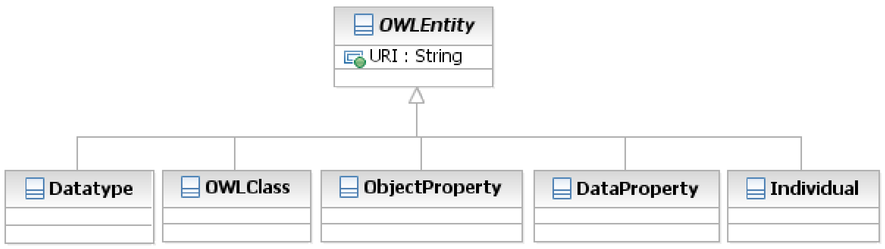


Figure 2.3: OWL 2 Entities

extended as they represent composite operations that are expected to be supported in any ontology representation language (such as move element or remove tree) and therefore they are modelled in the generic change ontology.

According to the specification of the OWL 2 Web Ontology Language in the W3C Working Draft 11 April 2008 (`http://www.w3.org/TR/2008/WD-owl2-syntax-20080411/`), OWL axioms can be classified into six main types as shown in Figure 2.2.

Four out of the six main axiom types (i.e. classAxiom, objectPropertyAxiom, dataPropertyAxiom and fact) are further specialized into subtypes: 4 class axioms, 13 objectProperty axioms, 6 dataProperty axioms and 7 fact axioms. For a complete description of all axioms we refer the reader to the NeOn Deliverable 1.1.2 [HBP$^+$07]).

The two main axiom types not specialized (i.e. declaration and entityAnnotation) plus all the axiom subtypes represent the possible *atomic operations* that can be performed over an OWL 2 ontology (i.e. add/remove axiom). Consequently we have 32 different types of atomic operations.

Furthermore, axioms are always associated to one or more OWL entities or descriptions. An OWL 2 entity can be classified into five different types as illustrated in Figure 2.3.

OWL 2 provides an expressive language for forming descriptions. A description is an abstract superclass for all class definition constructs and it is specialized into 18 different types (an owlClass is both a type of entity and a type of description). Again, we refer the reader to NeOn Deliverable 1.1.2 [HBP$^+$07] for additional information.

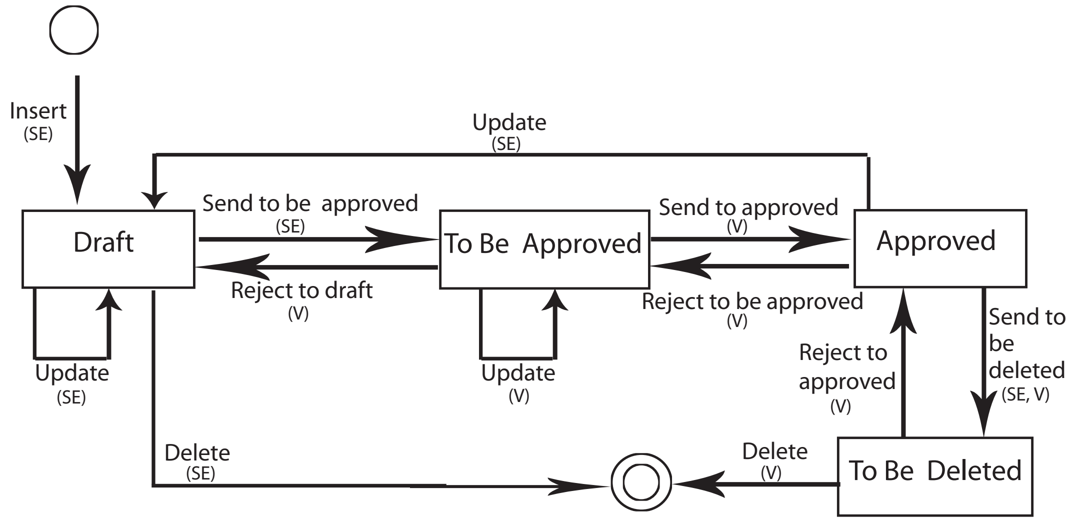


Figure 2.4: Editorial workflow at the element level

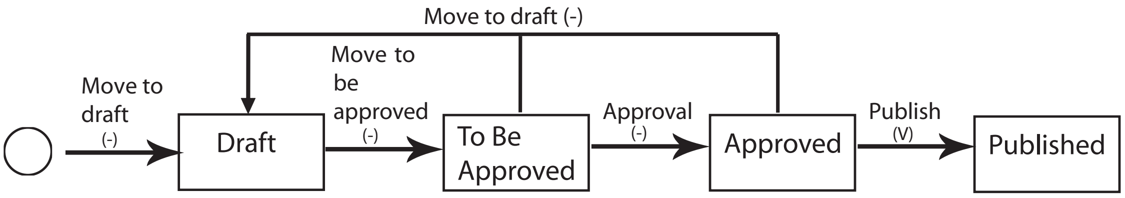


Figure 2.5: Editorial workflow at the ontology level

2.1.0.2 Workflow Model

In D1.3.1 we presented our model for the representation of the workflow. Hence, in this deliverable we briefly highlight the main contributions: Based on the analysis of the requirements presented in D1.3.1, our solution considers the editorial workflow at two levels: ontology level and ontology element level. Although the workflows can be used independently of the underlying ontology model, the specific set of ontology elements depend on the ontology model. In our approach we are mainly considering the OWL ontology model, in which an OWL ontology consists of a set of axioms and facts[2]. Facts and axioms can relate to classes, properties or individuals, and hence that is the set of ontology elements we are considering.

As previously discussed, the workflow details (e.g. the specific roles, actions, etc.) depend on the organisation setting. To exemplify, in the rest of this section we discuss our solution for the particular scenario in FAO. Figures 2.4 and 2.5 show the two different workflow levels (i.e. element and ontology level). States are denoted by rectangles and actions by arrows. The information in parenthesis specifies the actions that an editor can perform depending on its role, where "SE" denotes Subject Expert, "V" denotes Validator and "-" denotes that the action is performed automatically by the system.

The possible states (see Figure 2.4) that can be assigned to ontology elements are:

- *Draft*: This is the status assigned to any element when it passes first into the editorial workflow, or when it was approved and then updated by a subject expert.
- *To be approved*: Once a "SE" is confident with a change in draft status the element is passed to the

[2]In our current implementation we support the upcoming OWL 2 language. See `http://www.w3.org/TR/owl2-syntax/`

"To Be Approved" status, and remains there until a "V" approves/rejects it.

- *Approved*: If a "V" approves a change in an element in the "To Be Approved" status, it passes to the "Approved" status. Additionally, this is the default state for every element of the initial version of a stable ontology.
- *To be deleted*: If a "SE" or "V" consider that an element needs to be deleted, he proposes a change to delete it. The element will be flagged with the "To Be Deleted" status and removed from the ontology, although only a "V" will be able to definitively delete it.

The ontology has a state (see Figure 2.5) that is automatically assigned by the system (denoted by "-" in Figure 2.5), except for the "published" state as described below:

- *Draft*: Any change to an ontology in any state automatically sends it into draft state.
- *To be approved*: When all changes to an ontology version are in the "To Be Approved" state (or Deleted) the ontology is automatically sent to the "To Be Approved" state.
- *Approved*: When all changes to an ontology version are in "Approved" state (or Deleted) the ontology is automatically send to "Approved" state. Additionally, this is the default state of the initial version of a stable ontology.
- *Published*: Only when the ontology is in "Approved" state, can it be sent by a validator to "Published" state.

As described in NeOn deliverable 1.3.1, the editorial workflow starts after getting a stable populated ontology that satisfies all the organizational requirements. Hence, we assume that the initial state of this stable ontology (and all its elements) is "Approved"[3].

Note that during the editorial workflow, actions are performed either implicitly or explicitly. For instance, when a user updates (i.e. modifies) an element he does not explicitly perform an update action. In this case the action has to be captured from the user interface and recorded when the ontology is saved. In contrast, Validators explicitly approve/reject proposed changes and the action is recorded immediately when performed.

Workflow ontology Similarly to our change ontology, we decided to model the workflow elements (i.e. roles, status, actions) using an (OWL-Lite) ontology (i.e. a workflow ontology) that allows the formal and explicit representation of knowledge in a machine-understandable format. Furthermore, having both models (i.e. ontology changes and workflow) formalized as ontologies will facilitate the representation of the tight relationship that exists between them. For instance, consider a user with role "subject expert" who "inserts" a new ontology "class" into the ontology. That "class" will receive automatically the "draft" state. All the information related to the process of inserting a new ontology element will be captured by the workflow ontology, while the information related to the particular element inserted, along with the information about the ontology before and after the change is captured by the change ontology. Additionally, the workflow process also relies on OMV to refer to ontologies and users.

The main classes and properties of the workflow ontology and its relationships with the other ontologies in our approach are shown in Figure 2.6.

In a nutshell (for a complete description please refer to D1.3.1), the different roles of the ontology editors are modelled as individuals of the `Role` class that is related to the `Person` class of the OMV core ontology (i.e. a person has a role). Similarly to the roles, the possible values of the states (for entities and ontologies) are modelled as individuals of `EntityState` and `OntologyState` respectively.

[3]In a different scenario, the workflow could start with an empty ontology (without elements), which we could assume that will be by default in "Approved" state

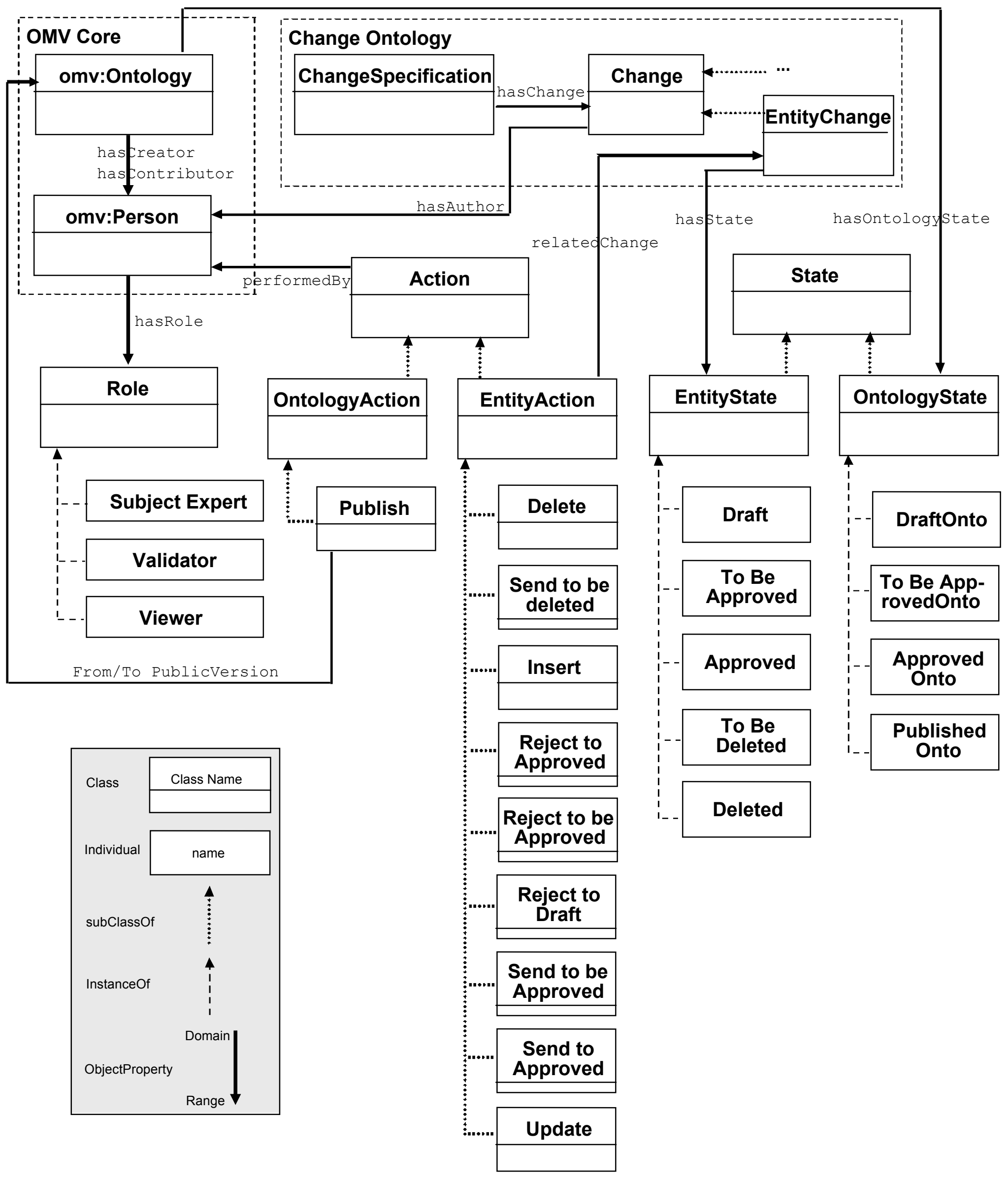


Figure 2.6: Workflow ontology

The properties `hasEntityState` and `hasOntologyState` model the relationship between the ontology element/ontology and its appropriate state. Finally, the possible entity and ontology actions are modelled accordingly as subclasses of `EntityAction` and `OntologyAction` and their relationship with the corresponding ontology element/ontology is represented with the properties `relatedChange` and `relatedOntology`.

2.2 Implementation

2.2.1 Implementation Support

Our approach has been implemented within the NeOn Toolkit[4], an extensible ontology engineering environment based on Eclipse, by means of a set of plugins and extensions. A high level conceptual architectural

[4]http://www.neon-toolkit.org/

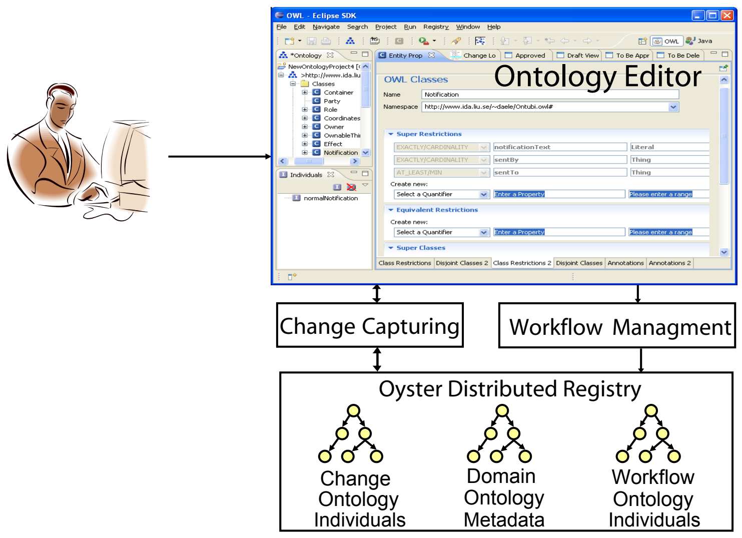

Ontology Editor
Change Capturing
Workflow Managment
Oyster Distributed Registry
Change Ontology Individuals
Domain Ontology Metadata
Workflow Ontology Individuals

Figure 2.7: Conceptual architecture for the collaborative ontology development support

diagram of the involved components is shown in Figure 2.7. We present in the following, first the change capturing related components (i.e. left side of the figure), then the workflow management related components (right side of the figure), next the user related components for editing and visualizing ontologies (and related information) in the editorial workflow (upper part of the figure) and finally our distributed registry implementation (bottom part of the figure). Unlike similar existing tools (see NeOn Deliverable 1.3.1), our solution provides a flexible mechanism to supports different collaborative scenarios as we describe in detail at the end of this section. For example, in addition to the typical scenario where a team of ontology editors are working collaboratively with a centralized copy of an ontology, we also support scenarios in which ontology editors are working collaboratively with distributed copies of the same ontology.

2.2.1.1 Change Capturing Components

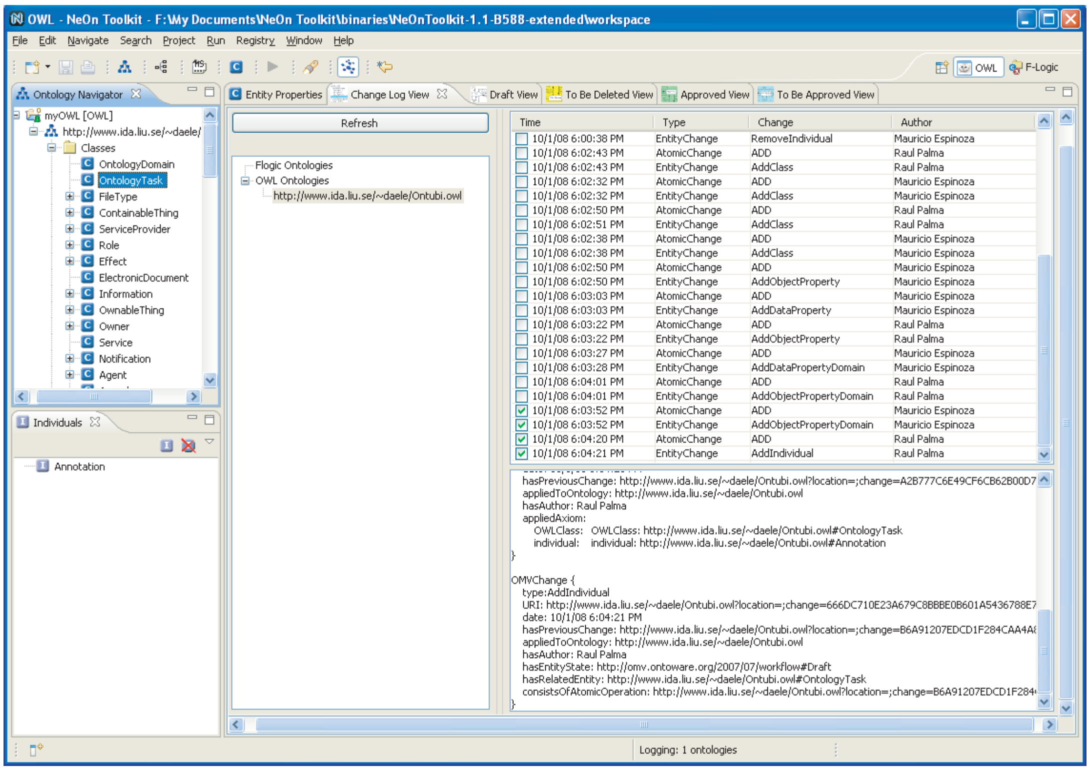
Once the ontology editor specifies that he wants to monitor an ontology, changes are automatically captured (*change management requirement*) from the ontology editor by a change capturing plugin. This plugin is notified about events that consist of ontology changes performed by the user in the ontology editor. For each of these events, the change is represented according to the change ontology by creating the appropriate individual. For example, adding a class individual in the ontology editor creates the entity change "Add Individual" and the two corresponding atomic changes (OWL 2 axioms): "Add Declaration" and "Add Class-Member". As described by the change ontology, each individual includes relevant information such as the author, the time, the related ontology, etc. The individuals are stored in the Oyster distributed registry [PH05] [5].

This plugin is also in charge of applying changes received from other clients to the same ontology after Oyster synchronizes the changes in the distributed environment (see last subsection). Finally, this plugin extends the NeOn Toolkit with a view to display the history of ontology changes (see Figure 2.8) (*visualisation requirements*).

2.2.1.2 Workflow Management Components

In our implementation, the workflow management component (i) takes care of enforcing the constraints imposed by the collaborative workflow, (ii) creates the appropriate action individuals of the workflow ontology

[5] In a different scenario, if the system is presented with two versions of the same ontology without their change history, ontology changes could be derived according to the change ontology, similar to the PROMPT tool[NM02]



	Time	Type	Change	Author
☐	10/1/08 6:00:38 PM	EntityChange	RemoveIndividual	Mauricio Espinoza
☐	10/1/08 6:02:43 PM	AtomicChange	ADD	Raul Palma
☐	10/1/08 6:02:43 PM	EntityChange	AddClass	Raul Palma
☐	10/1/08 6:02:32 PM	AtomicChange	ADD	Mauricio Espinoza
☐	10/1/08 6:02:32 PM	EntityChange	AddClass	Mauricio Espinoza
☐	10/1/08 6:02:50 PM	AtomicChange	ADD	Raul Palma
☐	10/1/08 6:02:51 PM	EntityChange	AddClass	Raul Palma
☐	10/1/08 6:02:38 PM	AtomicChange	ADD	Mauricio Espinoza
☐	10/1/08 6:02:38 PM	EntityChange	AddClass	Mauricio Espinoza
☐	10/1/08 6:02:50 PM	AtomicChange	ADD	Mauricio Espinoza
☐	10/1/08 6:02:50 PM	EntityChange	AddObjectProperty	Mauricio Espinoza
☐	10/1/08 6:03:03 PM	AtomicChange	ADD	Mauricio Espinoza
☐	10/1/08 6:03:03 PM	EntityChange	AddDataProperty	Mauricio Espinoza
☐	10/1/08 6:03:22 PM	AtomicChange	ADD	Raul Palma
☐	10/1/08 6:03:22 PM	EntityChange	AddObjectProperty	Raul Palma
☐	10/1/08 6:03:27 PM	AtomicChange	ADD	Mauricio Espinoza
☐	10/1/08 6:03:28 PM	EntityChange	AddDataPropertyDomain	Mauricio Espinoza
☐	10/1/08 6:04:01 PM	AtomicChange	ADD	Raul Palma
☐	10/1/08 6:04:01 PM	EntityChange	AddObjectPropertyDomain	Raul Palma
✓	10/1/08 6:03:52 PM	AtomicChange	ADD	Mauricio Espinoza
✓	10/1/08 6:03:52 PM	EntityChange	AddObjectPropertyDomain	Mauricio Espinoza
✓	10/1/08 6:04:20 PM	AtomicChange	ADD	Raul Palma
✓	10/1/08 6:04:21 PM	EntityChange	AddIndividual	Raul Palma

```
   hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=A2B777C6E49CF6CB62B00D7
   appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
   hasAuthor: Raul Palma
   appliedAxiom:
      OWLClass:   OWLClass: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
      individual:   individual: http://www.ida.liu.se/~daele/Ontubi.owl#Annotation
}

OMVChange {
   type:AddIndividual
   URI: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=666DC710E23A679C8BBBE0B601A5436788E7
   date: 10/1/08 6:04:21 PM
   hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=B6A91207EDCD1F284CAA4A
   appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
   hasAuthor: Raul Palma
   hasEntityState: http://omv.ontoware.org/2007/07/workflow#Draft
   hasRelatedEntity: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
   consistsOfAtomicOperation: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=B6A91207EDCD1F284
}
```

Figure 2.8: Change log view

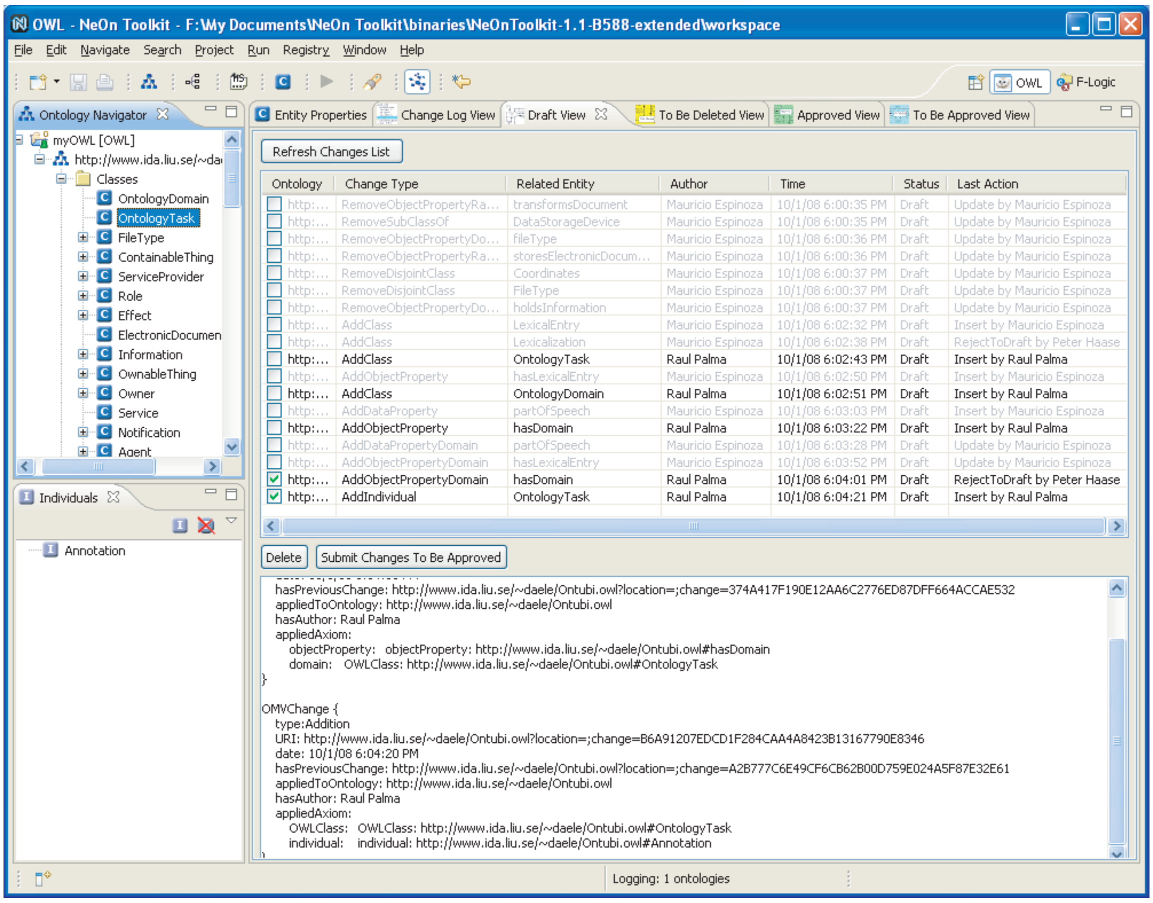
and (iii) registers them into the distributed registry. Hence, whenever a new workflow action is performed, the component performs the following tasks:

- It gets the identity and role of the user performing the action (if it is an explicit action) e.g. send to approve, or the associated change (if it is an implicit action) e.g. adding a new class implicitly creates an insert action.
- It gets the status of the ontology element associated to the action/change.
- It verifies that the role associated to the user can perform the requested action when the ontology element is in that particular status.
- If the verification succeeds, it creates the workflow action and registers it.
- If the verification fails, it undoes the associated change(s) for the implicit actions because the complete operation (e.g. adding a new class) failed.

2.2.1.3 Ontology Editing and Visualization Components

To support the workflow activities (*workflow activities requirements*) we rely on the NeOn Toolkit which comes with an ontology editor that allows the editing of ontology elements. Additionally, according to the *visualisation requirements* the NeOn Toolkit is extended with a set of views that allow editors to (i) see the appropriate information of ontologies in the editorial workflow and (ii) perform (as described in 2.1.0.2) the applicable workflow actions (*approve, reject*, etc.), depending on their role. There are four views[6]:

[6]Subject experts see the first two views, validators see the latter three.



	Ontology	Change Type	Related Entity	Author	Time	Status	Last Action
☐	http:...	RemoveObjectPropertyRa...	transformsDocument	Mauricio Espinoza	10/1/08 6:00:35 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveSubClassOf	DataStorageDevice	Mauricio Espinoza	10/1/08 6:00:35 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveObjectPropertyDo...	fileType	Mauricio Espinoza	10/1/08 6:00:36 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveObjectPropertyRa...	storesElectronicDocum...	Mauricio Espinoza	10/1/08 6:00:36 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveDisjointClass	Coordinates	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveDisjointClass	FileType	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
☐	http:...	RemoveObjectPropertyDo...	holdsInformation	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
☐	http:...	AddClass	LexicalEntry	Mauricio Espinoza	10/1/08 6:02:32 PM	Draft	Insert by Mauricio Espinoza
☐	http:...	AddClass	Lexicalization	Mauricio Espinoza	10/1/08 6:02:38 PM	Draft	RejectToDraft by Peter Haase
☐	http:...	AddClass	OntologyTask	Raul Palma	10/1/08 6:02:43 PM	Draft	Insert by Raul Palma
☐	http:...	AddObjectProperty	hasLexicalEntry	Mauricio Espinoza	10/1/08 6:02:50 PM	Draft	Insert by Mauricio Espinoza
☐	http:...	AddClass	OntologyDomain	Raul Palma	10/1/08 6:02:51 PM	Draft	Insert by Raul Palma
☐	http:...	AddDataProperty	partOfSpeech	Mauricio Espinoza	10/1/08 6:03:03 PM	Draft	Insert by Mauricio Espinoza
☐	http:...	AddObjectProperty	hasDomain	Raul Palma	10/1/08 6:03:22 PM	Draft	Insert by Raul Palma
☐	http:...	AddDataPropertyDomain	partOfSpeech	Mauricio Espinoza	10/1/08 6:03:28 PM	Draft	Update by Mauricio Espinoza
☐	http:...	AddObjectPropertyDomain	hasLexicalEntry	Mauricio Espinoza	10/1/08 6:03:52 PM	Draft	Update by Mauricio Espinoza
✓	http:...	AddObjectPropertyDomain	hasDomain	Raul Palma	10/1/08 6:04:01 PM	Draft	RejectToDraft by Peter Haase
✓	http:...	AddIndividual	OntologyTask	Raul Palma	10/1/08 6:04:21 PM	Draft	Insert by Raul Palma

```
  hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=374A417F190E12AA6C2776ED87DFF664ACCAE532
  appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
  hasAuthor: Raul Palma
  appliedAxiom:
    objectProperty:   objectProperty: http://www.ida.liu.se/~daele/Ontubi.owl#hasDomain
    domain:   OWLClass: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
}

OMVChange {
  type:Addition
  URI: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=B6A91207EDCD1F284CAA4A8423B13167790E8346
  date: 10/1/08 6:04:20 PM
  hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=A2B777C6E49CF6CB62B00D759E024A5F87E32E61
  appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
  hasAuthor: Raul Palma
  appliedAxiom:
    OWLClass:   OWLClass: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
    individual:   individual: http://www.ida.liu.se/~daele/Ontubi.owl#Annotation
```

Figure 2.9: Draft View in the NeOn Toolkit

- *Draft view*: Shows all proposed changes (from all editors) to that ontology version. In accordance to FAO scenario the changes of the current editor are editable while changes from other editors are non editable (see Figure 2.9).
- *Approved view*: Shows the approved changes.
- *To Be Approved view*: Shows all changes (from all editors) pending to be approved.
- *To Be Deleted view*: Shows all proposed deletions (from all editors).

2.2.1.4 Distributed Registry

Ontologies are stored within a repository and their metadata is managed by the Oyster distributed registry[7] (*change management requirement*). The metadata includes information about ontologies and users (represented using OMV), the changes to the ontology (represented using the change ontology) and about the actions performed (represented using the workflow ontology). For each change the status is also kept to support the editorial workflow. When a new change is registered into an Oyster node, Oyster automatically updates the log history keeping track of the chronological order of changes. In particular, it performs the following actions:

- gets the last registered change (using the "Log" class)

[7] http://ontoware.org/projects/oyster2/

- adds it as the previous change of the current one
- updates the "Log" class to point to the current change

The local Oyster nodes contact each other creating a distributed ontology registry. In this distributed environment, Oyster also propagates the ontology changes, thus allowing the notification of new changes to ontology editors (*change management requirement*). That is, once we have the required changes in a machine-understandable format, the system propagates them to the distributed copies of the ontology. In other words, changes are propagated to each node in the distributed network that maintains a copy of the ontology (and wants to receive those changes). There are two possible approaches for the propagation: push or pull. The benefits and disadvantages of both approaches have been already analyzed (e.g. [Sto04]). Since we are considering a distributed environment where we cannot guarantee the availability of nodes, we follow a combination approach of a push and pull mechanism that we call *synchronization*. During the synchronization, nodes periodically contact other nodes in the network to exchange updated information (pull changes) and optionally they can push their changes to a specific node (called the super node) such that if a node goes offline before all other nodes pull the new changes, the node changes are not lost. In this way, Oyster minimizes the conflicts or inconsistencies due to concurrent editing as it automatically synchronizes changes periodically (and it allows to force the synchronization immediately) in the distributed environment such that every editor will have an up-to-date copy of the ontology with the proposed changes (*concurrency control and conflict resolution requirement*). Nevertheless, conflicts in the collaborative workflow could still occur as logical conflicts in the form of inconsistencies or conflicts due to concurrent editing of an ontology. The strategies to deal with those potential problems are discussed in D1.3.1.

2.2.2 Possible scenarios and configurations of the framework

2.2.2.1 Configuration A

In this scenario, the team of ontology editors will be maintaining collaboratively the ontology "A" following a well defined process (i.e. workflow). The following characteristics describe the environment (see figure 2.10):

- There is only one copy of the ontology which is stored in a central server.
- Ontology editors are working in a distributed manner (i.e. they are localized at different PCs).
- Ontology editors are working concurrently.
- Each ontology editor uses his own NeOn Toolkit installation and connects to the central server.
- Each ontology editor specifies his credentials (e.g. name and role) in his NeOn Toolkit.
- The ontologies metadata (including change information) from all ontology editors are stored at one specific place.
- Each NeOn Toolkit has configured that metadata is stored at a specific location (i.e. a specific ontology registry -Oyster-) and connects to it.

Configuration of machines

- One PC is configured as the server. This PC has to be running the NeOn collaboration server.
- One PC (probably the same as above) is configured as the metadata provider (also known as the "super-node"). This PC has to be running Oyster (it is an Oyster node). In this scenario, Oyster is running in server mode, although it can be running in any of the possible ways e.g. within a NeOn Toolkit installation, as the Oyster java GUI, or within any other application via its API. For additional information about the push-node see Section 2.2.1.4.

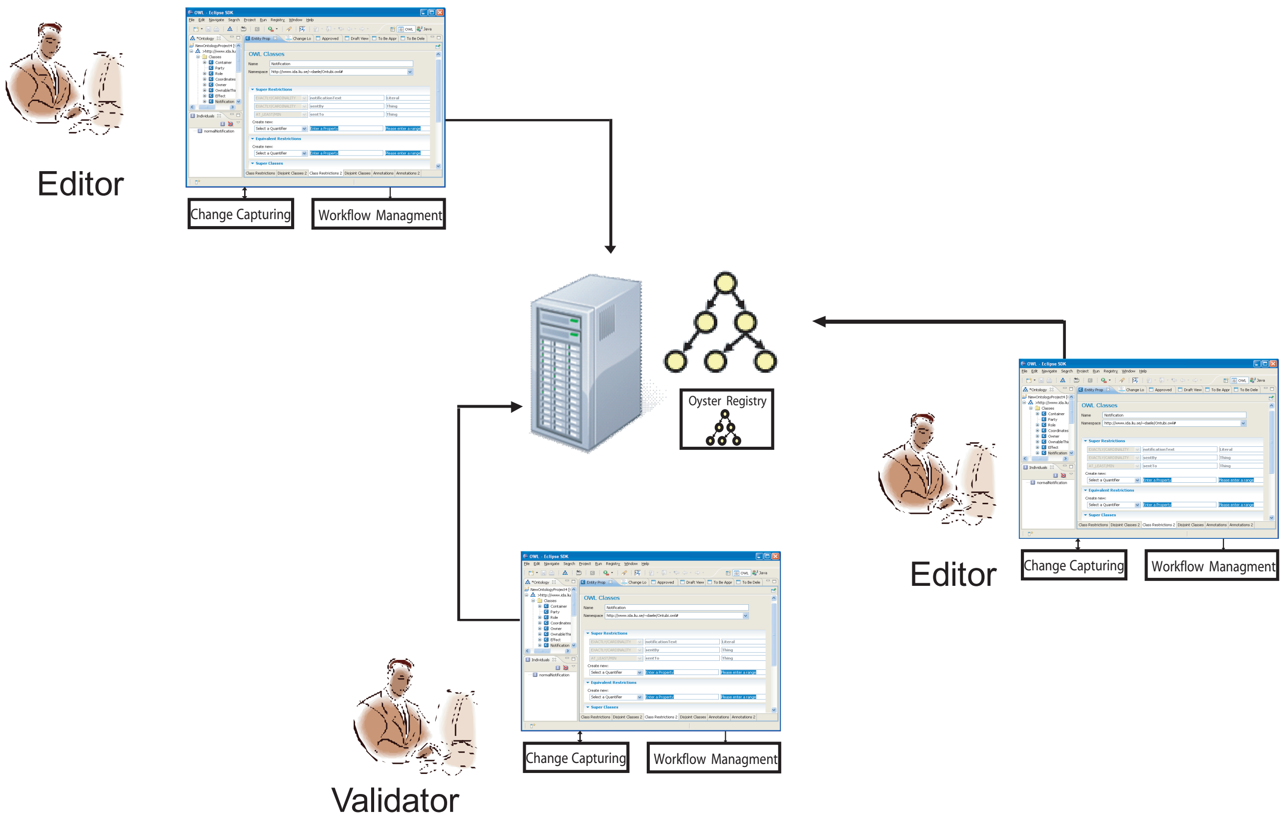


Figure 2.10: Configuration A

- The PCs used by the ontology editors are only running a NeOn Toolkit installation configured as described above.

Key benefits of configuration A

- Ontology editors are able to work concurrently. The NeOn collaboration server allows multiple clients for the same ontology at the same time.
- Conflicts are automatically handled by the NeOn collaboration server.

Drawbacks of configuration A

- The collaboration server has to be online at every moment or ontology editors won't be able to work.
- If the collaboration server (or the metadata provider) becomes permanently unavailable, the ontology (or the metadata) is lost.
- The network connection between the clients (PCs running NeOn Toolkit) and the collaboration server has to be fast and reliable because clients are communicating to the server at every moment they are maintaining the ontology.
- The collaboration server does not provide (at this moment) relevant information regarding the provenance of ontology changes or identity of clients.

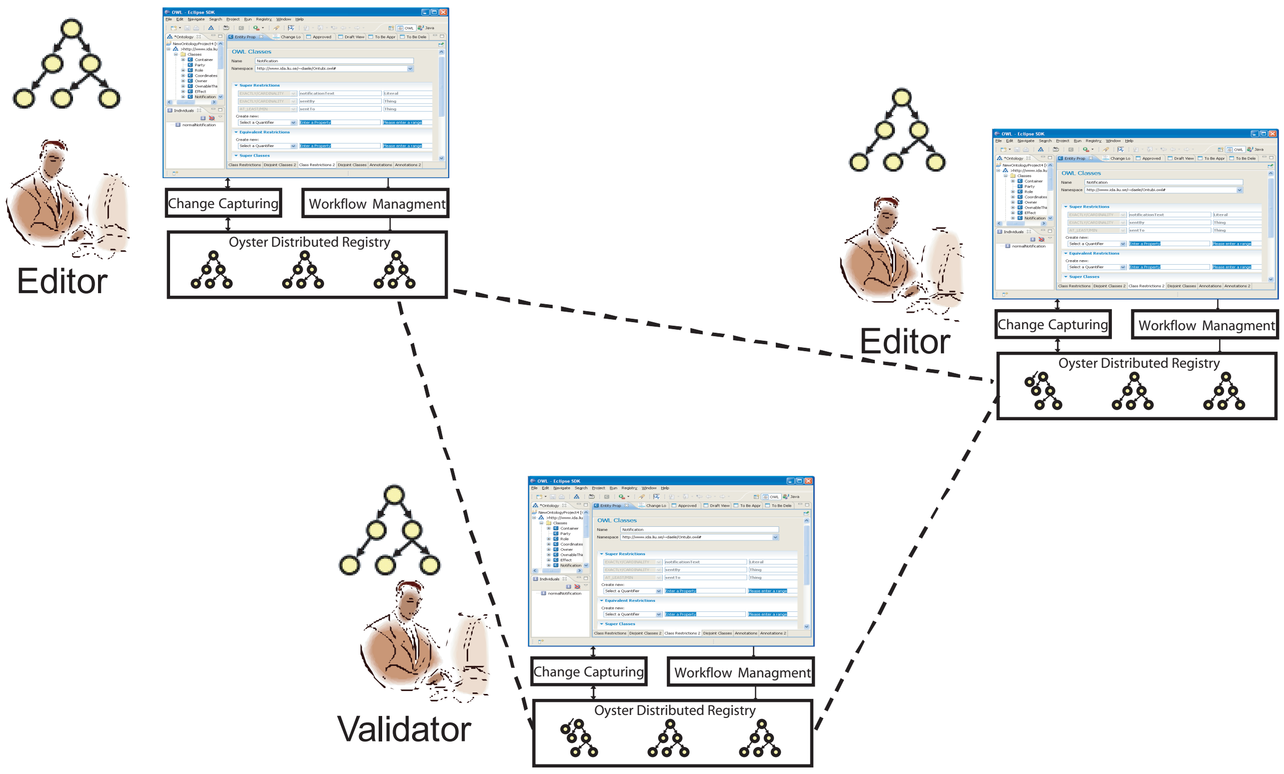


Figure 2.11: Configuration B

Recommended use of configuration A

This configuration is the most suitable for the cases when the members of the team of ontology editors are closely located (e.g. organization LAN) and when there are many ontology editors that are usually working at similar times.

2.2.2.2 Configuration B

In this scenario, the team of ontology editors will be maintaining collaboratively the ontology "B" following a well defined process (i.e. workflow). The following characteristics describe the environment (see figure 2.11):

- There are "n" copies of ontology "B", where "n" is the number of ontology editors working on that ontology. Each ontology editor has its own copy located at his PC.
- Ontology editors are working in a distributed manner (i.e. they are localized at different PCs).
- Ontology editors are not working concurrently.
- Each ontology editor uses his own NeOn Toolkit installation to work with his local copy of the ontology.
- Each ontology editor specifies his credentials (e.g. name and role) in his NeOn Toolkit.
- Each NeOn Toolkit is running a local ontology registry (i.e. Oyster).
- The ontology metadata (including change information) is stored in the local registry.
- The local registries (i.e. each Oyster node) exchange information about ontology metadata and synchronize the ontology changes. Changes received from other nodes are applied locally in the ontology copy to keep the distributed copies also synchronized (the synchronization process is described in Section 2.2.1.4).

Machines configuration

- The PCs used by the ontology editors are running a NeOn Toolkit installation.
- Each PC is running the ontology registry Oyster (i.e. it is an Oyster node).
- Optionally, one PC is configured as the "push-node". This PC is running Oyster. In this scenario, Oyster is running in server mode, although it can be running in any of the possible ways e.g. within a NeOn Toolkit installation, as the Oyster java GUI, or within any other application via its API. The push-node guaranties that no changes are lost even if the clients are going frequently offline (See section 2.2.1.4 for additional information about the push-node).

Key benefits of configuration B

- Ontology editors do not need a permanent network connection: they are working locally.
- The network connection does not have to be fast: it is used only during the synchronization process.
- There is no one single point of failure. The ontology (and metadata) is available in many different locations (i.e. each PC).
- Ontology editors can work independently. They dont't need to have another machine online (i.e. server) to work.

Drawbacks of configuration B

- There is no support for conflict resolution (at this moment) and therefore ontology editors cannot work concurrently.

Recommended use of configuration B
This configuration is the most suitable for the cases when the members of the team of ontology editors are geographically distributed (e.g. different organizations, different countries, etc.), without a reliable and fast network connection and when they are not working usually at similar times (e.g. when people are working on different countries with different time zones, or when they coordinate to work at different times).

2.2.2.3 Configuration C

In this scenario, the team of ontology editors will be maintaining collaboratively the ontology "C" following a well defined process (i.e. workflow). This scenario is a hybrid between scenarios "A" and "B". In this scenario, the team of ontology editors can be divided in at least two different groups: "X" and "Y". Each of these groups is working in an environment with the characteristics and machine configuration of scenario A. Additionally, the groups have between them an environment with the characteristics and machine configuration of scenario B. That is, each group has a collaboration server (and metadata provider), which in turn are treated as the ontology editor's PCs in configuration B (see figure 2.12).

Key benefits of configuration C

- The groups do not need a permanent network connection between each other.
- The network connection does not have to be fast between the groups: it is used only during the synchronization process.

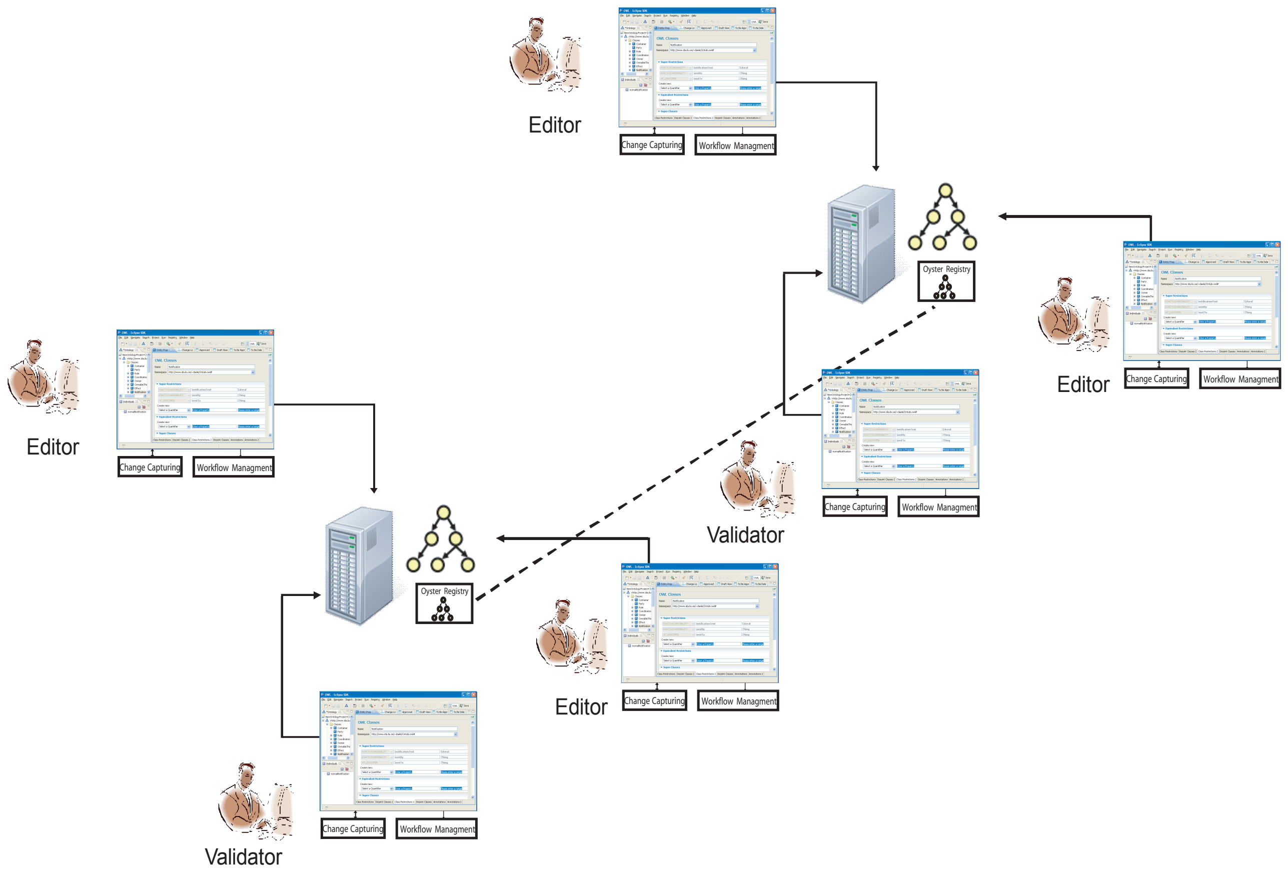


Figure 2.12: Configuration C

- There is no one single point of failure. The ontology (and metadata) is available in many different locations (i.e. each collaboration server (and metadata provider)).
- Ontology editors of each group can work independently.
- Ontology editors are able to work concurrently within each group. The NeOn collaboration server allows multiple clients for the same ontology at the same time.
- Conflicts are automatically handled by the NeOn collaboration server within each group.

Drawbacks of configuration C

- There is no support for conflict resolution between groups (at this moment) and therefore groups cannot work concurrently.
- The collaboration server has to be online at each group every moment or ontology editors won't be able to work.
- The network connection between the clients (PCs running NeOn Toolkit) and the collaboration server in each group has to be fast and reliable because clients are communicating to the server at every moment they are maintaining the ontology.
- The collaboration server in each group does not provide (at this moment) relevant information regarding the provenance of ontology changes or identity of clients.

Recommended use of configuration C

This configuration is the most suitable for the cases when the members of the team of ontology editors are affiliated to different parties. At each party, the ontology editors are closely located (e.g. organization LAN) and there are many ontology editors that are usually working at similar times. Additionally the parties are geographically distributed (e.g. different organizations, different countries, etc.), without a reliable and fast network connection and they are not working usually at similar times (e.g. when people are working on different countries with different time zones, or when they coordinate to work at different times). A typical example is a team composed of ontology editors from different organizations e.g. two universities. Within each university the editors are working at similar times but each university has a different schedule.

Chapter 3

Evaluation

We evaluated the components of our infrastructure using WP7 case study as test case. First, we prove the completeness of the change representation model (see 3.1). We analyzed the full set of possible changes for the OWL 2 ontology language and verified that every possible change can be represented in our model using as baseline the OWL 2 metamodel. Second, we conducted an experiment in FAO headquarters with a set of representative users that we describe in section 3.2.

3.1 Completeness of the change representation model with respect to the OWL 2 ontology language

To measure the completeness of the change representation model, we programmatically provided the system[1] with a set of changes consisting of at least one for each possible axiom and at least one associated for every possible type of entity/description. In particular, we simulated programmatically the changes in a test ontology[2] and let our system capture and represent each change (i.e. generate the change ontology instances). However, it was not possible to simulate all of them because a few of them are not yet supported by the NeOn Toolkit. For those few changes, we had to register them directly into the registry (programmatically) to simulate their execution to the ontology. In detail, we could simulate the following set of 52 changes as if they had happened in the ontology:

- 20 class axioms
  - (4) one for each possible axiom
  - (16) one for each possible description, except for owlClass because it was used in the previous 4 and objectExistsSelf because KAON2 does not serialize it correctly and the OWL file becomes corrupt).
- 5 declaration axioms
  - (4) one for each possible entity, except for datatype declaration that is not supported.
  - (1) one for the special type of entity called AnnotationProperty. It is mentioned in the specification but it was not still officially a type of entity.
- 5 dataProperty axioms (one for each possible axiom, except disjointDataProperties which is not supported).
- 12 objectProperty axioms (one for each possible axiom, except disjointObjectProperties which is not supported).

[1]The code is available as a JUNIT test (AllOWLChangesTest.java) included in the Change Logging plugin release
[2]For the test we used the pizza ontology `http://www.co-ode.org/ontologies/pizza/pizza_20041007.owl`

- 5 fact axioms (one for each possible axiom, except for negativeObjectPropertyAssertion and negativeDataPropertyAssertion which are not supported).
- 5 entityAnnotation axioms (one for each possible entity)

Therefore, the set of 6 changes that we had to register directly into the registry (programmatically) was:

- one class axiom associated to a objectExistsSelf description.
- Declaration of datatype.
- disjointDataProperties.
- disjointObjectProperties.
- negativeObjectPropertyAssertion.
- negativeDataPropertyAssertion.

After the test was completed, we verified that each of the 58 changes could be correctly represented in our representation model. For this task, we visualized the contents of the registry (190 KB) (i.e. the captured changes) from the Change Log View (c.f. 2.8) and the Draft View (c.f. 2.9) of our infrastructure. **The conclusion is that every change was successfully represented in our model**. That is for each of the 58 changes, we had 58 correct instances of the atomicChange concept of our model and 57 correct instances of the EntityChange concept of our model. Note that the relationship between atomicChange and EntityChange is almost 1 to 1 in this scenario. This is due to the way changes are captured and notified within the NeOn toolkit. The only atomicChange that didn't have a corresponding EntityChange was the declaration of individual, because within the NeOn toolkit a declaration of an individual is followed always by the assertion of this individual as a member of a class.

Finally to verify if the changes that could be made programmatically to the ontology were successfully executed, we opened the resulting ontology[3] within the NeOn toolkit. The conclusion was that all changes made (and that could be displayed by the GUI of the NeOn toolkit) to the ontology were successfully executed. Some changes such as adding the axiom asymetricObjectProperty are correctly captured in the OWL file but it is not possible to visualize them in the GUI at this moment.

[3]Available at `http://torresq.dia.fi.upm.es/neon/results/pizzaAll.owl`.

3.2 Experiments

Following the phases considered in most software experimentation approaches[BSH86],[Pfl95],[KPP$^+$02], the experiment was performed in the following three consecutive phases:

- Plan phase: describes the definition and design of the experiment (see 3.2.1)
- Experiment phase: describes the experiment execution (see 3.2.2)
- Analysis phase: describes the analysis of the experiment results (see 3.2.3)

3.2.1 Plan phase

3.2.1.1 Experiment Definition

Motivation
The main motivation of this experiment was to evaluate the models and strategies proposed for the management of ontology changes to support the development and maintenance of ontologies in a collaborative scenario.

Constraints
The infrastructure is constrained to be used by organizations with a defined process that coordinate the collaborative ontology development. The users, which are usually geographically distributed, are required to belong to the same organization i.e. they know each other and have associated permissions to perform some tasks based on their expertise and responsibilities. A representative organization for this scenario is FAO.

Goals
The goal of the experiment was to evaluate the benefits of using the infrastructure that implements the models and strategies described in this deliverable (and NeOn Deliverable 1.3.1) for the management of ontology changes to support the development and maintenance of ontologies collaboratively by a team of ontology editors with different roles and following a well-defined process for the proposal of ontology changes.

Beneficiaries of the experiment
Ontology editors at any organization following a well-defined process in the development and maintenance of ontologies.

Experiment subject
In particular, we evaluated the following items:

- The change representation model
- The workflow model
- The system implementation

3.2.1.2 Experiment Design

Relevant characteristics
We studied on the one hand the conceptual models that provide the foundations to represent the required information and on the other hand the implementation support. The following attributes were studied:

1. Conceptual Models
    - Change representation model:
        - The adequacy with respect to the users' requirements
    - Workflow model
        - The adequacy of the model with respect to the users' actions
2. System Implementation
    - The overall usability and performance of the system. According to the ISO standard 9241-11[ISO98], usability[4] refers to:
        - Effectiveness
        - Efficiency
        - User satisfaction

Metrics and criteria

- To measure the adequacy of the change representation model we analyzed the set of changes that ontology editors are usually required to perform. For every change proposed, we verified that it could be represented by our model and that it captured all the information required by the ontology editors.
- To measure the adequacy of the workflow model we analyzed if ontology editors were able to perform all of the required workflow actions. For each possible action, we verified that it could be represented with our model and that it captured all the information required by the ontology editors.
- To measure the effectiveness of the system we requested ontology editors to perform a set of tasks and we analyzed:
    - Percent of tasks completed
    - Ratio of successes to failures
    - Number of features or commands used
- To measure the efficiency of the system we used the same set of tasks as above and we determined:
    - Time to complete a task:
        - ∗ Change the ontology: Time required to perform all requested changes to the ontology taking into account the number of proposed changes.
        - ∗ Complete process: All the required actions (and proposed changes) to change one ontology from one stable version to another stable version (i.e. time to complete all tasks of the experiment) taking into account the number of proposed changes.
    - Time to learn
    - Percent or number of errors
    - Frequency of help or documentation use
- To measure the user satisfaction, we used as basis the Software Usability Measurement Inventory[5] (SUMI) which is a rigorously tested and proven method of measuring software quality from the end user's point of view. SUMI[vV98] studies the following 5 dimensions:

[4]" The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"

[5]`http://sumi.ucc.ie/index.html`

- Efficiency: measures the degree to which users feel that the software assists them in their work and is related to the concept of transparency.
- Affect: measures the user's general emotional reaction to the software (aka Likeability).
- Helpfulness: measures the degree to which the software is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation.
- Control: measures the extent to which the user feels in control of the software, as opposed to being controlled by the software, when carrying out the task.
- Learnability: measures the speed and facility with which the user feels that they have been able to master the system, or to learn how to use new features when necessary.

Hence, for this task we requested ontology editors to fulfill an online survey (e.g. at http://www.surveymonkey.com/) which included the de facto industry standard evaluation SUMI questionnaire consisting of 50 statements (10 for each dimension) to which the user had to reply either *Agree, Don't know, or Disagree*. The option (*Don't know*) is selected if the user is undecided, canŠt make up his mind, or if the statement has no relevance to the software or to the situation. Furthermore, in addition to the standard SUMI questions, the survey included 10 questions specific to the collaborative ontology development process to assess the user's satisfaction when using the infrastructure for this purpose.

Variables

1. Controlled
   - The experience and the background level of the ontology editors performing the experiments. Ontology editors were chosen from two major groups, Subject Experts and Validators. Subject experts know about specific aspects of the ontology domain and are in charge of adding or modifying ontology content. They usually know little or nothing about ontology software or design issues. Validators revise, approve or reject changes made by subject experts, and they are the only ones who can copy changes into the production environment for external availability. They have a broader knowledge of the ontology domain and have at least some knowledge about design issues.
2. Not controlled
   - The learning capabilities of the ontology editors could affect the result, but due to time and the ontology editors' availability restrictions, it was not used as a factor of the experiment.
   - Similarly, the motivation that ontology editors had to learn a new system can affect the result. In this case the effect was at leat reduced by an appropriate introduction of the system and the expected improvements.

Data collection process

To collect the required data regarding the adequacy of our models and the overall usability and performance of the system we conducted an experiment with a set of appropriate users. In particular, the experiment was carried out by a team of representative users from FAO, who carried out a series of representative tasks as follows:

- The team was provided with a stable version of a fishery ontology (e.g. v1).
- They were asked to perform collaboratively a set of typical changes and actions to the ontology in order to reach a new stable version (e.g. v2). The tasks included:

  - Add new ontology elements;
  - Modify ontology elements;
  - Delete ontology elements;
  - Approve proposed changes (including the previous required actions);
  - Reject proposed changes (including the previous required actions);

- Users were asked to visualize all the information recorded by the system for each of the performed changes and actions to confirm that all the required information is available.
- One person (the tester) was recording and documenting the time users take to accomplish the different tasks, the possible problems/errors and the users' comments during the process.
- At the end of the experiment, the editors were presented with a survey to evaluate their experience to perform the requested tasks.
- All the results were stored and documented by the tester person.

Requirements

At leat one complete execution of the experiment with a team of ontology editors working collaboratively on the maintenance of an ontology. The team was required to include at least one ontology editor for each possible role in the workflow i.e. one Subject Expert and one Validator. Additionally, the ontology used for the experiment was required to be one of the most frequently used fishery ontologies at FAO.

Analysis procedure

The collected data was validated to ensure that the ontology editors performed the required actions to reach the new stable version of the ontology (v2). The analysis of the data was performed with a combination of manual and automatic techniques. We analyzed if our models were adequate to represent the corresponding changes and actions based on the user's feedback and we computed the required ratios, percentages and times (e.g. time to complete a specific task) summarizing the individual results. We also used some of the automatic reporting tools (e.g. `http://www.surveymonkey.com/`) to analyze the results of the surveys about the user's satisfaction. We analyzed the possible reasons that affected those results and we compared them with the current time and efforts that takes to FAO ontology editors to accomplish the same task without the proposed infrastructure.

3.2.2 Experiment Phase

The experiment was conducted at FAO headquarters during the last week of October, 2008. We needed two days, one for the set-up of the collaborative infrastructure in FAO computers and one for running the experiment. Following the typical behavior of the ontology editors in FAO and because of time constraints, during the first day, the tester configured the collaborative infrastructure as the configuration A described in 2.2.2.1. The particular configuration was:

- One server running:
  - NeOn Collaboration Server
  - Oyster (server mode)
- Three clients, each running:
  - NeOn Toolkit extended with:
    - ∗ Registry Plugins

  * Change Management Plugin
  * Collaboration Plugins

For the experiment the following software versions were used:

Table 3.1: Software versions

Software	Version
NeOn Collaboration Server	OntoBroker-Enterprise-5.2-B719
NeOn Toolkit	NeOnToolkit-1.2.0-B766-extended
Oyster	Oyster2APIv2.3.1
Registry Plugins	org.neontoolkit.oyster.plugin.menu-1.8.0.jar org.neontoolkit.registry.api-2.3.0.jar
Change Management Plugins	org.neontoolkit.changelogging-1.8.0.jar
Collaboration Plugins	org.neontoolkit.collab.preference-1.8.0.jar org.neontoolkit.collab-1.8.0.jar

It is important to note that the version of the NeOn collaboration server used for the experiment has one critical limitation (that will be fixed soon): When a change is performed to an ontology managed by the server, no provenance information is available when the event is fired (i.e. it is not possible to know from which client the change was originally performed). Nevertheless, our infrastructure provides a temporary (but not perfect) solution to overcome this problem.

The ontology used for the experiment was species-v1.0-model.owl[6]. This ontology is the schema of one of the most important ontologies of FAO for the fishery domain. The tester uploaded this ontology to the server as part of the configuration.

On the second day, before running the actual experiment, the tester gave a brief introduction (30 min) of the system and the goal of the experiment to the FAO team composed of three ontology editors (Subject Expert A, Subject Expert B and Validator A). Each of the editors were in charge of maintaining the ontologies of the fishery domain and they had a different background profile:

- Subject Expert A had a great knowledge about the fishery domain but has never used the NeOn toolkit and in general he has a small knowledge of computer systems or modelling design issues.

- Subject Expert B had a fair knowledge about the fishery domain and the NeOn toolkit and some knowledge about modelling design issues.

- Validator A had also a fair knowledge about the fishery domain, the NeOn toolkit and also about modelling design issues.

All ontology editors were in the same room and each was provided with a detailed and personalized guide of the tasks he had to perform including the initialization of his NeOn Toolkit installation (i.e. each of them had to configure his client as in a real situation). The three complete guides are available at `http://torresq.dia.fi.upm.es/neon/guides`. In a nutshell, each subject expert (SE) had to perform 6 main tasks while the validator (V) had to perform 4 main tasks, as follows: every ontology editor was requested to configure and start the collaboration support within his NeOn toolkit (T1), then each subject expert was requested to make several changes to the ontology concurrently (SE's-T2), visualize the results of their changes and analyze the information provided by the system (SE's-T3) and submit their changes to be approved (SE's-T4). The chosen changes were 34 (17 changes for each SE) realistic modifications to the ontology including real information according to FAO experts. Examples of those changes are:

- Add Individual 31005-10001 (Species)

- Add Individual 31005-10000 DataProperty hasCodeAlpha3 value: DCR. Type: string

[6]Available at `http://www.fao.org/aims/neon.jsp`

- Add Individual 31005-10001 DataProperty hasNameScientific value: Pterodroma wrong macroptera. Type: string
- Add Root Class Speciation
- Add ObjectProperty hasScientificNameAuthor

Then the validator was requested to analyze the changes performed and to approve/reject them (V-T2). The subject experts were then requested to perform some additional actions according to the workflow to test the possible subject expert actions (e.g. delete a rejected change, modify an approved change, etc.) (SE's-T5 and T6). Finally the validator was requested also to perform some additional actions to test the possible validator actions (e.g. reject to be approved a change, delete an approved change, etc.) (V-T3 and T4).

As we explained in the previous section (see section 3.2.1.2), during the experiment the tester was taking note of the behavior of the editors, their questions and problems, and at the end of the experiment, each editor fulfilled an online survey consisting of 60 questions (50 of the standard SUMI questionnaire and 10 specific for the collaborative ontology development). The survey is available at `http://torresq.dia.fi.upm.es/neon/survey.htm` (see Figure 3.1). Here are some example statements:

- This software responds too slowly to inputs (SUMI)
- The way that system information is presented is clear and understandable (SUMI)
- I can understand and act on the information provided by this software (SUMI)
- The time to build an ontology collaboratively decreases with this software (Collaborative Ontology Development)
- The information captured for the ontology changes is enough (Collaborative Ontology Development)

The total time taken to complete the experiment was about two hours.

3.2.3 Analysis Phase

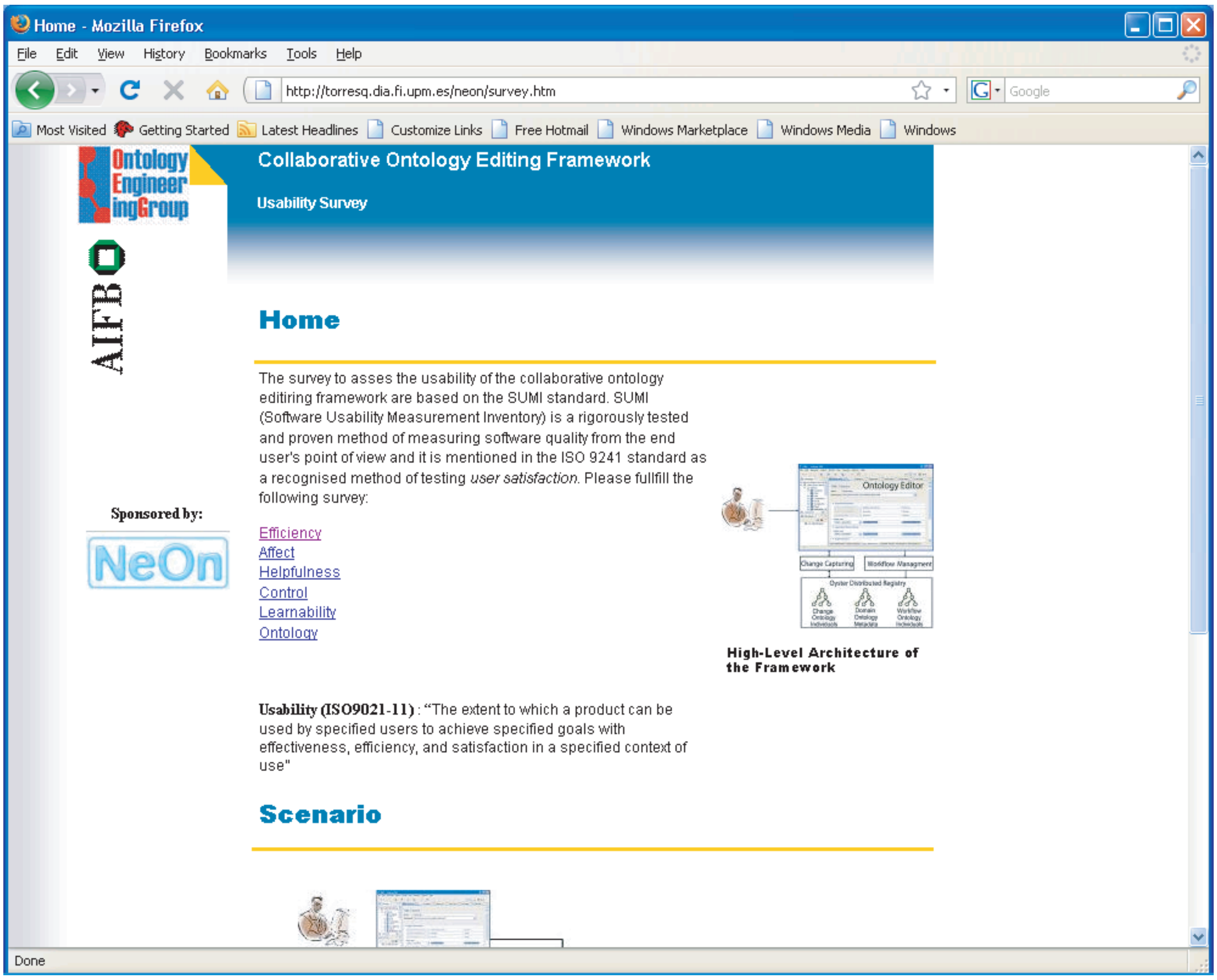
3.2.3.1 Adequacy of the change representation model with respect to the ontology editors needs

To measure the adequacy of the change representation model, we requested ontology editors of our case study at FAO to perform a set of representative changes and asked them to analyze the captured information for the change. The chosen set of changes was carefully selected with the cooperation of FAO experts to reflect valid modifications and using real data.

Based on those changes, we verified during the experiment that none of the ontology editors had problems to perform the changes to the ontology, and consequently that **all proposed changes could be captured by our representation model**.

Additionally, the survey that ontology editors had to fulfil after the experiment (see previous section) included two questions to verify the adequacy of the model from the ontology editors' perspective. From the three possible answers in the survey (i.e. *Agree, Don't know, Disagree*), we are mainly interested in the first and the third one. According to the three possible meanings of the option *Don't know* explained above (see 3.2.1.2), in this particular situation it could only mean that either the user is undecided or that he can't make up his mind, because the statement is relevant to the software and the situation. Consequently, if the user did not provide additional feedback or comments, we ignored it. In the following we analyze the results of the two statements:

- *Some changes were not captured correctly*. The result for this statement shows (see Figure 3.2) that nobody agreed with it, one person didn't know and the other two disagreed. Since, there was no additional feedback from the undecided user, we can affirm that **all proposed changes could be captured by our representation model**, since nobody disagreed.


Home - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://torresq.dia.fi.upm.es/neon/survey.htm

Most Visited Getting Started Latest Headlines Customize Links Free Hotmail Windows Marketplace Windows Media Windows

Ontology Engineering Group

AIFB

Collaborative Ontology Editing Framework

Usability Survey

Home

The survey to asses the usability of the collaborative ontology editiring framework are based on the SUMI standard. SUMI (Software Usability Measurement Inventory) is a rigorously tested and proven method of measuring software quality from the end user's point of view and it is mentioned in the ISO 9241 standard as a recognised method of testing *user satisfaction*. Please fullfill the following survey:

Efficiency
Affect
Helpfulness
Control
Learnability
Ontology

High-Level Architecture of the Framework

Sponsored by:

NeOn

Usability (ISO9021-11) : "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"

Scenario

Done

Figure 3.1: Online Survey

- *The information captured for the ontology changes is enough*. In this case, the result for the statement shows (see Figure 3.3) that one person agreed, one didn't know and one disagreed. Similar to the previous statement, the undecided user didn't give any additional feedback and so, we concentrate in the negative answer. For this survey item, we explicitly requested users to provide some feedback in the case they disagreed. The comments from the user were the following (textually):
  - "Would be nice to have some formatting, it's rather hard to read and takes a bit to figure out what the change actually was".
  - "Would be nice if you could click on a change and go to the relevant part of the ontology".
  - "The Change log view lacks a column present in the other views that tells you the concept or property associated to the change".

  As we can see from the comments, the user was providing feedback regarding the visualization of the changes rather than the information captured for the changes (i.e. the author, the date and time, the related entity, etc.). Therefore, as we don't have any request for additional information that should be captured for the changes, we can argue that **the information captured for the ontology changes is enough**.

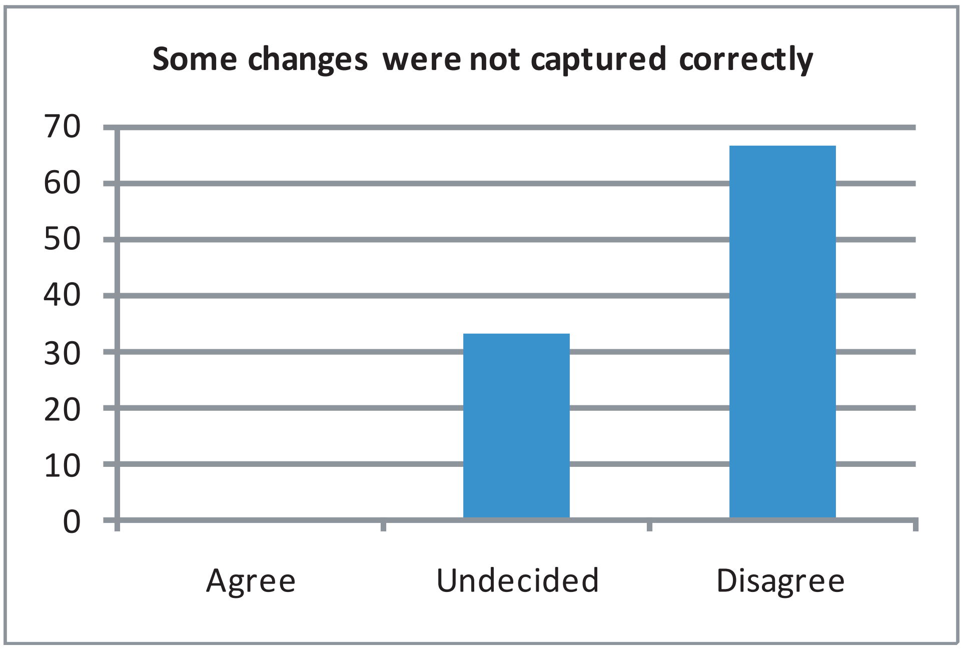


Figure 3.2: Survey Question 6-4

3.2.3.2 Adequacy of the workflow model with respect to the ontology editors' needs

Similarly to the previous metric, to measure the adequacy of the workflow model, we requested that the ontology editors of our case study at FAO perform every possible action according to their role, and asked them to verify that their actions were correctly executed i.e. to see the consequences of their actions (e.g. when a change draft state is sent to be validated, it should no longer appear in the list of draft changes) which in turn would confirm that the action captured all the required information.

During the experiment, we verified that none of the ontology editors had problems performing the set of possible actions according to his role and consequently that **all workflow actions could be captured and represented by our workflow model**.

Also like in the previous metric, the survey that ontology editors had to fulfil after the experiment included two questions to verify the adequacy of the model from the ontology editors' perspective. For the same reasons explained above, we focus our attention to the *Agree* or *Disagree* answers, and ignore the *Don't know* answer if the user did not provide additional feedback or comments. In the following we analyze the results of the two statements:

- *The software allows to perform all the required actions of our workflow.* The result for this statement shows (see Figure 3.4) that everyone agreed with it. Hence, we can affirm that **all workflow actions could be captured and represented by our workflow model**. Furthermore, if all ontology editors could perform all the required actions of the workflow, means that they could also verify the expected consequences of the actions. So, we can affirm that **actions captured all the required information**.

- *Users are able to perform all the actions they could perform according to their role and only those.* In this case, the result for the statement shows (see Figure 3.5) that nobody disagreed with it, one person didn't know and the other two agreed. Since, there was no additional feedback from the undecided user, we can affirm that **all workflow actions could be captured and represented by our representation model**, since nobody disagreed.

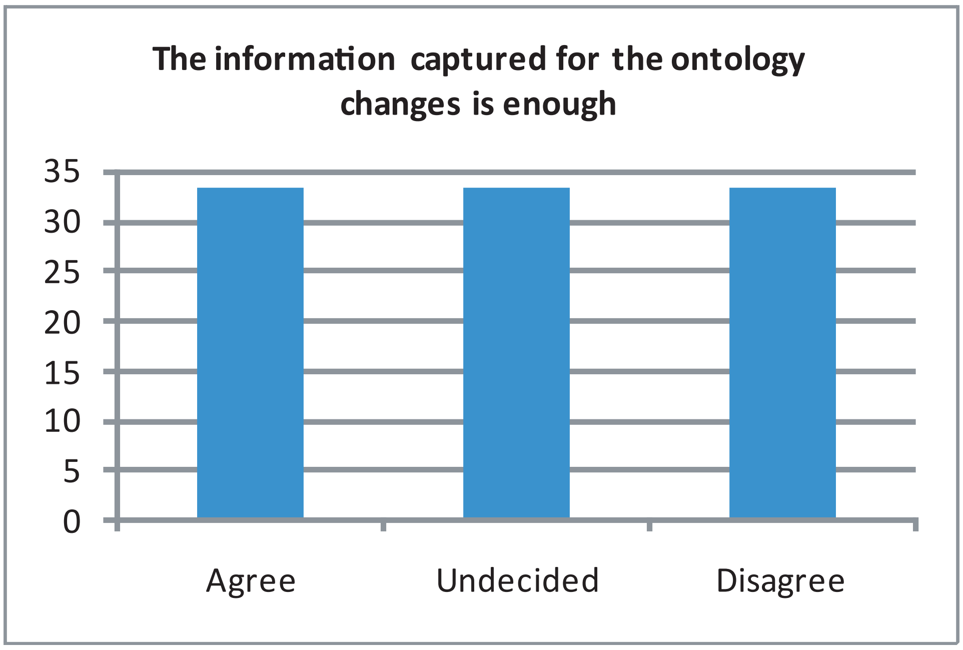


Figure 3.3: Survey Question 6-5

3.2.3.3 Effectiveness of the system

The experiment gave the following results for the effectiveness of the system:

Percent of tasks completed
Each ontology editor completed 100% of his assigned tasks (6 tasks for each subject expert and 4 for the validator). However after analyzing the log of changes of the experiment, it reveals that for the task 2 of the subject experts that consisted in making 17 changes to the ontology, one of the subject experts was author of only 13 of the required changes (76% individually or 92% globally). The other 4 changes were not present in the log, so either the user forgot to made them or there was a failure in the system. Since there was no report or complain during the experiment that a change was not created, we we assume there was no system failure. Furthermore, we found 13 changes that were not part of the guide. After analyzing each of them, we concluded that:

- 5 changes were performed to provide additional information or to test other possible changes.
- 5 changes were consequence of a human error (e.g. one SE created a subClass instead of a root class, some individuals were removed and then added again)
- 3 changes were incorrectly generated as a consequence of the limitation of the collaboration server described in the previous section.

Ratio of successes to failures
For this measure we analyzed the following two functionalities:

- Change logging: We consider it a success if a change performed was correctly captured, represented and registered, and a failure otherwise. We analyzed the 30 logged changes (from the guide) (17 from SE1 and 13 from SE2) and we verified that all of them were successfully logged. However, 3 of them were incorrectly logged twice due to the NeOn collaboration server limitation. Therefore, we have a 90% rate of success.

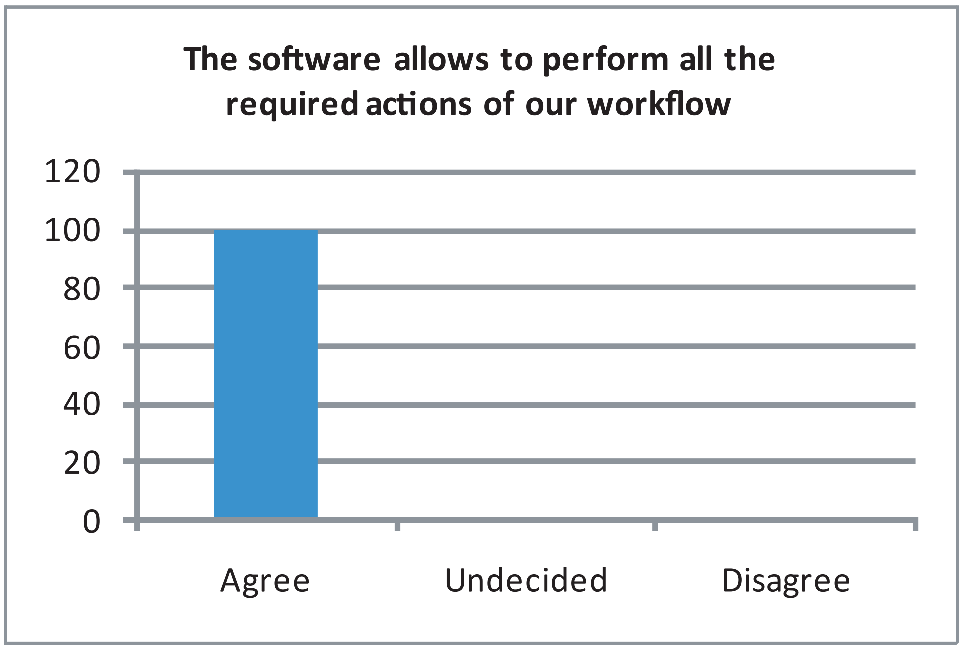


Figure 3.4: Survey Question 6-2

- Workflow management: We consider it a success if a workflow action (i) was correctly created, represented and registered and (ii) had the expected consequences (e.g. adding an ontology element, should create an *insert* workflow action and set to *draft* the element state). As we explained in section 2.1.0.2, actions are performed either implicitly (e.g. add element) or explicitly (e.g. submit change to be approved). During the experiment, SE1 performed 17 implicit actions and 20 explicit ones (from the guide), SE2 performed 13 implicit actions and 16 explicit ones (from the guide) and V1 performed 7 explicit actions. The total number of actions was 73. We analyzed each of them and verified that they were a success, giving a 100% rate of success.

Number of features or commands used

During the experiment ontology editors used the following 18 features or commands (classified according to the infrastructure components):

- Registry
  - Registry support
  - Remote storage location
- Change logging
  - Log changes
- Workflow management
  - Workflow actions (9): Insert, Update, Delete, Submit to Be Approved, Submit to Approved, Submit to Be Deleted, Reject to Approve, Reject to Be Approved, Reject to Draft.
  - Enforcement of workflow constraints
- Visualization components
  - Change log view

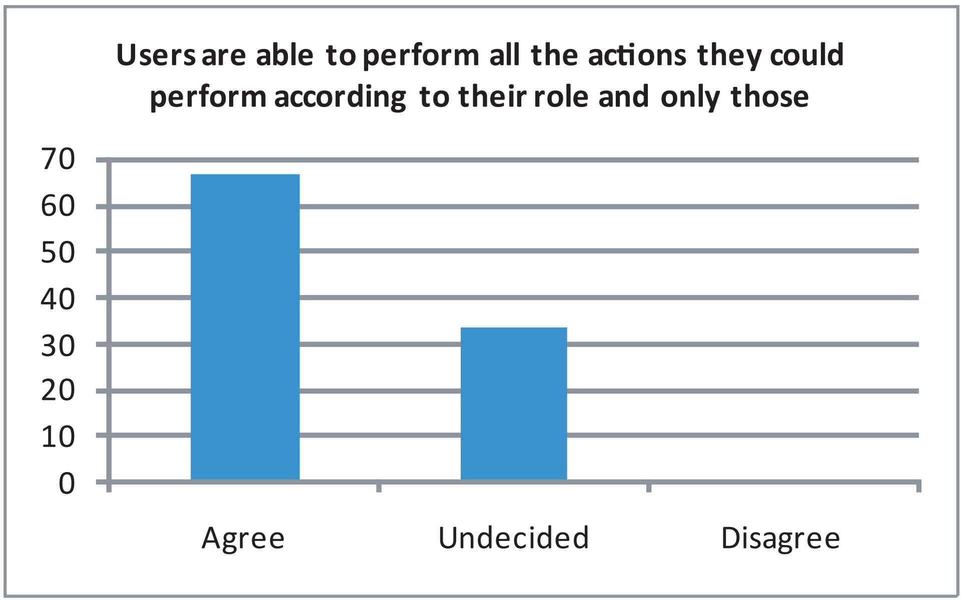


Figure 3.5: Survey Question 6-3

  - Draft view
  - To Be Approved view
  - Approved view
  - To Be Deleted view

Only the following two features of the infrastructure were not tested during the experiment (2 out of 20, or 10%) due to the configuration of our scenario (configuration A described in 2.2.2):

- Registry
  - Synchronization of changes
- Change logging
  - Propagation of changes to local ontology

3.2.3.4 Efficiency of the system

In the following we present the results of the times analyzed based on the log of changes, the information in the registry and the notes taken during the experiment.

Time to complete a task

As we explained in section 3.2.1.2, we analyzed two tasks (i.e. changing the ontology and the complete process). The results are:

- Change the ontology: The time required by both SEs to perform 40 changes concurrently (30 from the guide plus 10 they wished to test or had to repeat after they made a mistake) took 47 minutes and 47 seconds (47:47).
- Complete process: The time required by all ontology editors to complete the experiment i.e change the ontology, send changes to be approved, approve/reject changes and all additional workflow actions from the guide, was 1 hour and 50 minutes (1:50:03). During the experiment, the NeOn collaboration server crashed once and it took around 5 minutes to restart it since it was in a remote location.

Time to learn

As we described in the previous section, before starting the experiment we gave a brief introduction (30 minutes) of the system and the goal of the experiment to the FAO team. Then, during the experiment, users asked a few times for assistance when they had a problem or wanted to know additional information.

Percent or number of errors

During the experiment, we did not get any errors from the infrastructure. However, we got the following problems:

- The NeOn Collaboration server crashed once. After analyzing the log of the server, we found out that the problem was that the server was out of memory.
- After the server crashed, the SEs had to repeat the last change they were performing when it crashed.
- One of the SEs had to repeat his first 5 changes because they were not logged at all. In this case, the SE didn't configure correctly his NeOn Toolkit and had to restart it.

Frequency of help or documentation use

Throughout the experiment, ontology editors asked regularly for assistance of the tester. However, in general they wanted either to ask how to do something on the ontology editor (i.e. which is provided by the NeOn toolkit not by us), or to provide some feedback. In general, they didn't have problems using the features of the infrastructure. In any case, taking into account that users had only a brief introduction to the collaborative infrastructure (and the experiment), in addition to the fact that they did not use the NeOn toolkit regularly, it is totally understandable that they needed some help during the experiment.

3.2.3.5 User satisfaction

In the following we analyze the results of the 60 questions of the survey filled by the ontology editors at the end of the experiment. First, we focus on the 10 questions specific to the collaborative ontology development and then we present the results for the 5 SUMI dimensions, consisting of 10 questions each.

Collaborative Ontology Development

Out of the 10 questions of the survey specific for this matter, 8 required the user to reply either *Agree, Don't know, or Disagree*. The last 2 questions requested written information from the user: the best and worst part of the infrastructure and any final feedback/comments.

The general impression of the infrastructure to support the collaborative ontology development process was calculated as the average of the first 8 questions and it is shown in Figure 3.6.

As we can see from the figure, the impression of the users was around 63% positive, around 29% was undecided and only around 8% was negative. The 29% reflects the cases when the user is undecided or when he can't make up his mind and, as we explained above, if the user does not provide additional feedback or comments we ignore it. After analyzing the feedback received from the users on the negative answers and undecided answers when available (see the detailed analysis of each question below), we found that in general they refer only to desired improvements in the GUI to facilitate or make some tasks more intuitive. Nevertheless, feedback also showed that users were in general highly satisfied with the infrastructure and they agreed on its usefulness and correctness. Hence, taking into account that this is the first implementation (at the best of our knowledge) of a complete infrastructure that addresses the collaborative ontology development in organizations with a well-defined edition process, **we claim that the results are highly encouraging and motivational. In particular, the results provide an indication of the real value and practical usability of the models and methods proposed in this work. Nevertheless, we need additional experiments and more users to draw full conclusions**.

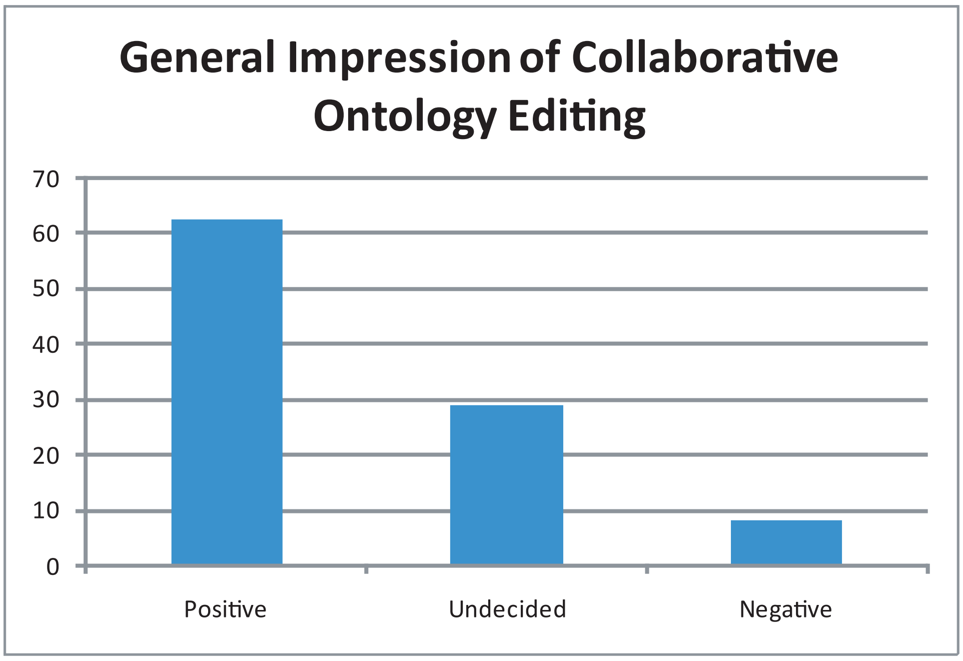


Figure 3.6: Collaborative Ontology Development Survey

In the rest of this section, we analyze the individual questions grouped by their relationship. Note that questions 2 to 5 were already analyzed in the previous sections.

Statements related to the usability and helpfulness of the infrastructure

- *The time to build an ontology collaboratively decreases with this software.* The result for this statement shows (see Figure 3.7) that everybody agreed with it. Therefore, we can affirm in a straightforward manner that **users agree that the infrastructure supports and improves the time required to develop ontologies collaboratively**.
- *I prefer to use this software when developing an ontology collaboratively rather than the previous approach.* In this case, the result for the statement shows (see Figure 3.8) that nobody disagreed with it, one person didn't know and the other two agreed. Since, there was no additional feedback from the undecided user, we can conclude that **users prefer to use the presented infrastructure to develop ontologies collaboratively rather than any previous approach**.

Statements related to the user interfaces of the infrastructure

- *The information shown in the workflow interfaces is what I expected.* The result for this statement shows (see Figure 3.9) that one person disagreed with it and the other two didn't know. As we can see, this was the only case when the answer was mainly negative. However, since users were requested explicitly to provide feedback on this question, we found out that in general they didn't think that something was wrong, but that they desire additional features to make the interface more intuitive. The comments from the users were the following (textually):
  - "For validators, it would be useful a more compact way to summarize changes made by subject experts"
  - "The GUI is not enough intuitive, there should be the possibility of sorting by author and by time"

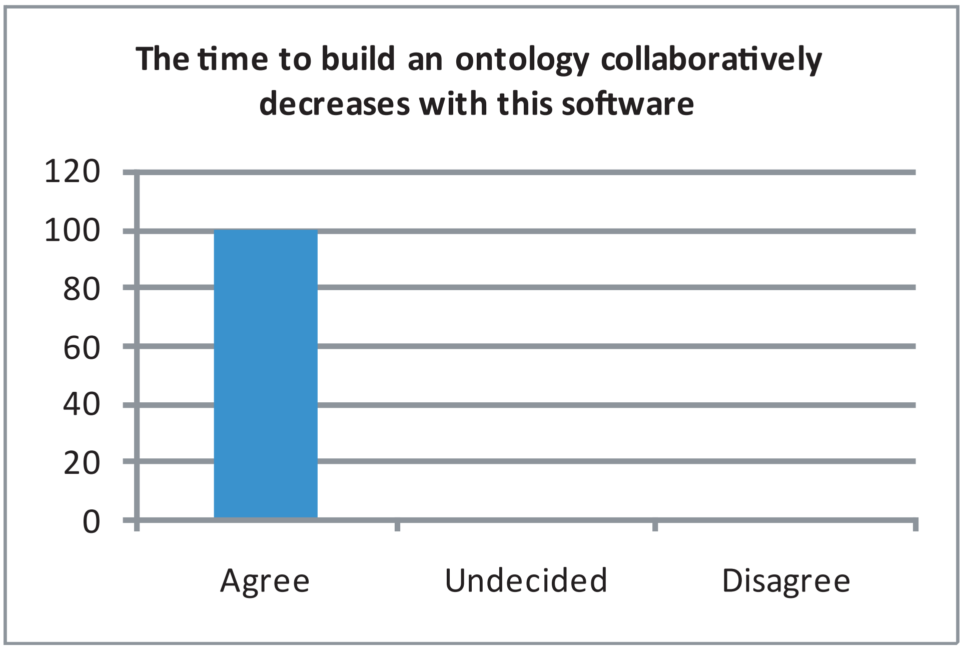


Figure 3.7: Survey Question 6-1

- "Would be nice if the view could be grouped around concepts to help understand the changes more easily, maybe a tree view"

Hence, we can learn from the previous comments that the workflow interfaces can be improved still in order to satisfy users. Nevertheless, the feedback was critical as it is the exact kind of information we needed from the actual users that couldn't be anticipated in our first implementation. Moreover, most of these desired improvements are simple modifications easy to implement. So, we can conclude that **although the workflow interfaces can be improved they provide a good and correct starting point.**

- *The software takes the user role correctly into account when displaying information.* In this case, the result for the statement shows (see Figure 3.10) that nobody disagreed with it, one person didn't know and the other two agreed. Since, there was no additional feedback from the undecided user, we can conclude that **the workflow interfaces take correctly into account the user role when displaying information**.

The best and worst thing of the software

The comments from the users were the following (textually):

- Best
  - "Seeing changes of others"
  - "A unified view of everything happening in the workflow"
  - "Great capability and real time update"
- Worst
  - "Log messages hard to read"
  - "Presentation of list of changes can be improved. No connection between graphical visualization of the ontology and changes made/proposed."

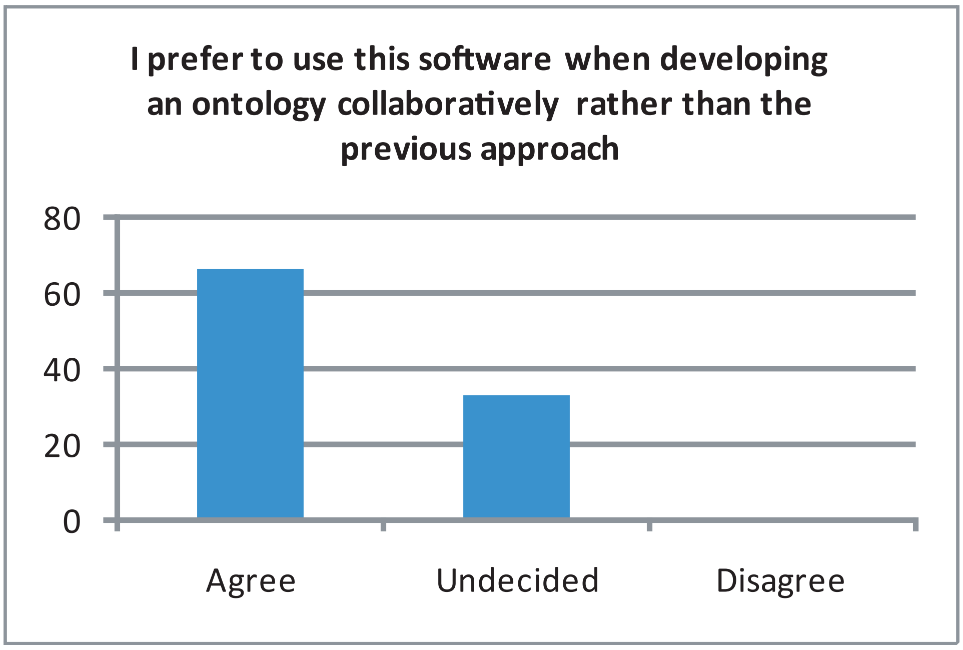


Figure 3.8: Survey Question 6-8

  - "Slow, difficult to add class and instance"

Note that the comments regarding the best things refer to more fundamental matters than the worst things i.e. our main goal is to provide a practical and usable infrastructure that offers a unified view of everything that is happening in the workflow, supporting users in the collaborative ontology development. This relies on the management of changes in distributed environments where users can access and see changes from every editor. Furthermore, the worst things are not criticizing or in disagreement with the goal of the infrastructure. Therefore, from the previous comments, we can conclude that **although some things can be improved (in particular in the visualization), in general users are satisfied and find useful the whole infrastructure for the development of ontologies collaboratively in a real scenario.**

Additional feedback

Some of the final comments received from the users were (textually):

- "Grouping of changes (by: change type, user, related entity etc.) would be useful"
- "An hour glass should be shown when the application is taking time to update a data submission or whatever action..."
- "It should be possible to sort by author and time and by default it should by descending time"
- "A button saying "submit all" may help"
- "Really useful would be to be able to view the ontology and have the changed areas highlighted in some way together with their current change state"
- "When first opening a view would be nice if it automatically refreshed itself"
- "...when going into the workflow a wizard could lead you through the various steps that involve accessing several different functional areas of the toolkit"

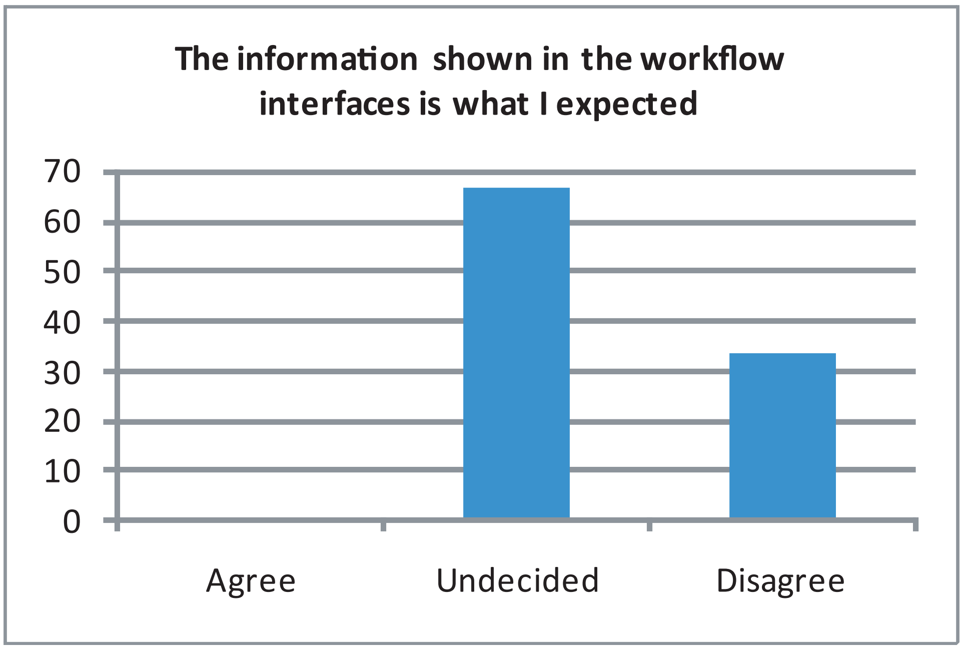


Figure 3.9: Survey Question 6-6

- "When adding an instance, since there is a lag time as the ontology is being accessed remotely, the instance label shows a strange string until updated which is disconcerting"

Similar to the previous question, all these comments (except the last two) are desired improvements to the interfaces to make it more user friendly. The penultimate item, which suggests the creation of a wizard to guide editors when using the NeOn toolkit functionalities, would require not only support in our infrastructure but from the toolkit itself. Finally, the last item refers to a small issue with the NeOn collaboration server that we discuss in the analysis of the efficiency dimension.

Efficiency

The global efficiency was calculated as the average of the 10 SUMI questions for this dimension. According to the definition of efficiency in SUMI[vV98], users felt in 50% of the times that the software assisted them in their work (aka. transparency), in around 23% they were undecided and in around 27% of the times they disagreed (see Figure 3.11). After analyzing each of the 10 questions for this dimension, we found out that the following two contributed in particular to the 27% of disagreement:

- *"The software has at some time stopped unexpectedly".* As was explained above, the NeOn collaboration server crashed once during the experiment. However, after analyzing the server log, we learned that the reason was that the process ran out of memory. Additionally, although we decided to use the NeOn collaboration server for this scenario, it is not part of the implementation of our work in T1.3, and hence we consider it an external component for this evaluation.

- *"This software responds too slowly to inputs".* In this case, it is true that sometimes the interaction with the NeOn collaboration server was slow. In fact, the last comment of the previous section refers to this issue. However, as we explained in the previous paragraph, the server was not running with the required memory (that was the reason it crashed). Consequently, it is very likely that the interaction with the server can be improved by assigning additional memory to the server process. In any case, this is still an issue related to an external component for this evaluation.

From the previous analysis, we can conclude that most of the 27% of disagreement was caused by an external component of the infrastructure. Moreover, the result consists of 50% of positive answers and

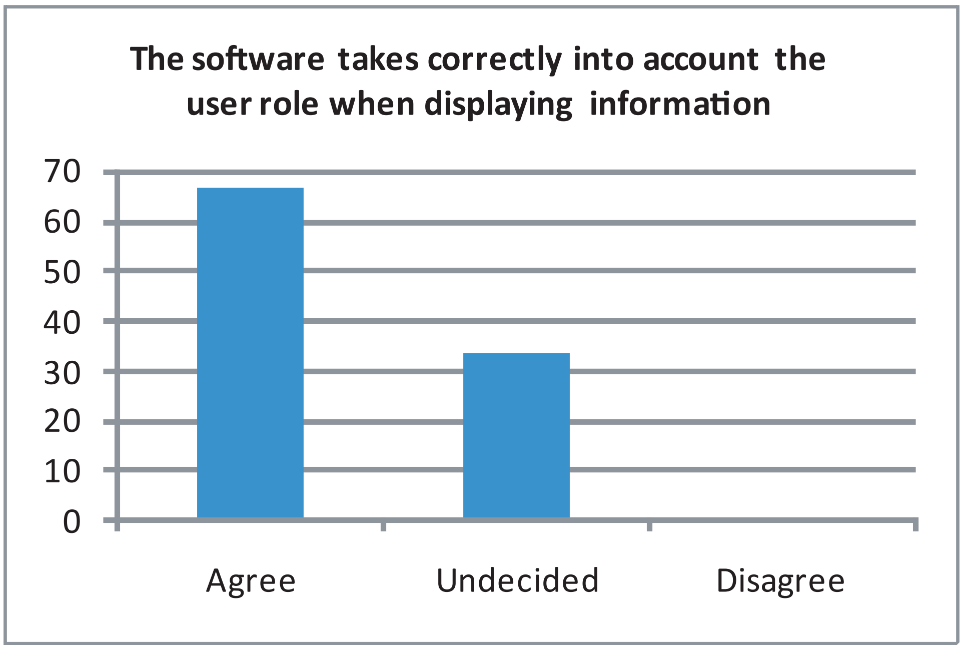


Figure 3.10: Survey Question 6-7

only about one fourth of the total (27%) were negative answers. Therefore, we consider the evaluation on efficiency satisfactory, as **users didn't feel that the software does not help them in their work**.

Affect

The global Affect was calculated as the average of the 10 SUMI questions for this dimension. As we can see from Figure 3.12, users had a positive emotional reaction to the software (i.e. they liked it) in around 63% of the cases, they didn't know in around 27% and only in 10% of the cases they had a negative reaction. After analyzing each of the 10 questions for this dimension, we found out that one of the few questions with negative answers was the following:

- *The way that system information is presented is clear and understandable.* As we can see this is a question related to the visualization of the information and as we already analyzed previously, this was one of the aspects where users gave some comments on how to improve it.

Moreover, from the others questions for this dimension, the results show that users found the software *satisfying* and *stimulating*. Therefore, we can conclude that **in general users liked the infrastructure**.

Helpfulness

The global Helpfulness was calculated as the average of the 10 SUMI questions for this dimension. In this case, as we can see in Figure 3.13, users agreed 50% of the time that the software is self-explanatory (including things like adequate help facilities and documentation), 30% of the time they were not sure and only 20% of the time they disagreed. We analyzed each of the 10 questions for this dimension, and we found that the only question where most of the answers were negative (2 out of 3) was the following:

- *The speed of this software is fast enough.* In this case, like it was explained above it is true that sometimes it was slow the interaction with the NeOn collaboration server, but this is an issue external to our implementation and therefore it is not part of this evaluation.

Furthermore, from the other questions for this dimension, we found out that all of the users considered that the *software was consistent* and that they could *understand and act on the information provided by the*

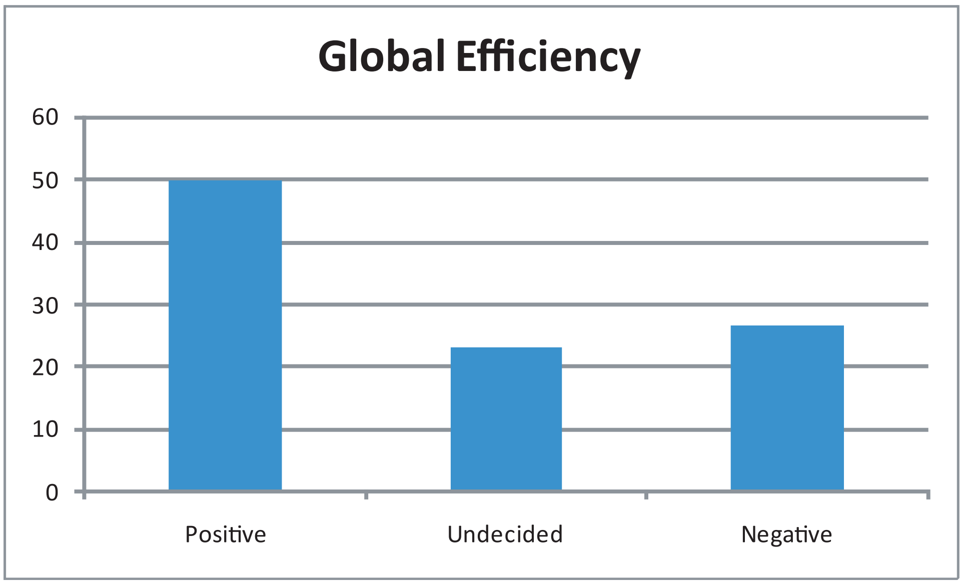


Figure 3.11: Global Efficiency

software. So, from the previous analysis and having 80% of non-negative answers, we can conclude that **in general users find the software self-explanatory**.

Control

The global Control was calculated as the average of the 10 SUMI questions for this dimension. Figure 3.14 shows that users felt in control of the software in 50% of the times, in around 27% they were not sure and in around 23% of the times they disagreed. After analyzing each of the 10 questions for this dimension, we found out that the only questions where most of the answers were negative (2 out of 3) are the following:

- *There have been times in using this software when I have felt quite tense.* In this case, taking into account that it was the first time for the users using the infrastructure after a brief introduction of 30 minutes and that they knew they were being monitored to performed several tasks, it is understandable they felt tense.
- *Error prevention messages are not adequate.* This issue is something that can be improved in the interfaces. For instance, when the NeOn collaboration server crashed, user didn't get any message preventing them from continuing their work because the server was down. So, similar to our previous analysis, this means that some aspects of the visualization can be improved.

Nevertheless, none of the users thought they would never *learn to use all that is offered in this software*. In fact most of them thought that *the organization of the menus or information lists seems quite logical* and that *the software allows the user to be economic of keystrokes*. So, similar to the previous dimension, based on the previous analysis and having more than 75% of non-negative answers, we can conclude that in general **users didn't feel as being controlled by the software, when carrying out the task**.

Learnability

The global Learnability was calculated as the average of the 10 SUMI questions for this dimension. As we can see from Figure 3.15, 50% of the time, the users felt positive with the speed and facility needed to master the system or to learn new features, around 27% of the time they were not sure and around 23% of the time they disagreed. After analyzing each of the 10 questions for this dimension, we found out that none of the questions had a majority of negative answers (at least 2 out of 3). In contrast, most of the users

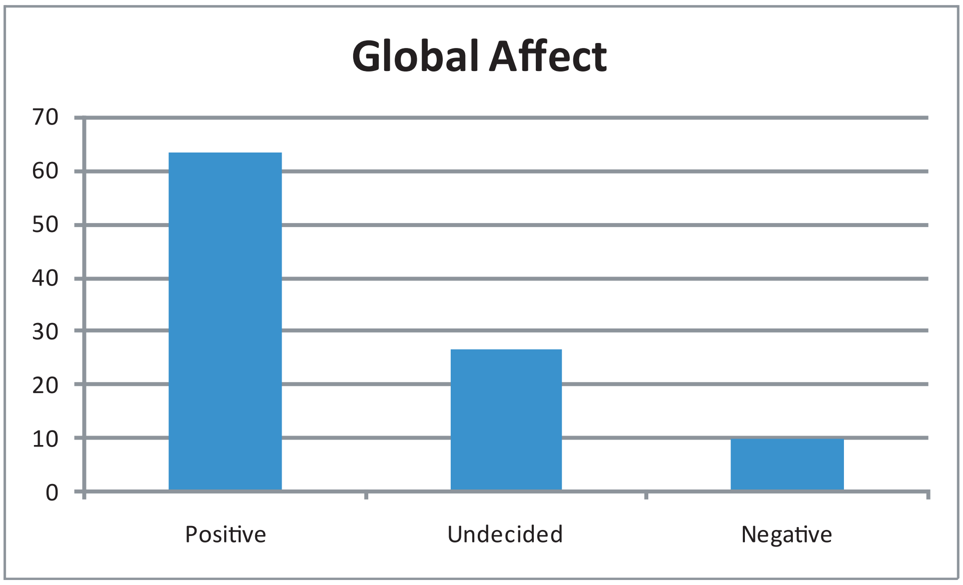


Figure 3.12: Global Affect

considered that *most times when they use this software, they didn't have to look for assistance*, that *this software behaves in an understandable way* and that *the software behaved as expected* (to mention a few).

Based on the previous analysis, and taking into account the more than 75% of non-negative answers, we can conclude that in general **users didn't think that it is difficult to master the system or to learn new features**.

3.3 Evaluation Summary and Recommendations

As a general conclusion we can say that the results of the evaluation are very positive. First, we showed that our layered approach for the representation of changes can be easily instantiated for a specific ontology language (e.g. OWL) and we proved the completeness of the change ontology extension with respect to the OWL 2 ontology language i.e. every change was successfully represented in our model.

Second, the analysis of the results of the experiment that we conducted at FAO shows several good points of the infrastructure as well as some issues that could be improved as part of our future work. In particular, the results showed:

- Our models (change representation, workflow model) are adequate with respect to the ontology editors needs. That is, representative changes and workflow operations from our use case could be captured and represented correctly by our models along with their required information.

- The overall system effectiveness was positive (90% or above) which demonstrates the good capability of our infrastructure to produce the overall goal i.e. collaborative ontology development.

- The efficiency of the system was in general satisfactory. A very positive point is that the time users required to complete their tasks was better than with their previous approach (see results of question 1 of the part of the survey specific to collaborative ontology development). Regarding the frequency of help use, it is understandable that users asked frequently for assistance, taking into account that they had only a brief introduction to the collaborative infrastructure (and the experiment) (30 min), in addition to the fact that they did not use the NeOn toolkit regularly. Finally, as most of the problems we found during the experiment were related to the NeOn Collaboration server, which is not part of this

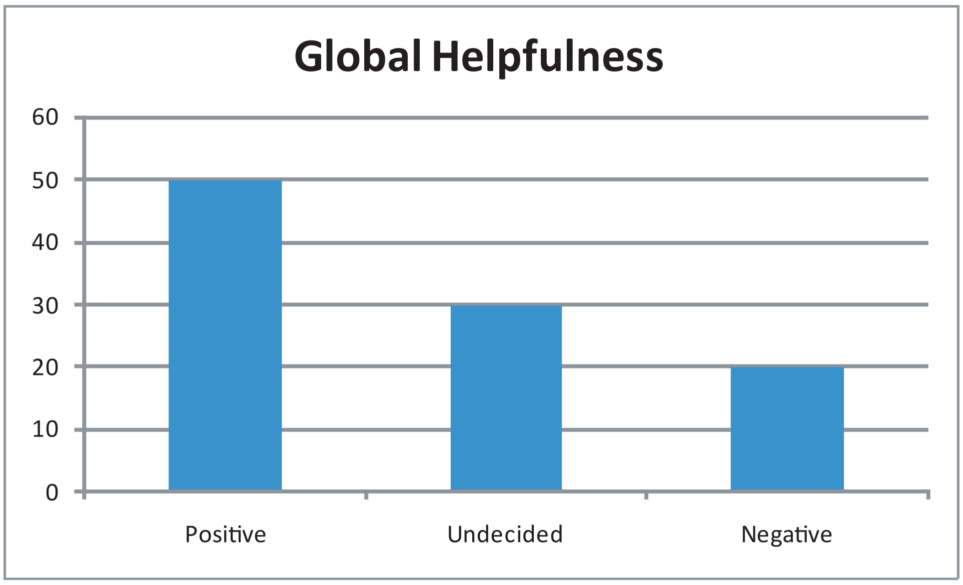


Figure 3.13: Global Helpfulness

work, we feel satisfied with the results. Note that the problem related to the server crash was just a lack of memory of the process which can be easily fixed.

Finally, the results of the survey to measure the user satisfaction showed that users were in general highly satisfied with the infrastructure and they agreed on its usefulness and correctness. For instance, the questions of the survey that evaluated the editors's satisfaction regarding the collaborative ontology development, show that editors think our infrastructure is better than the previous approach i.e. it is faster and they prefer it. Moreover, ontology editors actually liked the main features of the system (e.g. the integrated view of the workflow, the management of changes in a collaborative environment, etc.) as we can see from the feedback received in the textual answers. Nevertheless, the feedback received also shows some aspects that can be improved. In general those aspects are related to improvements in the user interfaces (which are very useful for our future work) such as being able to select multiple changes in one click, being able to sort the changes according to different criteria, or improving the readability of the change information. We also received a few comments regarding the NeOn collaboration server such as speed issues when performing some operations (e.g. adding individuals), or the problem when the server crashed. However, although we used the NeOn collaboration server in this scenario, it is not part of the work we are evaluating.

The overall results for each of the five SUMI dimensions have a similar pattern. In all cases, only around one fourth (25%) of the total answers were negative, another 25% (approximately) were undecided and at least half of the answers (50%) were always positive. So, in general we concluded that the results were fairly satisfying, specially if we take into account that users had only a brief introduction to the system before the experiment and none of them had much experience with the NeOn toolkit. The results show that in general users liked the infrastructure and that they find it self-explanatory. Furthermore, users didn't feel that the software does not help them in their work, or that they are being controlled by the software when carrying out a task or it is difficult to master the system (i.e. learn new features).

During the experiment and as part of the analysis of the results, we learned important lessons from which we can get some recommendations. For instance, we found out that sometimes users were interested to see only specific changes (e.g. from specific users, from specific type, etc.), in specific order or grouped by some criteria, instead of having the complete history of changes in chronological order (as it is at this moment). Another interesting observation is that users wanted to have a quick view of the changes related to a specific ontology element instead of having again a complete list of changes. Also, we could observe that users can easily get doubtful (i.e. try to repeat the action) whenever there is a small delay with the communication with

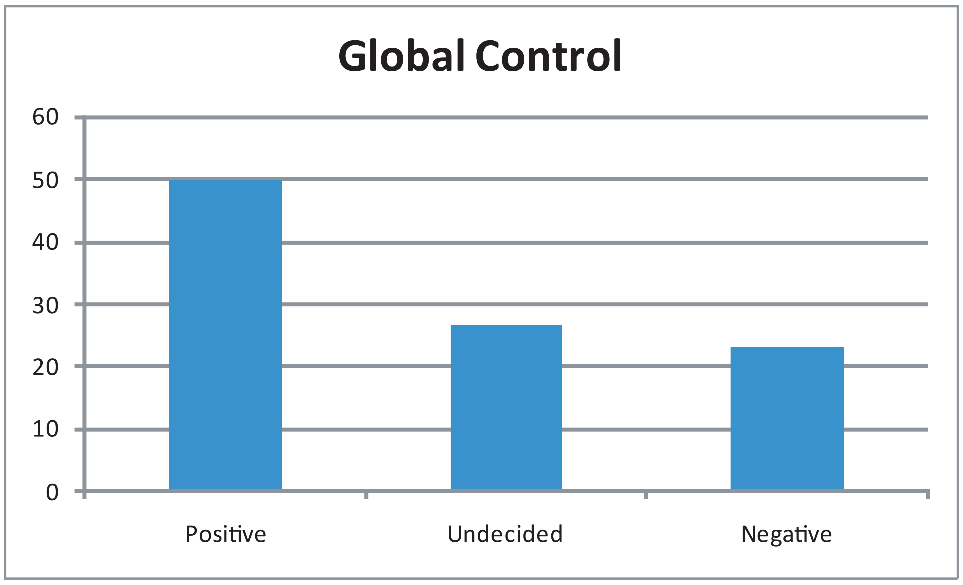


Figure 3.14: Global Control

the server. From these (and other) observations we got some recommendations to improve our infrastructure, specifically at the GUI level. First we should improve our views with additional features such as: sorting, grouping and filtering. Second, we should also add additional user-friendly features to our interfaces such as the ability to select several changes in one click or refreshing automatically the views when opening them. Third, we should provide a tighter link between the ontology navigator and the information displayed in our views. Finally, in the case of the NeOn collaboration server, as it is an external component of our infrastructure, we learned that we should provide additional resources (memory) to avoid any crash and to improve the speed of the communications.

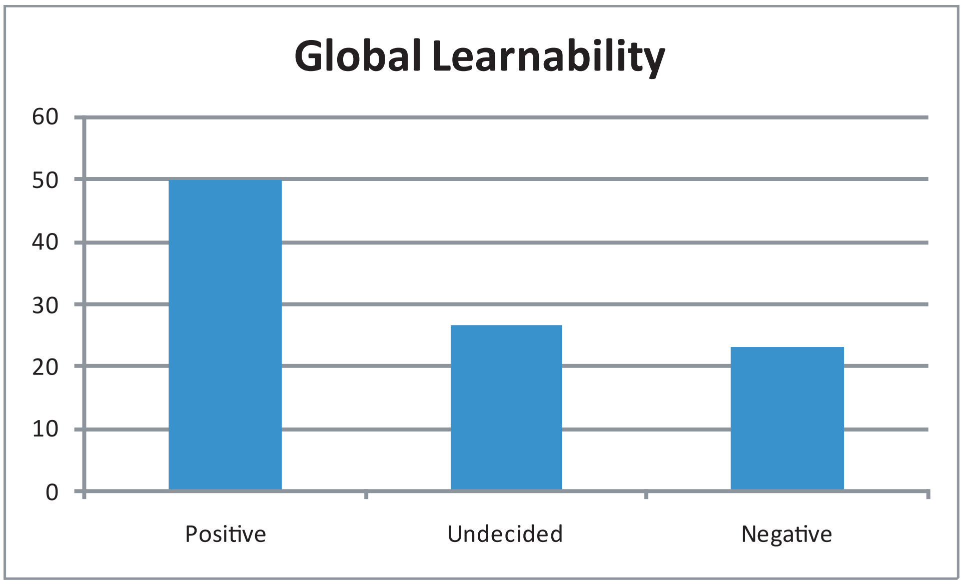


Figure 3.15: Global Learnability

Chapter 4

Conclusions and Outlook

4.1 Summary

In this deliverable, we presented our implementation of the methods and strategies for propagating networked ontologies that we proposed in NeOn deliverable D1.3.1 in order to support the collaborative ontology development based on editorial workflows. In particular, we presented:

- the updated conceptual models, originally presented in D1.3.1, that provide the foundations to represent the required information in our solution, including:
  - A brief overview of the Change representation model we introduced in D1.3.1 and an extended description of the OWL Change Ontology Extension.
  - A brief overview of the updated workflow model and the corresponding workflow ontology.
- the implementation support of our models and strategies as components of the NeOn toolkit. We provided the description of the architecture of our infrastructure, which includes:
  - Change Capturing Components
  - Workflow Management Components
  - Ontology Editing and Visualization Components
  - Distributed Registry
- the possible scenarios and configurations of the infrastructure. We show how the different components can be configured to support different organizational requirements.

It is interesting to note that although there exists already some methodological and technological support for the collaborative ontology development, our infrastructure is (to the best of our knowledge) the first complete implementation that addresses the collaborative ontology development in organizations with a well-defined edition process. Moreover, from all the possible scenarios in which our infrastructure can be used, only one has been considered up to now in the literature and tools (configuration A in 2.2.2.1). There is no support for the development of ontology collaboratively in a complete distributed environments.

We also presented a detailed description of the evaluation of our work. First, we tested the completeness of our representation model with respect to the OWL 2 ontology language. Second we described the experiment that we conducted at FAO following the phases considered in most software experimentation approaches. Hence, we explained the plan of our experiment (the definition and the design) and how we executed it. In a nutshell, a team of representative ontology editors performed a set of representative actions and operations collaboratively on one of the FAO ontologies from the fishery domain. During the experiment, one person was taking notes of the events and feedback received from the editors, and after the experiment editors had to fulfil a survey, to measure the user's satisfaction, which consisted of 60 statements : 50 from the standard

SUMI questionnaire (10 for each SUMI dimension) and 10 specific statements for the collaborative ontology development experience.

Finally, we presented the analysis of the results of the experiment. The general conclusion we got of the evaluation is very positive. The results are highly encouraging and motivational. In particular, the results provide an indication of the real value and practical usability of the models and methods proposed in this work. Nevertheless, we need additional experiments and more users to draw full conclusions. Moreover, we also learned some points that can be improved and that we hope to address in the future.

4.2 Future Work

There are still some challenges that we have to address in the future. First we need to improve some aspects of our infrastructure according to the results of the experiment and feedback received. Second, we should integrate our work with other threads of work within NeOn addressing collaborative aspects. In fact, currently we are testing the integration of our work with the argumentation support from WP2. Some of these aspects are:

- Improve some GUI aspects of our visualization components
- Finish and evaluate the integration of the argumentation support with our infrastructure
- Improve our synchronization process to support the resolution of conflicts and consequently the concurrent editing of distributed copies of one ontology
- Support configurable workflows

Finally, we would like to conduct additional experiments in other organizations and in particular we would like to evaluate the other configurations supported by our infrastructure.

Bibliography

[BSH86] Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentation in software engineering. *IEEE Trans. Software Eng.*, 12(7):733–743, 1986.

[FPJ97] Mariano Fernandez, Asuncion G. Perez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.

[GLP$^+$07] A. Gangemi, J. Lehmann, V. Presutti, M. Nissim, and C. Catenacci. C-ODO: an OWL meta-model for collaborative ontology design. In *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at WWW 2007*, Banff, Canada, 2007.

[HBP$^+$07] Peter Haase, Saartje Brockmans, Raul Palma, Jérôme Euzenat, and Mathieu d'Aquin. Updated version of the networked ontology model. Technical Report D1.1.2, University of Karlsruhe, AUG 2007.

[HP05] J. Hartmann and R. Palma. OMV - Ontology Metadata Vocabulary for the Semantic Web, 2005. v. 1.0, available at `http://omv.ontoware.org/`.

[ISO98] ISO, editor. *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. 1998.

[Kle04] M. Klein. *Change Management for Distributed Ontologies*. PhD thesis, Vrije Universiteit, Amsterdam), 2004.

[KPP$^+$02] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, El, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *Software Engineering, IEEE Transactions on*, 28(8):721–734, 2002.

[MAP$^+$08] D. Maynard, N. Aswani, W. Peters, S. Angeletou, and M. dâĂŹAquin. D1.5.2 implementation of metadata evolution. Technical Report D1.5.2, University of Sheffield (USFD); NeOn Deliverable, FEB 2008.

[MGGPISK07] Ó. Muñoz-García, A. Gómez-Pérez, M. Iglesias-Sucasas, and S. Kim. A workflow for the networked ontologies lifecycle. A case study in FAO of the UN. In *Proceedings of the CAEPIA-TTIA 2007*, Spain, 2007. Springer.

[MGKS$^+$07] O. Muñoz-García, S. Kim, M. Iglesias Sucasas, C. Caracciolo, A. Bagdanov, Y. Wang, P. Haase, M. Suarez-Figueroa, and A. Gomez-Perez. Software architecture for managing the fisheries ontologies lifecycle. Technical Report D7.4.1, NeOn Consortium, OCT 2007.

[NM02] Natalya F. Noy and Mark A. Musen. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 744–750, Edmonton, Alberta, Canada, July 2002.

[Pfl95] Shari Lawrence Pfleeger. Experimental design and analysis in software engineering: Part 2: how to set up and experiment. *SIGSOFT Softw. Eng. Notes*, 20(1):22–26, 1995.

[PH05] R. Palma and P. Haase. Oyster - sharing and re-using ontologies in a peer-to-peer community. In *International Semantic Web Conference*, pages 1059–1062, 2005.

[PHWd07] R. Palma, P. Haase, Y. Wang, and M. d'Aquin. D1.3.1 propagation models and strategies. Technical Report D1.3.1, UPM; NeOn Deliverable, NOV 2007.

[SSSS01] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.

[Sto04] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe (TH), Germany, August 2004.

[Tem06] C. Tempich. *Ontology Engineering and Routing in Distributed Knowledge Management Applications*. PhD thesis, University of Karlsruhe (TH), Germany, 2006.

[TN07] T. Tudorache and N. Noy. Collaborative protege. In *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at WWW 2007*, Banff, Canada, 2007.

[vV98] Erik P.W.M van Veenendaal. Questionnaire based usability testing. In *In Proceedings of the European Software Quality Week, Brussels*, 1998.