NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D2.3.1 Practical Methods to Support Collaborative Ontology Design

Deliverable Co-ordinator: **Klaas Dellschaft**

Deliverable Co-ordinating Institution: **Universität Koblenz-Landau (UKO-LD)**

Other Authors: **Klaas Dellschaft(UKO-LD), Aldo Gangemi(CNR), Jose Manuel Gomez (ISOCO), Holger Lewen(UKARL), Valentina Presutti(CNR), Margherita Sini (FAO)**

This deliverable describes a set of methods for collaborative editing of ontologies, consensus reaching and capturing the design rationale of ontology elements.

Document Identifier:	NEON/2008/D2.3.1/1.2	Date due:	January 31, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 11, 2008
Project start date	March 1, 2006	Version:	1.2
Project duration:	4 years	State:	Draft
		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 writing parts of this document:

- Consiglio Nazionale delle Ricerche (CNR)
- Food and Agriculture Organization of the United Nations (FAO)
- Intelligent Software Components S.A. (ISOCO)
- Universität Karlsruhe – TH (UKARL)
- Universität Koblenz-Landau (UKO-LD)

Change Log

Version	Date	Amended by	Changes
0.1	09-10-2007	Klaas Dellschaft	First draft outline
0.2	14-11-2007	Klaas Dellschaft	Integrated the contribution of FAO
0.3	01-02-2008	Klaas Dellschaft	Updated contributions from FAO. Editing of chapter 2.
0.4	06-02-2008	Klaas Dellschaft	Editing of chapter 2. Added conclusions and executive summary.
1.0	11-02-2008	Klaas Dellschaft	Final editing and proof reading before sending to QA.
1.1	06-03-2008	Klaas Dellschaft	Integrated short definition of collaboration and clarification of relationship between work in WP2 and WP3.
1.2	14-04-2008	Klaas Dellschaft	Implemented the suggested changes from the QA.

Executive Summary

In this deliverable, we will identify a list of features that are required for supporting collaborative ontology design. The focus is on scenarios where the development team is distributed over multiple locations, like it is the case for the FAO and the invoicing case study.

For example, in the FAO case study the world-wide users of the AGROVOC ontology can also participate in its further development. Currently, they can only provide suggestions via mailing lists or a web form. But for the future it is planned to allow direct collaborative editing by the users. Nevertheless, FAO staff will still review the changes before they are published in an official version of AGROVOC.

In the invoicing case study, several Spanish laboratories organized in the PharmaInnova cluster are collaborating for defining a common invoice model which can be used for exchanging invoices between the laboratories. Currently, all participants have to physically meet at one place but for reducing the costs and required person effort the physical meetings should be reduced to a minimum and be replaced with online techniques for collaboration over the internet.

Based on an analysis of the previously described case studies and state-of-the-art ontology editing tools with collaboration support like Collaborative Protégé, a list of required features can be identified. The list includes features like the annotation of ontology elements with further explanations of, e. g. their design rationale or the support for computer-mediated communication between the different participants of the ontology design process.

Three different techniques will be explained in more detail, each addressing one or more of the identified required features. Furthermore, we will provide pointers to related work in other workpackages of NeOn which also provide techniques for supporting collaborative ontology design.

First, we will describe the WikiFactory framework that addresses the problem of collaboratively editing an ontology. It allows for deploying a given ontology into a semantic wiki like the Semantic MediaWiki. Then, the ontology can be modified by editing the created wiki pages. It has the advantage that also domain experts, who are typically not familiar with sophisticated ontology editing tools, can easily contribute to maintaining an ontology. Furthermore, the collaborative editing features of the wiki can be reused for editing ontologies.

The second technique is related to supporting the discussion and decision taking processes during the ontology design process. Cicero, a wiki based tool will be presented. It supports its users in applying the idea of issue based information systems (IBIS). In previous studies it was shown that an IBIS like structure leads to more efficient and better structured discussions.

The discussions captured with Cicero reflect the design rationale of the corresponding discussed ontology elements. They are an important part of the provenance information of ontology elements, explaining their current state. Knowing and documenting the design rationale of ontology elements is especially of importance in the case of collaborative ontology design. In this deliverable, we will make a proposal of how to easily add the required provenance annotations to an ontology by integrating the Cicero tool with the NeOn toolkit.

Note on Sources and Original Contributions

The NeOn consortium is an inter-disciplinary team, and in order to make deliverables self-contained and comprehensible to all partners, some deliverables thus necessarily include state-of-the-art surveys and associated critical assessment. Where there is no advantage to recreating such materials from first principles, partners follow standard scientific practice and occasionally make use of their own pre-existing intellectual property in such sections. In the interests of transparency, we here identify the main sources of such pre-existing materials in this deliverable:

- Chapter 4 is partially based on [DS08].

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this deliverable, we will identify a list of features that are required for supporting collaborative ontology design. Loosely speaking, with the term *collaboration* we mean either (i) situations of reuse-oriented collaboration (e. g. reusing existing ontologies or ontology design patterns, making an ontology functional to a given task through appropriate evaluation and selection, re-engineering thesauri, folksonomies, subject directories, textual data) or (ii) situations of active interaction between agents for producing a knowledge resource (e. g. collaborative editing and annotation of ontology elements and changes, discussing and getting consensus on design solutions and their rationales in context).

More precisely, the notion of *collaborative workflow* has been defined in C-ODO [CGL+07a] as a special type of *epistemic workflow*. Epistemic workflow is one of C-ODOŠs key-concepts, it subsumes other types of workflows, like interaction or usage, for which the bonds between the agents involved in the workflow are weaker than in the case of collaboration.

Collaborative workflows are typically based on teams that include accountable agents, i. e. accountable agents are conscious of their participation to the same plan and share a same goal. The focus of this deliverable is on collaboration scenarios where the development team is distributed over multiple locations, like it is the case for the FAO and the invoicing case study. The relevant case study scenarios will be described in Chapter 2. Based on an analysis of the scenarios and state-of-the-art ontology editing tools with collaboration support like Collaborative Protégé, this chapter also contains a list of features required for collaborative ontology design.

In the rest of the deliverable, we will describe concrete techniques that support the required features and that facilitate collaborative ontology design. In Chapter 3, we will describe the WikiFactory framework that addresses the problem of collaboratively editing an ontology. It allows for deploying a given ontology into a semantic wiki like the Semantic MediaWiki. The WikiFactory approach has the advantage that also domain experts, who are typically not familiar with sophisticated ontology editing tools, can easily contribute to maintaining an ontology.

The second technique described in Chapter 4 is related to supporting the discussion and decision taking processes during collaborative ontology design. The Cicero extension for Semantic MediaWiki supports its users in applying the idea of issue based information systems (IBIS). Having an IBIS like structure leads to more efficient and better structured discussions. The manual of the tool is available in Appendix A. Furthermore, we will describe in Section 4.3.2 how Cicero may be used for capturing provenance information of ontology elements by integrating the Cicero tool and the NeOn toolkit.

Chapter 2

Requirements for Collaborative Ontology Engineering

In this chapter, we will identify features that are relevant for collaborative ontology engineering and that should be supported within NeOn. We start by describing the collaboration scenarios in the case studies. After that, we will analyze the collaboration features of existing tools like Collaborative Protégé. We close this chapter with the list of required features and give pointers to the corresponding chapters in this deliverable or to other NeOn deliverables which address one or more of the listed features.

2.1 Collaboration Scenarios: Ontology Engineering at FAO

In the following, we will describe the actual ontology engineering workflows of three different ontologies that were developed at FAO. We will show that, despite their differences in size and scope, for developing all three ontologies the collaboration between several persons was crucial. We will also discuss how the workflows may be further improved by providing a better collaboration support.

2.1.1 The AGROVOC Thesaurus and Concept Server

AGROVOC is a multilingual thesaurus developed by FAO since 1983. Originally, it was only available in printed copy, but since 2000 it is available online. Originally, the development of the thesaurus was done by several experts who met regularly in order to discuss how to enrich it. The collaboration was therefore with face-to-face meetings.

Translations of the thesaurus are currently done in partner countries. Experts meet and organize themselves for the development of a version of the thesaurus in the national language. FAO provides guidelines[1] and tools[2] for this task, but some countries also have their own tools. Once the national version is completed, it is sent to FAO for inclusion in the master copy of the thesaurus and published online.

For some time now, it has been possible to add new terms to AGROVOC by direct suggestions from users all over the world: the suggestions, in multiple languages, arrive to FAO through the means of a specific web-form or emails. The suggestions are then evaluated and committed to the ontology by the FAO AGROVOC team (see Fig. 2.1).

At the beginning of 2007, FAO has promoted a virtual conference, called the AGROVOC E-conference. This has been recognized as an effective means for further collaboration between AGROVOC experts and users all over the world. It didn't require any working mission or trips by participants, with consequent save of money and time, and allowed asynchronous participations, with consequent benefit of giving time to users to

[1] ftp://ftp.fao.org/gi/gil/gilws/aims/publications/papers/agrovoc_translation_guidelines.pdf
[2] http://www.fao.org/aims/tools_thes.jsp

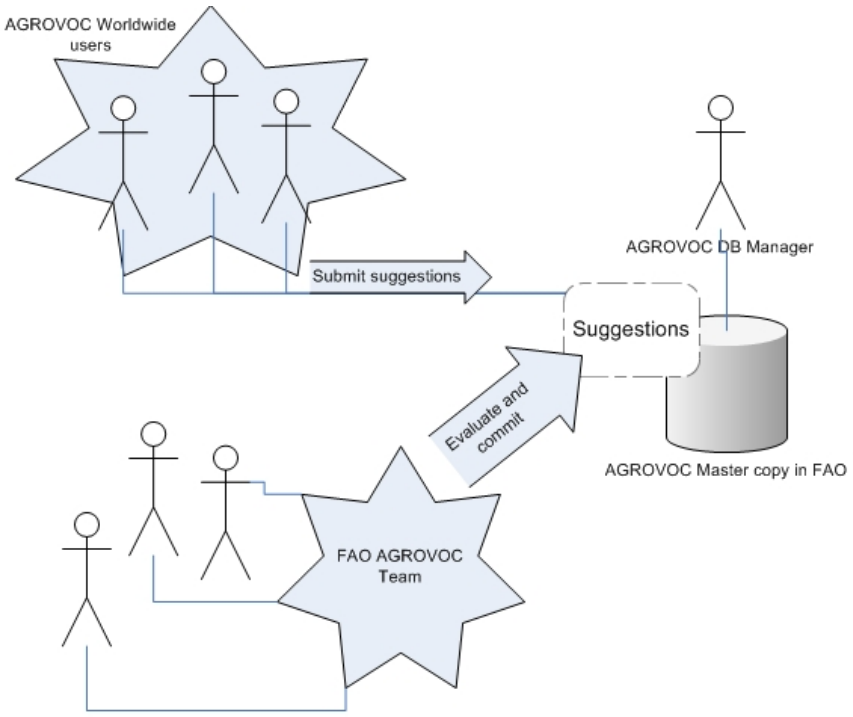

Figure 2.1: Current workflow for collaborative content maintenance of AGROVOC.

reply when and from where was more convenient for them. In this case, a mediated communication over the internet was used for organizing the work.

Both activities of creating and maintaining AGROVOC involve many actors who need to collaborate, e.g. during the sequential actions of initial creation and translation. The collaboration takes place between the different sequential actions as well as within them:

- The creation of terms involves many domain experts to identify what are the best terms to use for the thesaurus. Furthermore, it has to be decided which terms may be created as non-descriptors or where they should be put in the hierarchy (i.e. assign broader terms and/or narrower terms).
- The translation of terms involves discussions between translators and domain experts in order to evaluate and define the best translation of a term.

Therefore, for the collaborative aspects of the creation and maintenance of an ontology we need to consider collaborations over different steps performed by different people as well as collaboration among several participants for every single step. For these reasons, FAO recognizes the need for making the development and the maintenance of AGROVOC more collaborative and especially more direct for users without the mediating action of FAO staff members. In parallel with this, the idea to convert AGROVOC to a more complete structure allowing the representation of more information (such as linguistic information for terms, or the ability to have multiple translations of a specific term, etc.) has been investigated.

As a result, FAO has undertaken the development of a new system, called the AGROVOC Concept Server Workbench, which allows collaborative and distributed management of the new restructured AGROVOC (in this case we can talk of an AGROVOC Ontology). The resulting workflow is shown in Fig. 2.2 (more details about the editorial workflow are also available in [PWHd07]). The collaboration in this case is much more effective because:

- AGROVOC editors all over the world can have direct access to a unique and homogenized maintenance tool;

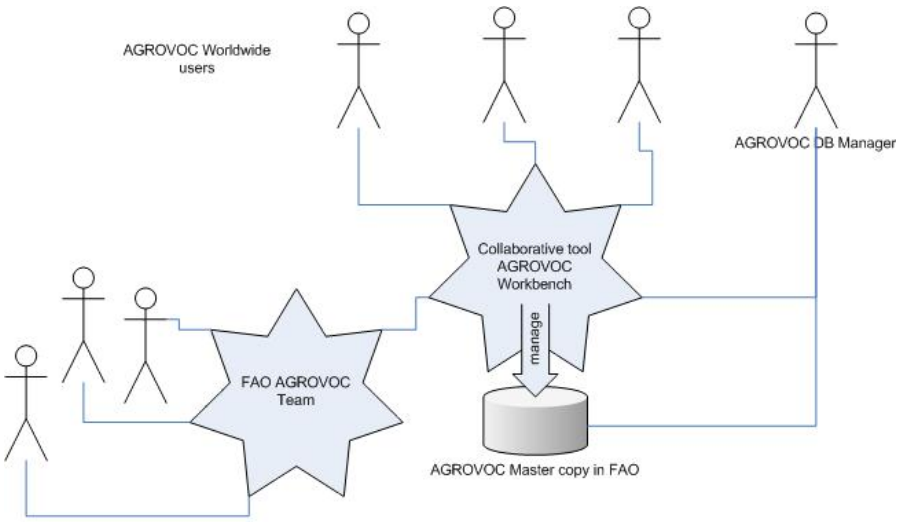

Figure 2.2: Planned workflow for collaborative management of AGROVOC.

- changes are immediate and there is no need to wait for FAO actions (eventually only validations of proposed changes);
- all users can immediately see and benefit from other users contributions;
- the cycle of adding data to AGROVOC and using it is more immediate, because after the data are inserted and eventually validated, they become immediately available for remote access through web services or for download (currently, a new version of AGROVOC is published only every three months).

In this case we have identified several roles for people interacting with each other:

- Domain Experts: Librarians, indexers and AGROVOC users all over the world can be seen as domain experts which need to enrich the ontology (AGROVOC concept server). They can be further divided into:
  - ontology experts (the ones that work on the hierarchies and relationships of concepts),
  - terminology specialists (the ones that are interested only on the translations of terms).
- AGROVOC Experts: The AGROVOC managers who can validate and/or publish changes of the domain experts.
- Ontology Engineers and the AGROVOC DB Manager.

There are several activities these actors may be involved into, such as:

- concept or term creation and editing,
- concept or term deprecation,
- concept or term annotation (e.g. management of images and definitions, etc.),
- relationship management,
- validation of changes,
- publication of new versions.

Collaborative tools should in this case analyze the different actions users may perform in order to facilitate maintenance tasks (e.g. identify categories of annotations, etc.). In addition to that, logs should record every user actions, and the system may provide statistics on user activities.

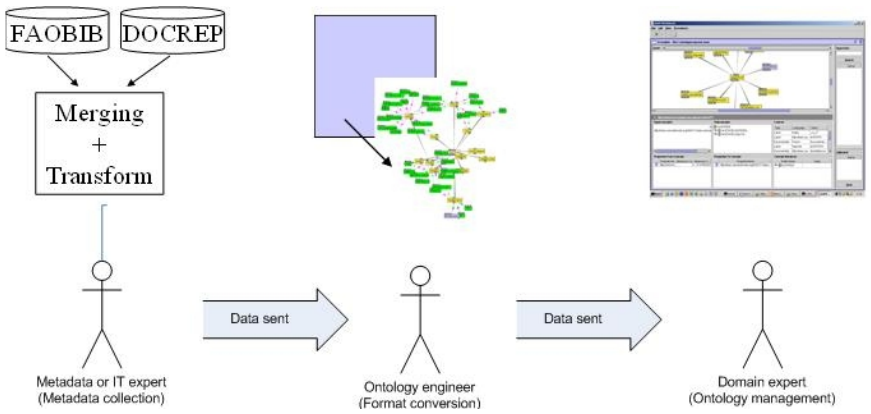

Figure 2.3: Workflow for the FNA Ontology.

2.1.2 FNA: The Food, Nutrition and Agriculture Ontology

The "Food, Nutrition and Agriculture" Ontology has been developed in FAO in 2004 to store metadata elements of the FNA journals[3]. During its development, the KAON format and tools were used (mainly the OI-modeller). The ontology was developed using the effort of 3 people (one metadata expert, one ontology engineer and one domain expert).

The collaborative workflow for the creation of the ontology was pretty easy, because the metadata expert exported needed data from databases and compiled them into a topic-map file. This file was then passed to the ontology expert who converted this file into a KAON ontology and checked that all the needed elements were present and eventually created them. Then, a trained domain expert took the KAON file and used the KAON interface for completing the ontology with all needed relationships and adding multi-linguality. The editorial workflow can in this case be considered linear (see Fig. 2.3).

The maintenance of the FNA ontology does not require specific tools or actions because it is only seldom changed and thus it is sufficient if the trained ontology editor updates the ontology using the KAON OI-modeler tool. However, this is a special case requiring only a minimum of collaboration.

2.1.3 CWR: Crop Wild Relative Ontology

The Crop Wild Relative Ontology contains a subset of about 400 terms highly relevant to crop wild relatives. Terms with high relevance to the CWR domain are grouped into themes, roughly corresponding to AGROVOC (top level) categories, or indicatives of the thematic sources from which the terms were collected (biological and geographical online dictionaries etc.), with an attempt to balance the number of terms between the groups. For the import into the ontology structure, the themes were converted to namespaces in order to preserve the grouping and allow manipulation within ontology client programs on terms based on namespace grouping. Before the import, the namespaces were slightly modified and adapted to some other existing ontologies. The ontology is available as OWL Full and OWL DL.

The collaborative workflow involved FAO staff and an external collaborator who provided his knowledge as a domain expert. FAO prepared the requirement and the minimal set of terms from AGROVOC that were used for the development of the ontology. The compiled information was sent to the domain expert who manually populated the ontology. The ontology was sent back to FAO for analysis and final set-up (including model restructuring and language enrichment). The resulting workflow is shown in Fig. 2.4.

In this case we also have several steps performed by different actors. The CWR development is particularly interesting as there was a clear distinction between ontology engineers and ontology editors. Their roles are very different and they performed actions in different steps. However, it is desirable and needed to have a tighter collaboration between the different actors.

[3] http://www.fao.org/ag/agn/publications/fna/index.jsp?lang=en

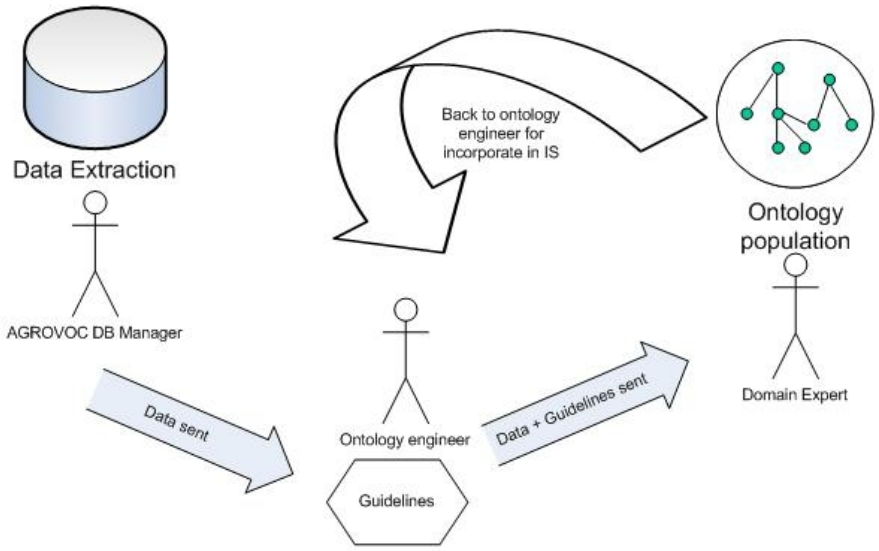

Figure 2.4: Workflow for the CWR Ontology.

2.1.4 Generalized Scenario

In general, the collaborative workflow for the development of ontologies has the following steps and corresponding actors (cf. Fig. 2.5):

- The initial ontology design and population is done by one or more ontology engineers who are generally preparing the structure of the ontology.
- Afterward, one or more domain experts or ontology editors are further populating the initial ontology. Also information specialists may be involved who populate the ontology from databases etc.
- Next, translators may translate the elements of the ontology.
- Finally, the ontology is integrated into the information system and published by ontology or IT experts.

The identified activities may be executed in subsequent steps (like in the case of the FNA or CWR ontology) or they may be executed in parallel (like in the case of AGROVOC). Having parallel activities instead of subsequent steps increases the amount of interaction and collaboration between the different participants and roles of an ontology engineering project.

2.2 Collaboration Scenarios: Electronic Invoice Management

The term collaboration can be defined as the act of working together while accomplishing a united goal. This definition implies involvement of a number of different agents cooperating in such task. However, collaborative processes can be supported by tools that are not themselves of a collaborative nature, e. g. workflow representation and reasoning tools, but which on the other hand can assist processes of a collaborative nature.

In this regard, collaboration in the context of electronic invoicing can be seen in the activity of defining invoice models. During this activity, creators of invoice models and standards take care of the definition of invoice models for the pharmaceutical sector. This is usually a collaborative process where representatives of the different participating organizations, like the laboratories of PharmaInnova, physically meet and negotiate the information that must be contained in such models. For the case of the PharmaInnova pharmaceutical cluster, the subjects of discussion for this argumentation-intensive process are:

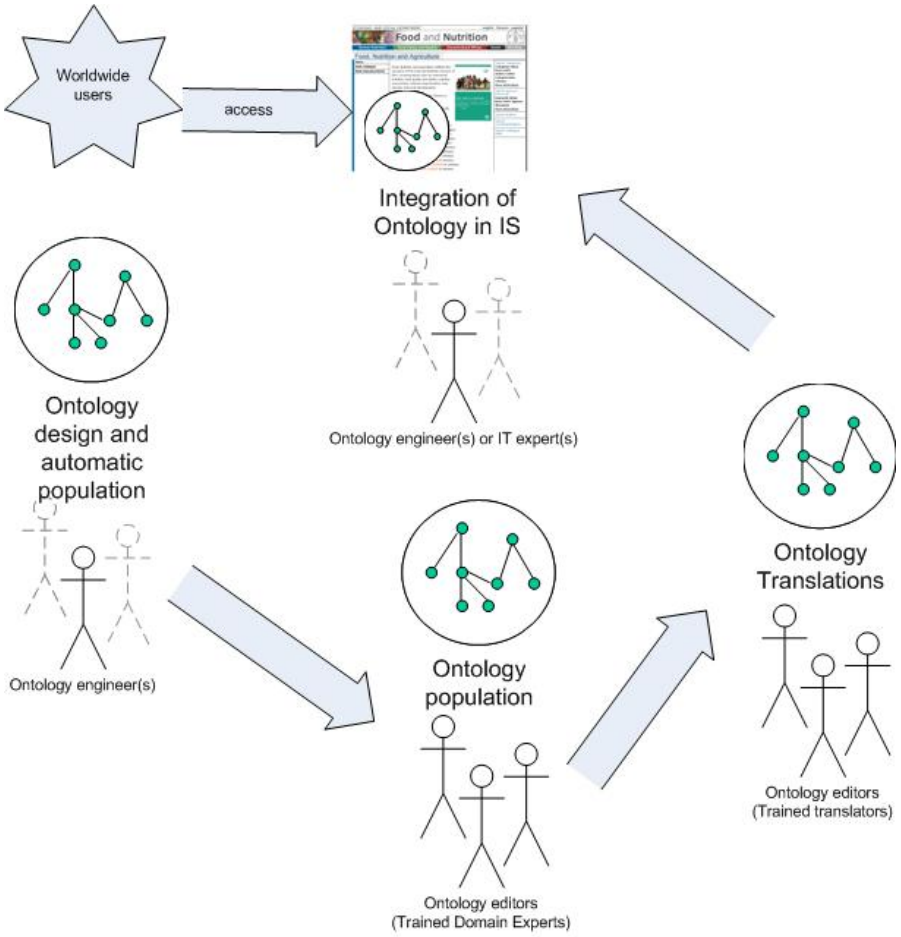

Figure 2.5: Proposed workflow for Collaborative Ontology management and use (covering entire life cycle). Participants may access the same ontology through the same tool, or the same ontology from different tools

- Data types contained in the invoice model. So far: compulsory by law, compulsory for integration, and informative.
- Main concepts of the invoice model: head, body, and summary.
- Specific fields in each concept.
- Relations and constraints between the different fields of the invoice model.

As can be anticipated, this is not an efficient way to agree on a common conceptual model to be shared by the different organizations members of a cluster like PharmaInnova. More powerful tools for supporting the consensus reaching process are required. They would especially be useful in the case of dynamic sectorial clusters like PharmaInnova, where the addition of a new member might trigger a new round of discussions.

In this context, collaborative argumentation tools should enable domain experts with no background in ontology engineering to develop themselves such consensual models representing the required information. The key issue here is usability: how to make the use of such tools easy for domain experts. If this is achieved, the models, resulting from the cooperation among partners, will reflect with higher accuracy the requirements of all the members of the target community, in our case the pharmaceutical laboratories.

But such an approach would also have two further advantages: First, it would result in reduced effort of expensive human resources and travel costs because face-to-face meetings can be avoided. Second, such approach would also advance the democratization of the internal processes of a cluster like PharmaInnova because it would ensure that all member laboratories, regardless of their size, have equal rights to participate.

For the future, it is planned to do an experiment that involves the WikiFactory framework and the Cicero argumentation tool that are described in Chapter 3 and 4. In the experiment, it will be assumed that a new member has joined the PharamaInnova cluster and that its invoice model has to be incorporated into the common model. In this scenario, WikiFactory will be used for viewing and comparing the two invoice models with each other while Cicero supports an asynchronous discussion and decision taking process between the participants in the experiment. Further planning and conducting the experiment will be subject to future work in WP2.

2.3 Analysis of Existing Tools

In the following, we will analyze the collaboration features in existing tools. The main focus will be on the recently developed Collaborative Protégé, but also other tools will be described.

2.3.1 Collaborative Protégé

In order to extend the Protégé Ontology Engineering environment with collaborative functionalities, several new features have become part of the full Protégé installation. In Collaborative Protégé[4] users can annotate ontology elements as well as ontology changes. Also voting and user interaction via chat are supported. Collaborative Protégé has been developed as an extension of the multi-user Protégé system, which already allowed multi-user-access and editing [TN07]. It works by having one dedicated server running that coordinates all users connecting to the server in client mode.

Figure 2.6 presents a sketch of the core components of the Collaborative Protégé architecture. The user only interacts with the ontology editor component (which is provided by the underlying Protégé system) and the annotation component which is driven by a RDF(S) annotation ontology providing structure for the annotation types. The change tracking component converts the user GUI interactions into change annotations attached to the changed ontology components.

[4] http://protege.stanford.edu/doc/collab-protege/

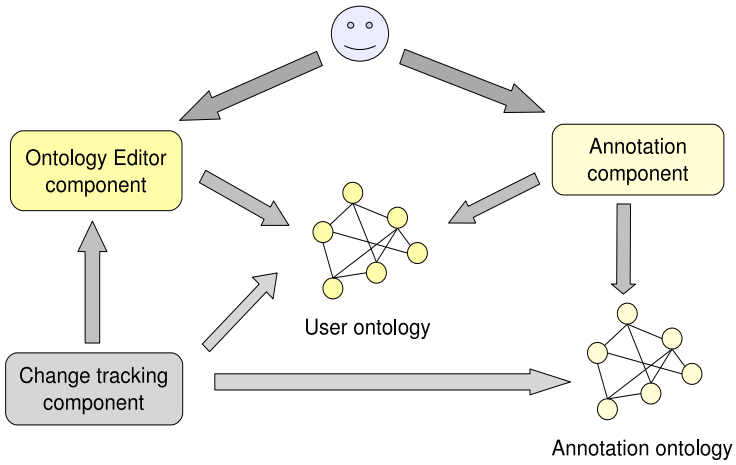

Figure 2.6: Core components of the Collaborative Protégé architecture [TN07].

It is also noteworthy that the current version can be used both as a stand-alone application (several users edit the same ontology at different timepoints) or in a multi-user setting (where multiple clients can edit the same ontology in a concurrent fashion). More details can be found in a first publication on Collaborative Protégé [TN07]. In the following, we will list its collaboration features in a little bit more detail.

Annotation of ontology elements

Ontology components (classes, properties and instances) can be annotated by using the annotation tab in the collaborative panel. This is done by clicking on the component you want to annotate so it is selected and then selecting the annotation to add. As can be seen in Fig. 2.7, other users can see the annotation along with details such as author.

Annotation of ontology changes

In Collaborative Protégé, all changes are tracked by the system. Changes include the creation, deletion or renaming of ontology components. This enables the user to also annotate changes. In order to do so, the change has to be selected in the collaboration panel and then the type of annotation has to be selected (see also Fig. 2.8.

Support for change proposals and voting proposals

Currently, two different types of change and voting proposals are implemented in Collaborative Protégé: The *Agree / Disagree* proposal with the respective *Agree / Disagree* votes, and a *Five Stars* proposal with the corresponding *Five Star* vote. Proposals can be started in any of the Annotations, Changes, All or Discussion Threads tabs. They can also be linked to ontology components or changes by selecting them and then starting the proposal. In Fig. 2.9 you can see an example of such a voting. Currently, a proper workflow for the voting mechanism has not yet been implemented, but is announced for upcoming versions.

Support for filtering of existing annotations

The mechanism for filtering annotations becomes especially useful when a multitude of comments or annotations are present for an ontology component and one is only interested in certain types of annotation or annotation by a specific user. In Fig. 2.10 you can see an example of such a filtered list of annotations.

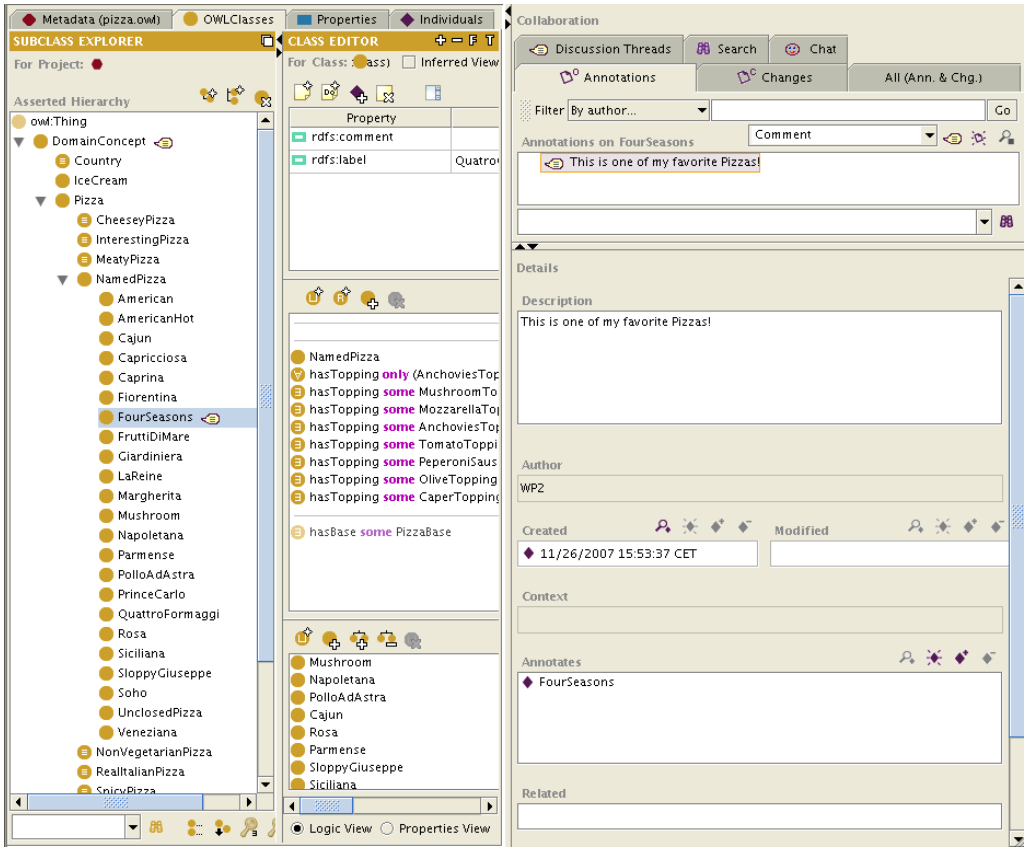

Figure 2.7: Example of an annotated class. The icon next to the class “FourSeasons” indicates an existing annotation. The annotation is displayed in the collaboration panel along with details such as author or creation date.

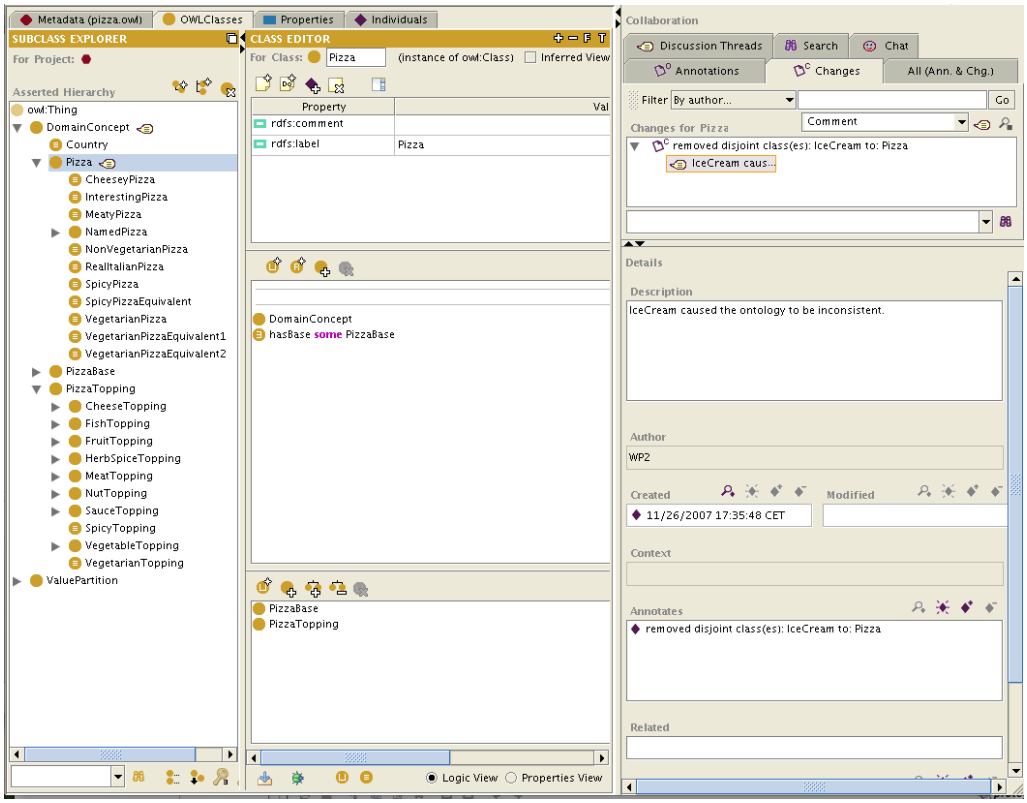

Figure 2.8: Example of an annotated change. The class “IceCream” has been deleted and the collaboration panel displays the change with an additional explanation by user WP2.

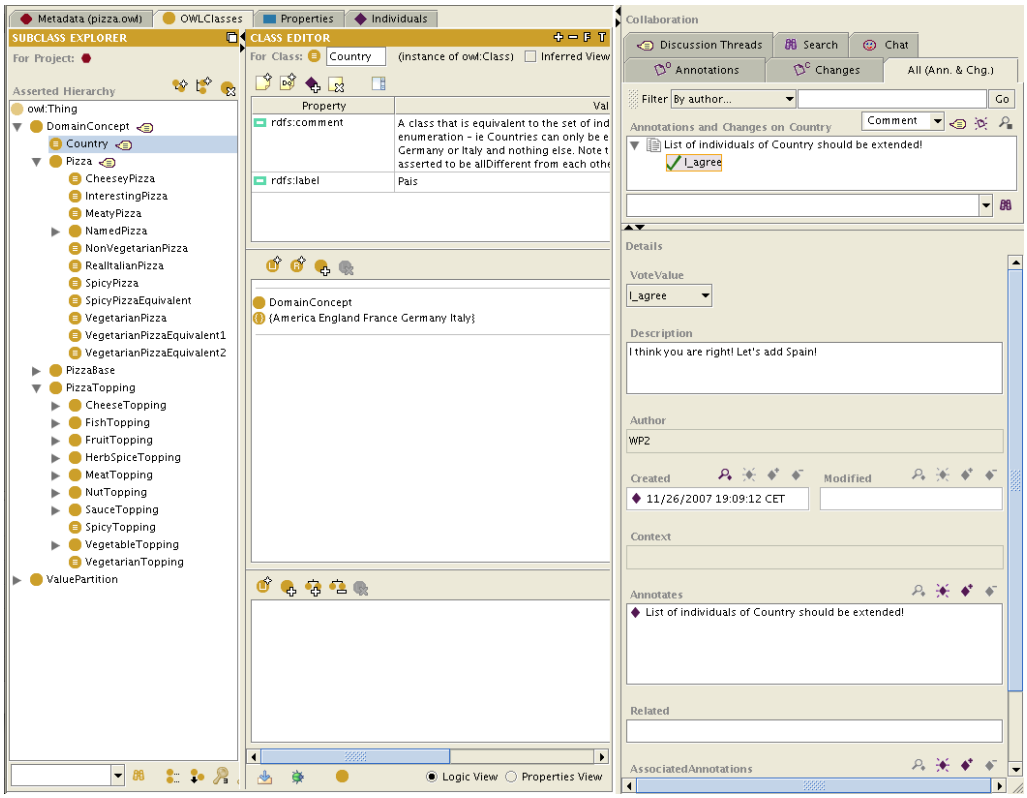

Figure 2.9: Example of an Agree / Disagree vote proposal. For class “Country” it is proposed that the list of individuals should be extended. User WP2 has voted “I agree” and given an explanation for the vote.

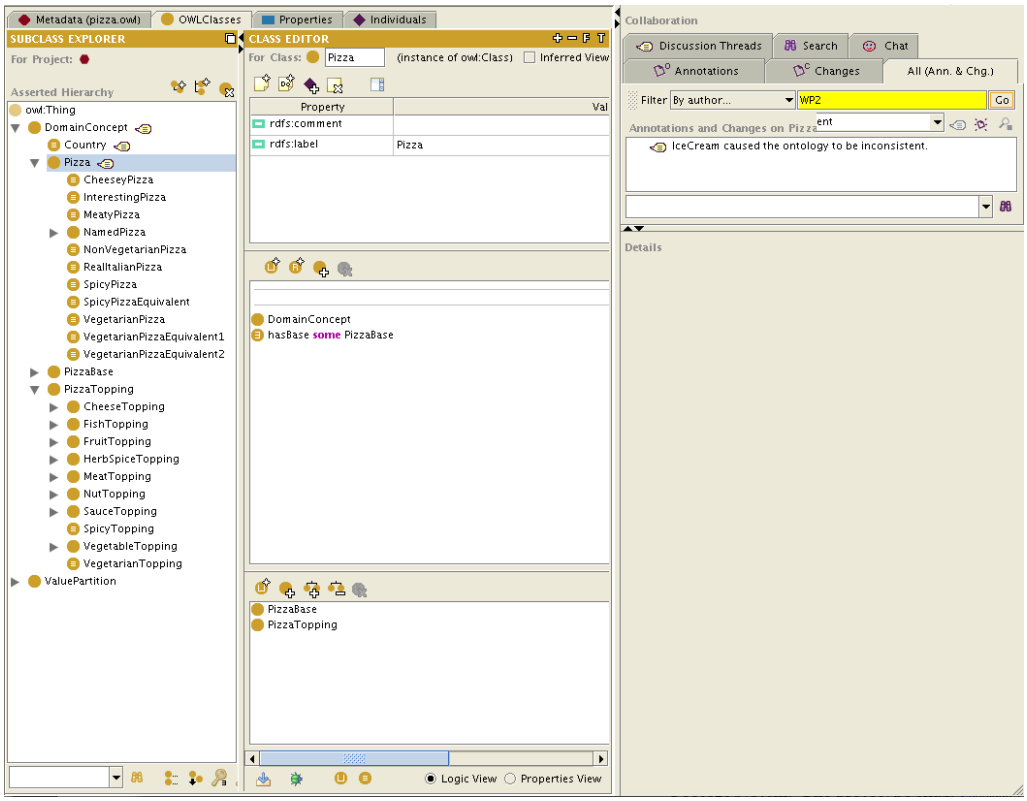

Figure 2.10: Example of the filtering mechanism. The class “Pizza” was selected and then the resulting comments and annotations where filtered by author WP2. This results in a list of annotations which can then be selected to see the details.

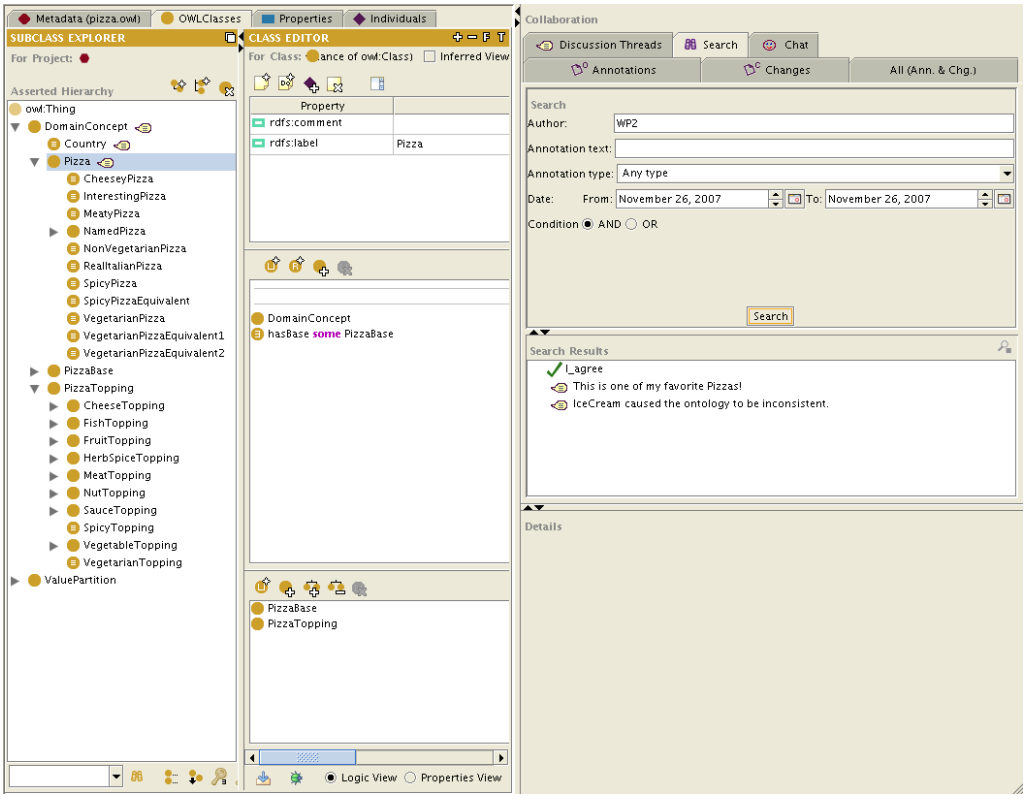

Figure 2.11: Example of the search mechanism. The user searched for all annotations by author WP2 that were created November 26th 2007.

Support for searching of annotations based on simple or complex criteria

When filtering is not expressive enough, users can use the search tab to ask more expressive queries. For example the time period, type of annotation, author or text can be entered. Fig. 2.11 provides a screenshot of such a query.

Support for discussion threads

Similar to the world wide web forums a discussion in Collaborative Protégé consists of different discussion threads where users can post comments and replies. This can also be used for a more asynchronous discussion. Fig. 2.12 provides an exemplary discussion thread.

Live Chat

Basically like any other chat on the world wide web, the chat feature implemented in Collaborative Protégé enables all users connected to the same server to interact in live discussions. A chat tab (see Fig. 2.13) displays messages along with author and time stamp.

2.3.2 Collaborative Ontology Engineering in the 90s

Before we discuss collaborative features in other current ontology editors, we briefly outline features supported by tools that were used in the 90s. [DSW+00] is a widely cited study that was performed by the University of Amsterdam in 2000 to compare the state of the art in ontology engineering tools back then. We will now give a quick summary of these tools' collaborative features based on the study. We will only men-

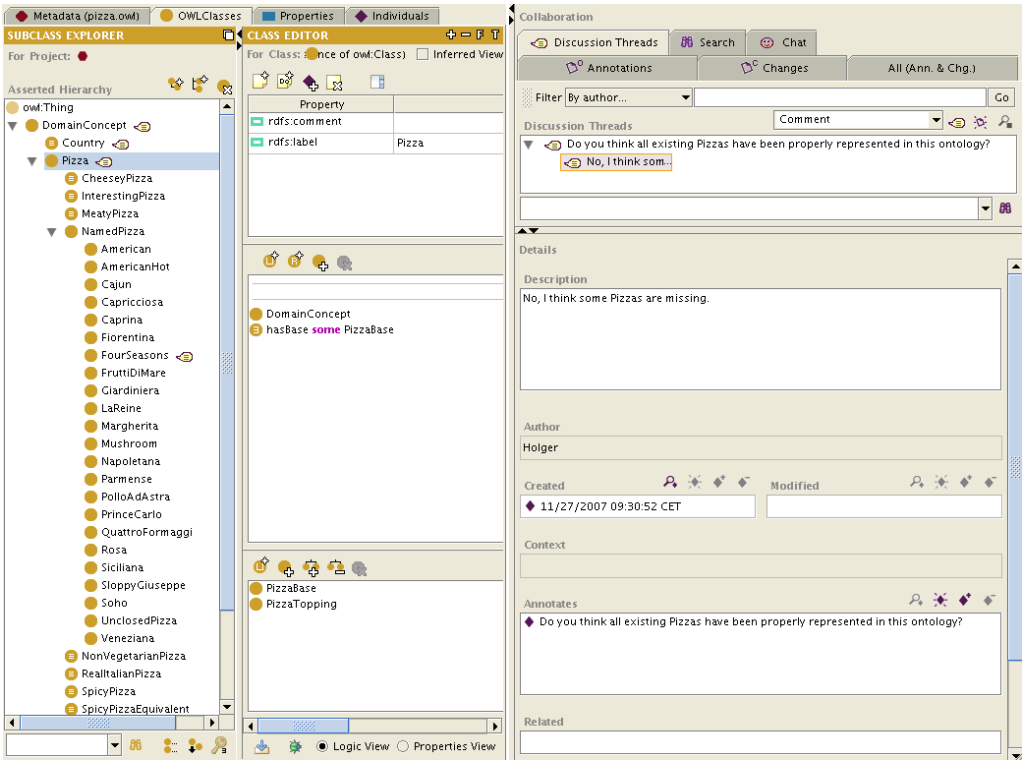

Figure 2.12: Example of a discussion thread. The reply “No, I think some Pizzas are missing.” annotates the question “Do you think all existing Pizzas have been properly represented in this ontology?”. Creator and creation date are stored and displayed on the details tab.

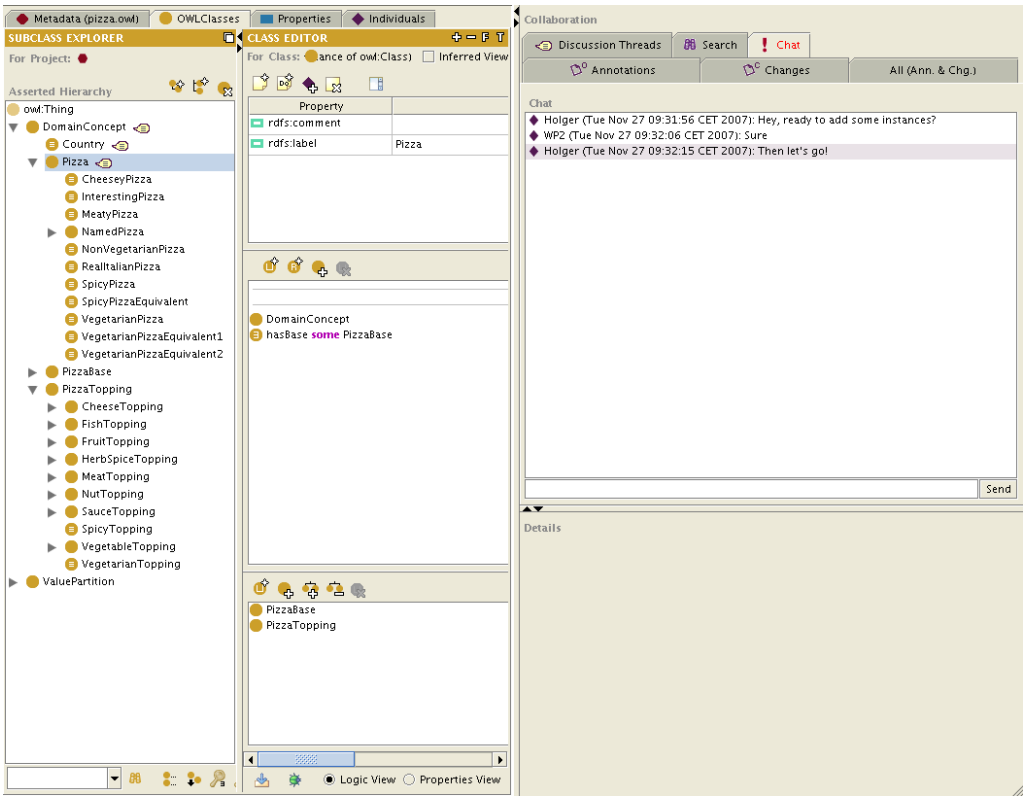

Figure 2.13: Example of a chat. The red exclamation mark indicates new messages.

tion those tools, that had collaborative features, and dismiss tools that only allowed asynchronous editing by exchanging the ontologies themselves via file.

Ontolingua

Ontolingua[5] [FFR97] was designed specifically with collaboration in mind. It is the only tool in the study supporting full synchronous editing of ontologies. Ontolingua runs on a server and can be accessed by multiple users using a standard web-browser. Using Ontolingua, several users can edit the same ontology at the same time with changes made visible to other users after they click on anything. Thereby consistency between the server version and the user version of an ontology is ensured. Also all users in a shared session are aware of changes made by other users and they have access to a shared undo-list. Together with the redo-list, this enables quick reverts to different versions of the ontology, similar to the way versioning of Wiki pages are managed in current Wikis. Using access management, ontologies can be locked for editing while still being available for browsing by other users.

WebOnto

The collaborative mode in WebOnto[6] [Dom98] is based on a feature called broadcast and receive mode. Basically when a user enables the broadcast mode and then enters the edit mode, changes are broadcasted to other users in receiving mode. However concurrent editing is not possible, since the ontology is automatically locked when one user is entering the edit mode. Viewing is still possible for other users.

Another feature for supporting collaboration in WebOnto is the annotation tool, which allows users to draw anything with a “drawing pencil” into the graphical window. These drawings are then displayed in all connected clients in real time.

Ontosaurus

In Ontosaurus[7][SPKR96] it is possible to work synchronously on an ontology, but every time a change is made the ontology is locked. Other users can see which user is editing and when the lock was created. They can then choose to email the editor or override the lock. After changes are made, the ontology must be updated and the user should return to the browse mode. This then removes the edit locks. One drawback is that it is not easily possible for other users to see which change was made, and by whom and when.

2.3.3 Further Current Tools

After seeing the collaboration features of first generation ontology engineering tools, we will now explore collaborative features of current tools. While it is safe to say that currently Collaborative Protégé has no real competitor since it can build both on the huge user-base and also the stable underlying Protégé ontology engineering environment, there are some ontology editors also putting more focus on collaborative aspects. An overview (including a table comparing functionalities) can also be found in a report on the Collaborative Knowledge Construction (CKC) challenge [NCA07], in which several tools competed. In the following we will give a brief description of the ontology engineering tools that can be deemed useful.

HOZO

The latest version of the Hozo Ontology Engineering Environment[8] [KKIM02] also supports asynchronous ontology construction [KSKM07]. In Hozo, ontologies usually consist of (multiple) inter-connected modules.

[5] http://www.ksl.stanford.edu/software/ontolingua/
[6] http://kmi.open.ac.uk/projects/webonto/
[7] http://www.isi.edu/isd/ontosaurus.html
[8] Hozo.http://www.hozo.jp/

When a user checks out a module for editing it locally, it is locked and other users cannot edit it. When it is checked in again, Hozo uses the declared dependencies between the modules to identify conflicts. Based on a list of changes, users can decide to accept or reject changes made by other users. Its main focus is building and integrating ontology modules along with a conflict detection mechanism.

OntoWiki

OntoWiki[9] [ADR06] is a purely web-based application that allows collaborative building of ontologies. It also features different views on the data. The system keeps track of all the changes applied to the knowledge base and users can review this information. When editing classes or properties, OntoWiki redirects to pOWL, a web-based ontology authoring and management tool. OntoWikis main purpose is knowledge acquisition.

DBin

DBin[10] [TMN06] is a P2P system that allows users to collaboratively create knowledge bases. To facilitate this task, DBin allows the creation of domain- or task-specific user interfaces from a collection of components (such as ontology editors, views, forums or other plugins). Users can edit an ontology directly on the server and also track the changes made by other users. Provenance information is also provided by the system. DBin runs locally and connects to a P2P network from there.

SWOOP

SWOOP [KPS+06] allows collaborative ontology development by using an Annotea [KKPS02] plugin. The generated annotations can then be published and distributed using any public Annotea Server.

2.3.4 Trends in collaborative ontology editing

Comparing the tools from the 90s with up-to-date ontology engineering tools, one can see that the concurrent editing of an ontology has become more and more important. In Collaborative Protégé, the locking and protection mechanisms can not be observed by the users. So the ontology is not locked because one user edits it, but the system itself ensures consistencies of all clients. Relying on open standards like Annotea ensures compatibility with other tools and should be encouraged. Also live communication and the ability to annotate changes as well as ontology elements can be considered must-have features. So collaboration should be made easy, with the tool tackling the underlying problems like synchronization without having a noticeable effect (like an edit-lock) on the user.

2.4 List of Required Features

Based on the collaboration scenarios described in Section 2.1 and 2.2 as well as the analysis of existing tools for collaborative ontology engineering in Section 2.3, the following features should be supported within NeOn:

- *Annotation of ontology elements and changes*: One of the most important features of all collaborative ontology engineering tools is the annotation of ontology elements, be it classes, properties or instances. Furthermore, ontology changes can also be annotated for justifying them. A good integration and visualization of the annotation feature within the tool is important. While the editing and adding of annotations can be performed in a separate annotation tab, an in place mouse-over display

[9] http://3ba.se
[10] http://dbin.org

right at the annotated ontology element would increase usability. Also an icon indicating the existence of annotations is helpful (depending on the number of annotations, the opacity / color could change).

- *Discussion threads and change/voting proposals*: To guide the collaborative workflow, discussion threads and voting proposals are indispensable. Ideally, these mechanisms would incorporate a formalized argumentation framework, structuring the decision making process.
- *Chat*: As with any form of communication mechanism on the web nowadays, the direct chat enables people to talk about things ad hoc without thinking where to put it. Also, sometimes one wants instant feedback and not start a forum thread or similar. And when talking about multiple ontology elements at once, it is easier to just start discussing in the chat instead of distributing the discussion among multiple ontology elements with individual discussion threads or annotations.
- *Logging and versioning*: Similar to Wikipedia, sometimes it is needed to quickly undo a change. If someone accidentally deletes a concept or with malicious intents, other users should be able to quickly restore it. Also it is good to keep an overview on who changed what so that one can for example see if an expert user has made a change or a novice. Furthermore, the logging can be analyzed for providing user statistics, e. g. the number of actions a user has done. In the FAO use case, it would also be interesting to group the statistics by country or language.
- *Search within ontology elements and annotations / changes*: Ideally, the search functionality of a tool allows not only for searching the ontology elements, but also the annotations and changes users made.
- *Access right control and security*: Depending on the type of the project and the sensitivity of the information, not everybody should have access to the whole ontology. Here, access control and security measures come into play. Access right management should work on both individual user and group level.
- *Customization of user interfaces*: Different users may have different needs and roles and/or they may perform different actions (e. g. work on a specific module or with specific languages). For example, in the FAO case study we may have searches that customize the results based on the user – e. g. not logged in terminology editors can find only published concepts or terms, while registered users may retrieve also temporary or not yet validated terms. Based on this, it would be good to make the tool customizable and be able to save the preferences.
- *Sharing of documents*: While creating terminologies or ontologies it is very essential to make use of existing documents which may be a reference for specific terminologies. One important function may be sharing documents or text corpora in specific languages or covering a specific domain.

2.5 Collaboration Support in NeOn

Within NeOn, several techniques have been developed for supporting the above mentioned required features for collaborative ontology engineering. Two concrete techniques, the WikiFactory framework and the Cicero argumentation tool, will be described in this deliverable while others (e. g. techniques for versioning of ontologies and propagation of metadata) are described in the deliverables of other NeOn workpackages.

The WikiFactory framework will be described in Chapter 3. It allows for deploying an existing ontology into semantic wikis (e. g. the Semantic MediaWiki). The wiki can then be used for collaboratively viewing and editing the ontology. Typically, wikis have integrated logging and versioning mechanisms as well as other collaboration features like discussion pages and access right control (usually on the level of the whole ontology and not on individual elements). WikiFactory thus addresses several of the required collaboration features.

Furthermore, in Chapter 4 we will describe the Cicero argumentation tool. It facilitates an asynchronous discussion and decision taking process between participants of ontology engineering projects and supports

its users in applying the idea of issue based information systems. Cicero addresses the need for holding discussion in collaborative ontology engineering as well as change and voting proposals. Furthermore, the planned integration with the NeOn toolkit will allow for the annotation of ontology elements with the corresponding discussion as well as searching the content of discussions from within the NeOn toolkit.

As mentioned above, further techniques for supporting collaborative ontology engineering have been developed in other NeOn workpackages. For example, in WP1 concrete proposals of status annotations for ontology elements are made. Annotations might be related to the scope or the definition of an ontology element. Furthermore, a framework for managing changes and versions of ontologies is proposed. More details are available in [PWHd07].

Other required features will be supported by techniques from WP4. For example, there is ongoing work on how to manage access rights in ontologies (see [KGLCA07]) or how to provide customized views and user interfaces.

Chapter 3

Collaborative Editing

A web ontology describes a domain of interest, and is used by software applications in order to accomplish specific tasks on that domain. In order to be effective, a web ontology should encapsulate as much knowledge as needed for addressing such tasks. In business and organizational contexts, collaboration between ontology engineers and domain experts is essential in order to effectively build ontologies.

Over the semantic web, domain experts are web users, who need simple supporting tools to create and maintain their knowledge. Domain experts might not be familiar with formal languages, while they are usually confident with web browsing. Therefore, a web-based interaction between domain experts and ontologies is an obvious choice.

Hypertextual browsing and web-based editing of ontologies is not a new thing: hypertextual visualization of ontologies has been available since Ontolingua in 1992 [ARJ96], and web-based editing since Ontosaurus in 1995 [BRKT96]. Hypertexts and web forms are good visualization solutions, but past and current tools typically pre-configure one or more visualization choices. They neither attempt to align the structure of ontologies to related web content, nor allow the customization of the interface to the expertise level of the users. The WikiFactory tool is a first step towards semantically synchronous, customizable web ontology development and collaborative knowledge management. Simply put, WikiFactory is able to couple each element of an ontology (e. g., class, property, individual) to a wiki page (or to another web element), which contains the description of the semantics for that element. The coupling keeps web elements and ontology elements synchronized. Once the coupling is made, a domain expert is able to add elements, or to detect errors or gaps, i. e., he/she is able to validate the ontology or the associated content, by simply browsing and editing those pages.

A good story that supports the motivation for such a tool is exemplified here with reference to the UN-FAO case study described in Section 2.1. A group of FAO domain experts, in charge of maintaining the AGROVOC [agr] ontology, has been asked to validate the OWL version of AGROVOC [SLL$^+$05]. They have explicitly required not to use ontology editors and tools such as Protégé [Sta] or Swoop [MIN], as they find those tools non-intuitive. Furthermore, they prefer browsing the ontology by simply 'clicking' on its links, using a common web browser.

Their requests have been partly satisfied by automatically generating HTML documentation from Protégé, exploiting the OWLDoc functionality. OWLDoc can be compared to the Javadoc documentation [Sun] for Java classes. In an OWLDoc, each HTML page represents either a class, a property, or an individual of the ontology, and contains links to other ontology elements. Users can browse the ontology by following those links.

FAO domain experts have found the OWLDoc very friendly, and they have been able to efficiently discover errors and gaps. However, the OWLDoc does not provide them with editing functionalities, as it just generates static pages. Therefore, feedback coming from the domain experts evaluation has to be reported to ontology engineers, who in turn have to apply those changes to the ontology, and to produce an updated OWLDoc. This collaboration may be recursively necessary once further changes to the OWLDoc have to be applied.

Although the use of such technologies has been effective, the resulting workflow is awkward for several

reasons, including the fact that domain experts are geographically distributed, and the bad practice of putting tasks that can be performed by experts on the shoulders of ontology engineers.

On the contrary, if domain experts would use pages of a Wiki site, they could modify their contents directly from a web browser, and immediately see the changes.

Wikis are script-based applications that enable the friendly and quick online creation and editing of web pages. An example of a wiki is the MediaWiki [med][1]. Some wikis also provide semantic features; these wikis are known as semantic wikis (e. g. Semantic Media Wiki [Wik], IkeWiki [Sch06] or SweetWiki [BG06]).

A page in a semantic wiki encapsulates informal as well as formal definitions about the ontology element it represents. Formal content is encoded within the pages by including annotations, which are expressed in terms of additional syntactic constructs (very similar to syntactic constructs used for traditional formatting purposes). Existing semantic wikis typically allow users to (i) associate a page with either a class, a property, or an individual, (ii) type links between individuals by means of semantic relations that express facts, and (iii) express *subclassof* relations. Furthermore, semantic wikis typically allow users to export and save the underlying ontology in a standard format e. g., RDF[W3C] or OWL[MvH04]. Exporting the ontology is useful to perform some automatic processing and to reason on it. However, in order to fully exploit such semantic features, users must be familiar with formal languages.

Potentially, semantic wikis can be seen as powerful 'lightweight' collaborative tools for ontology creation and evolution, as they are capable of combining a friendly collaborative authoring environment with the strengths of semantic web technologies. To this aim, they should support a large number of semantic features and present them by means of an appropriate user-oriented interface.

Based on the above discussion, we have designed and developed a software framework called WikiFactory, a semantic web application whose innovative features can be summarized as follows:

Automatic deployment. Given an OWL ontology, WikiFactory is capable of creating automatically a corresponding semantic wiki.

Synchronization. Any change applied to the wiki content is reflected upon the OWL ontology and vice versa.

Help for users. WikiFactory helps users with a semantic assistant that exploits the capabilities of an OWL reasoner.

Therefore, WikiFactory enables the same facilities as OWLDoc (in this case through a Semantic Wiki), but it does not require a complex workflow such as the one described before, as it performs run-time synchronization between an ontology and its wiki representation.

In the following, we describe the current beta version of WikiFactory, exemplifying its features within a test case. In the test, we have used the FOSGeo ontology, which encodes a geographic ontology for the fishery domain. The rest of the chapter is structured as follows. In the next section we discuss the main WikiFactory functionalities and architecture. Section 3.2 describes the test case. Section 3.3 compares and contrasts our solution to related work. Section 3.4 summarizes current and future work.

3.1 The WikiFactory Framework

WikiFactory, downloadable at the Sourceforge web site [Sou], is an open source framework that enables the automatic creation of semantic wiki-based web sites and their dynamical management at run time. Figure 3.1 depicts the WikiFactory framework, which consists of distributed architectural components. The main features of our framework are summarized as follows.

WikiFactory is ontology-driven: a web ontology represents the domain that the web site supports.[2] It is

[1] See section 3.3 for additional references.

[2] Following common usage in semantic application descriptions, by *ontology* we mean here both OWL TBox and its related ABox, i. e. its knowledge base.

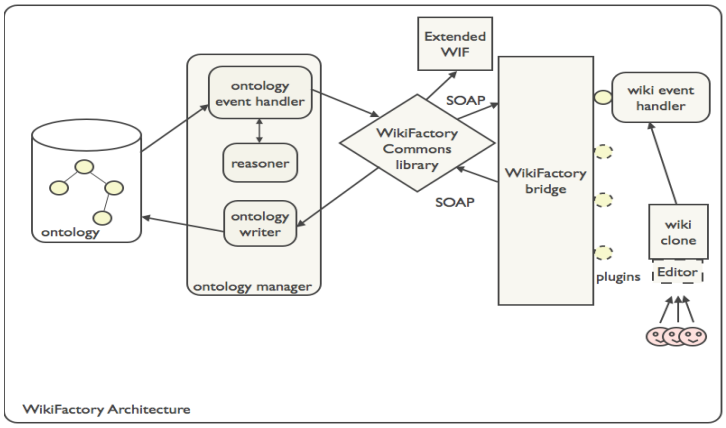

Figure 3.1: WikiFactory architecture

used to automatically deploy the wiki and then to manage the life-cycle of the wiki at run time, thus allowing the complete automatic detection of domain knowledge evolution.

WikiFactory enables the synchronization between the wiki and the ontology: on the server-side, as illustrated in Figure 3.1, our framework includes an ontology manager for OWL ontologies. The domain supported by the wiki site is represented by an OWL ontology, given as input to WikiFactory. The ontology manager of our framework is then responsible for (i) detecting ("reasoner" in Figure 3.1) and managing ("ontology event handler" in Figure 3.1) changes that occur in the ontology and that must be reflected on the wiki site and (ii) writing ("ontology writer" in Figure 3.1) the changes of the wiki site content that must be reflected on the ontology.

WikiFactory is based on IRE and DnS theories: The way WikiFactory implements the deployment and the synchronization between the OWL ontology and the wiki content is based on two models, namely IRE [GP07] and DnS [GM03]. The OWL versions of IRE and DnS are available on the web[3]. DnS and IRE are used to model WikiFactory operational semantics: for each ontology element, WikiFactory generates OWL code so that (i) each ontology element is reified according to DnS, and (ii) the representation of web structure elements (e. g., a wiki page) that contain information about an ontology element are typed according to IRE. Ontology element reification and its associated web structure element are mutually associated by means of a semantic relation that is defined in IRE [GP07], i. e., *ProxyFor*. The ontology writer is the WikiFactory server-side component that carries out that task.

WikiFactory enables the use of Web Services and Semantic Web technologies: WikiFactory consists of server-side and client-side components. Such components communicate by means of Web Service technologies, i. e. they use the SOAP protocol [GHM+06] and expose WSDL [CCMW01] interfaces to improve interoperability with other semantic web applications.

WikiFactory enables wiki platform independence: the input OWL ontology is deployed to wiki pages through an intermediate format named WIF [VO06]. For WikiFactory purposes we have extended the WIF format in order to support semantic features. To this aim we referred to RDFa [AB06] expressivity. The syntactic elements, which map semantic information (e. g., the type of a link) are organized by means of patterns for document structures and embedded in specific attributes of in-line elements and heading metadata. Any `WikiPage` is associated to a `WikiTemplate` indicating (i) the internal structures of that page, (ii) how they are nested and connected, and (iii) where are the placeholders that will be substituted by instance values. By applying templates, WikiFactory generates new documents (according to the WIF extended syntax) through a server-side component named WikiFactory Commons Library. Such documents contain data for each instance of the domain; these documents are finally delivered to a specific wiki clone. Extended WIF expresses whatever can be written in a semantic wiki dialect. Developers can bind WikiFactory to their platforms by implementing a bidirectional converter from WIF to any specific wiki syntax (see next bullet

[3] http://www.loa-cnr.it/ontologies/

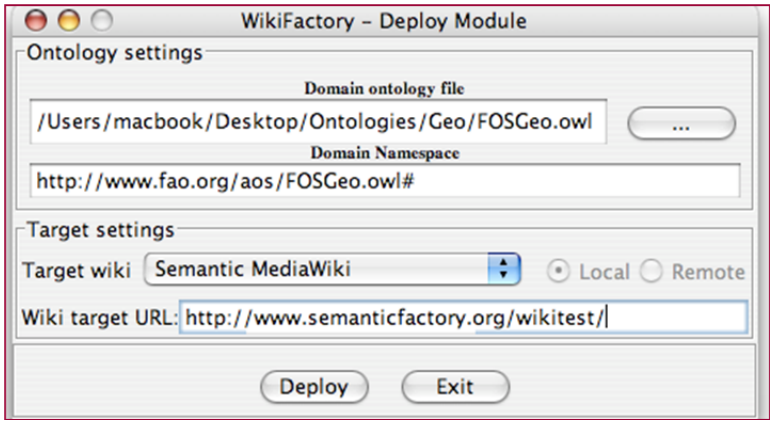

Figure 3.2: The WikiFactory Deployer GUI

point).

WikiFactory provides the automatic ontology-driven deployment and run time evolution of web sites: WikiFactory client-side components are wiki clone-dependent, attached to the wiki and managed by WikiFactory. They are responsible for (i) detecting and managing changes that wiki end-users perform upon the content of the wiki pages (wiki event handler in Figure 3.1) and (ii) abstracting the underlying wiki clone interactions. To achieve the latter purpose, we have designed a middleware layer constructed out of wiki-dependent plugins. The plugins hide the implementation details of the specific wiki clone being used, and expose an interface (the WikiFactory Bridge in Figure 3.1), used by WikiFactory server-side components. In the current version, WikiFactory includes a plug-in for Semantic MediaWiki [Wik].

WikiFactory allows user-oriented support to express semantics over a semantic web: WikiFactory guides users to express semantics as safely as possible: it enables users to express OWL constructs (it supports a subset of OWL DL), and contextually disallows those constructs that can result as mistakes. For example, for each page WikiFactory lists all properties that are applicable to the ontology element that wiki page is about, according to definitions expressed in the OWL ontology for that ontology element. In addition to end-user-oriented support, WikiFactory is designed to support ontology designers; to this aim, a set of functionalities has been already designed and implemented to be part of an editor embedded within the wiki (as shown in Figure 3.1).

3.2 How WikiFactory Works: a running example

In order to show how WikiFactory works in a realistic scenario, in this section we describe a test we have carried out in order to verify two WikiFactory functionalities, among those earlier described; namely the automatic ontology-driven deployment of semantic wikis and the synchronization between the wiki content and the ontology.

The test ontology is a simplified version of the *FAO Fishery Ontology Service Geo (FOSGeo)* ontology, which consists of 50 classes, 29 individuals and 29 object properties. The test consists of two main steps. The first step regarded the ontology deployment over a semantic wiki. Note that, in general, this deployment produces an ontology representation that is similar to OWLDoc. However, in such ontology representation hyperlinks in pages represent semantic relations between resources and when the wiki content is changed by the wiki users, the semantic content also changes dynamically.

The initial deployment is performed by the WikiFactory Deployer component, a Java application that requires as input the path of the domain ontology to deploy, the ontology default namespace, the target wiki platform and the URL of the target wiki site. Figure 3.2 shows the WikiFactory Deployer interface.

In our test, as illustrated in Figure 3.2, the WikiFactory Deployer is configured with the path of the FOSGeo ontology and the default namespace "`http://www.fao.org/aos/FOSGeo.owl#`". The chosen wiki

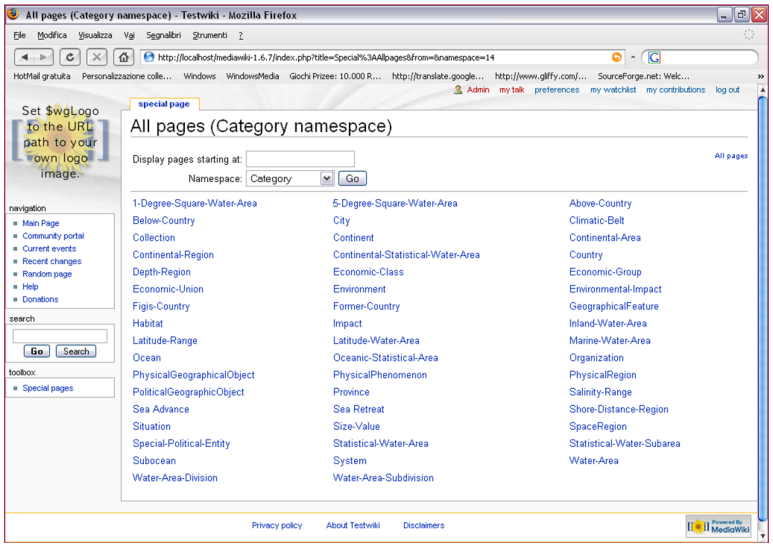

Figure 3.3: SMW categories pages

target platform is Semantic MediaWiki (note that WikiFactory can support any other semantic wiki platform) and the wiki site URL is `http://www.semanticfactory.org/wikitest/`.

Once the deployment process is started, a parsing of the ontology is performed. In this parsing phase, for each concept found (i. e., classes, individuals, properties) the WikiFactory Deployer creates a representation of its wiki page by using an extended version of the Wiki Interchange Format (WIF)[VO06] (extended WIF is able to capture the document semantics and helps to abstract from a particular wiki syntax). After the parsing phase, the Deployer converts each WIF concept description to the wiki target syntax by applying an XSLT transformation. This allows us to obtain the raw syntax for the final wiki page. Then, through a plugin component installed in the wiki host, the Deployer physically writes the new pages in the wiki.

The output of the deployment process is a semantic wiki instantiated over the input ontology. Figure 3.3 shows the Semantic MediaWiki page obtained from the test ontology; the page lists all the class pages available, given the initial ontology. For space reason we do not show the pages that list relations and individuals.

Each wiki page contains hyperlinks with different meanings. In particular, class and individual pages have links to classes (with the value of superclasses or rdf:types, respectively), and to applicable object properties. Moreover, a special box that represents active restrictions is added to each page of a class or an individual (in this latter case restrictions refer to those defined on the type of the individual). In addition, as WikiFactory recognizes class disjointness, it builds a list of properties applicable to classes, i. e. this list does not contain properties whose domain is disjoint from the current class.

The second step of the test verifies the synchronization functionality offered by our framework. We edit a new page and WikiFactory synchronizes the new wiki content with the underlying ontology model. Afterwards, a domain expert adds a page of an individual of type `Province` i. e., Napoli and WikiFactory automatically inserts (i) a link in the page for each class type (subclasses included), (ii) a box for the active restrictions and, finally (iii) the list of the applicable properties. Figure 3.4 shows the results of these operations.

On the ontology side, the corresponding OWL code is generated.

A further functionality offered by WikiFactory is the definition of a new restriction for a class. The restriction is added by using the special wiki page *Restriction Editor*, shown in Figure 3.5. Each property displayed in the list of the applicable properties for the class page has a link that allows us to add a new restriction over the property. When we click on this link we are redirected to the Restriction Editor page, where we can choose the type of restriction we wish to define over the property, and the corresponding restricted value type. Once we submit our choice, the new restriction is created in the ontology and correctly displayed in the restriction box of the restricted class' wiki page.

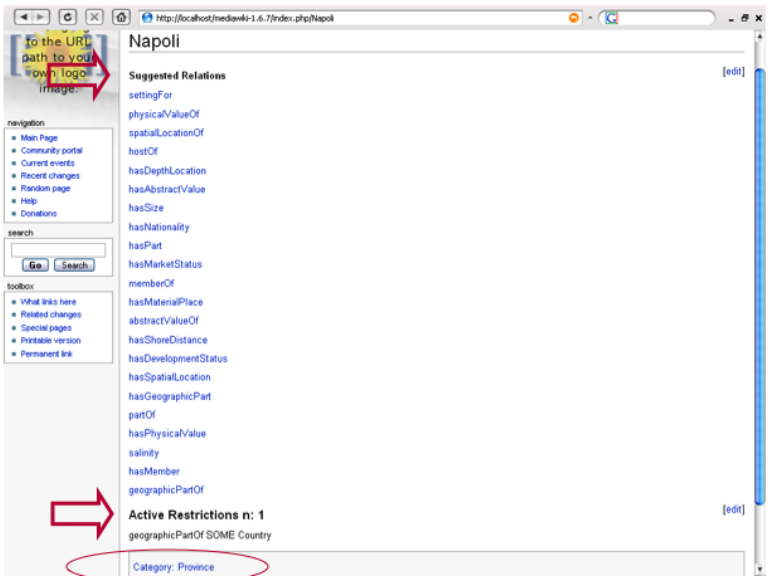

Figure 3.4: New Individual page details

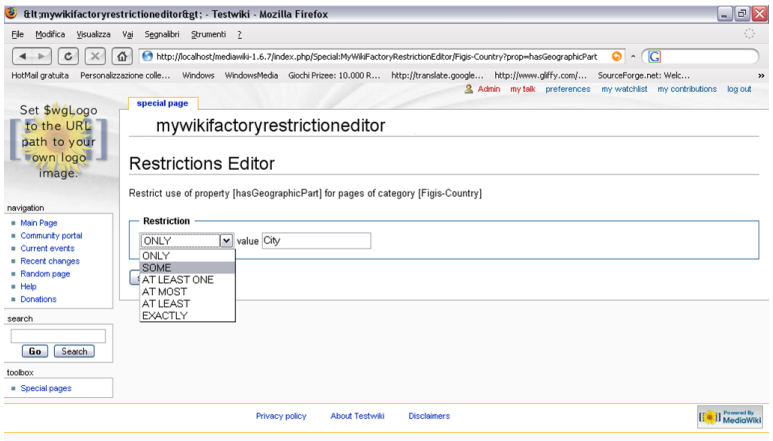

Figure 3.5: Restriction Editor page

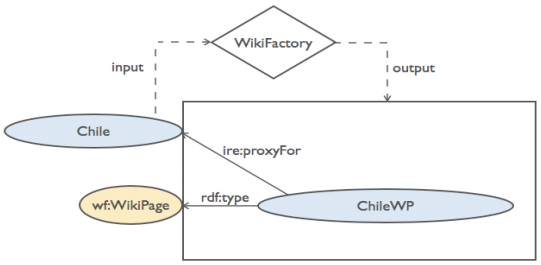

Figure 3.6: IRE Individual representation

As briefly stated above, WikiFactory implements the IRE [GP07] and DnS [GM03] models to control the generation and the coupling of the OWL code for both the domain ontology and the wiki structure ontology. The generated OWL code depends on the type of the ontology element, and describes the web element (e. g. a wiki page) representing that specific ontology element.

WikiFactory generates and maintains two OWL ontologies, namely the domain ontology and the structure ontology. The structure ontology is filled e. g. with individuals of type `wf:WikiPage`, a class which represents wiki pages. For example, consider the individual `Chile` in the domain ontology. As shown in Figure 3.6, WikiFactory generates an OWL individual named `ChileWP` which is of type `wf:WikiPage`. Furthermore, WikiFactory adds an assertion to the ontology, which states that the information contained in the wiki page `ChileWP` is about the individual `Chile` of the domain ontology i. e., `ChileWP ire:proxyFor Chile`. The relation `ire:proxyFor` is defined in the IRE ontology [GP07], and its rationale is motivated by the need to distinguish between references to the information structure and references to the domain, in order to enable correct identification of resources on the web.

Consider the class `Figis-Country` from the test domain ontology. In order to represent the domain concept expressed by such class and the structure element associated with it, it would be needed to relate the class i. e., `Figis-Country`, to an individual i. e., `Figis-CountryWP`. This statement would be in OWL Full. The same applies when object properties e. g., `memberOf`, are related to wiki page individuals e. g., `memberOfWP`. Figures 3.7, and 3.8 show how WikiFactory deals with this issue. The ontology writer reifies classes to individuals of the class `edns:Concept`, and object properties to individuals of class `edns:Description`. these classes are defined in the DnS [GM03] ontology. In the test case, the class `Figis-Country` is reified as the `Figis-CountryConcept` individual, while the object property `memberOf` is reified as the `memberOfDescription` individual. `Figis-CountryWP` and `memberOfWP` are respectively related to them by means of the `ire:proxyFor` object property. In order to identify which concept is related to which class, an `owl:hasValue` restriction is used. For descriptions, WikiFactory maintains an annotation property, namely `wf:expressedByAnnotation`, which allows the tool to identify the reified object property.

3.3 Related Work

WikiFactory can be compared to other work by focusing on its functionalities:

- automatic model-driven deployment of semantic wiki sites (wiki-platform independence);
- synchronization between wiki content and ontology;
- user-oriented functionalities for exploiting semantic features;
- support for collaborative maintenance of web ontologies.

WikiFactory contribution is then characterized with reference to model-driven solutions for semantic web portal management, to collaborative aspects of ontology design, and to semantic wikis.

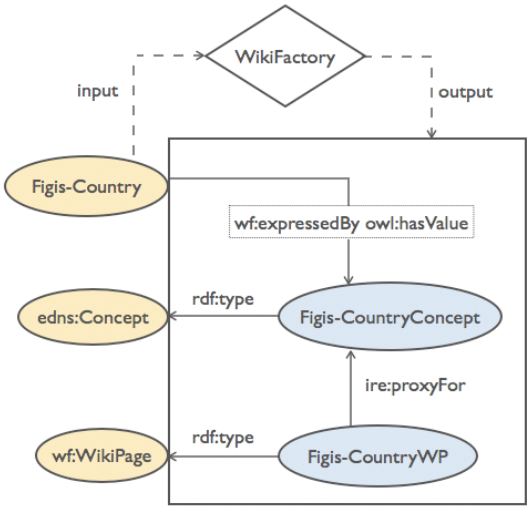

Figure 3.7: IRE Class representation

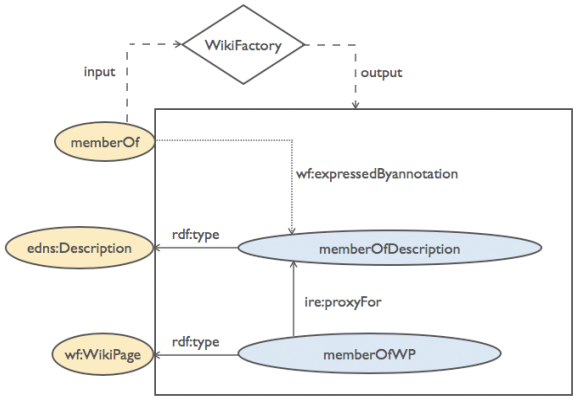

Figure 3.8: IRE Property representation

WebML[CFB00] is the most relevant example of model-driven web portal development. WebML has a notation for specifying complex web sites at the conceptual level. The WebML approach allows users to generate and deploy large web sites with a development process that undergoes several cycles, each of them producing a prototype or a partial version of the application, which allows conducting testing and evaluation since the early development phases. WebML users are mainly web designers, who can apply a software development process for the creation of a complex web portal. However, WebML is not intended neither for collaborative evolution nor for synchronization with a semantic knowledge base, as WikiFactory does.

There is valuable research effort on how to deal with collaborative ontology creation and growth. For an extended discussion of the state-of-art, which spans from social to technological aspects of this issue, the reader can refer to [CGL+06]. For example, some approaches address social aspects to be considered when building tools [Ack01], [MS06]. [Ack01] clearly states the problems coming from the social-technical divide. [Kol96] is a brief survey of the main studies on principles that seem to underline successful cooperative communities. As far as ontology design is concerned, the DILIGENT methodology provides several requirements for supporting collaborative workflows [Vea06], [Tem06], and deals with argumentation [STV+06]. In [NCLM06], further requirements for collaborative ontology development are identified. Available tools include e. g. Ontology Builder and Ontology Server (OBOS) [DWM01], and Co-OPR [Sea06], which presents the integration of two existing tools (i. e., Compendium, and I-X). [Lu03] is a source of interesting points on requirements and tool support, while [SSN93] and [Cha01] analyze aspects of human-computer interaction (HCI). These contributions mainly address ontology engineers' requirements. Our approach targets the collaboration between teams of domain experts and ontology engineers, and the synchronization between domain and structure ontologies, which enables the customization of the web elements (wiki pages or other), based on the components of the collaborative lifecycle.

Wiki applications (the so-called WikiClones, to stress that they are derived from the original WikiWikiWeb application [Web]) share the same basic philosophy as open editing and simple text-based syntax, but are different on the side of additional services and application modules they offer. These features cover the most varied areas: PurpleWiki [KEE] provides a full control on content fragments, JotSpot[Inc] integrates collaborative tools, SnipSnap [JMLSSJ] allows users to in-line organigrams and UML diagrams. Particularly interesting in the context of our work are those projects that aim at integrating wikis with Semantic Web technologies, known as "SemanticWikiWikiWebs" [Sem].

Probably, the most popular semantic wiki is Semantic MediaWiki [Wik] (a MediaWiki [med] extension), which provides users with semantic features such as defining categories, relations and articles, which correspond to classes, properties, and individuals respectively. Furthermore, it provides functionalities for importing and exporting OWL ontologies. Although Semantic MediaWiki allows users to define queries by supporting a

subset of SPARQL [PS05], it does not provide any reasoning yet.

OntoWiki [HBS05] is a semantic wiki with facilities for the visual presentation of a knowledge base, which is represented as an information map with different views on instance data. In addition, it enhances the browsing and retrieval of data of the knowledge base by offering semantic search strategies.

WikiFactory differs from the above mentioned semantic wikis. First, it is not a semantic wiki; rather it is wiki-platform independent. WikiFactory is designed with a plug-in based architecture, i. e., it is possible to plug-in modules for specific wiki platforms. In addition, WikiFactory is a server application, which automatically enables the ontology-driven generation of a semantic wiki, and enriches the resulting semantic wiki with additional semantic features. Moreover, it maintains the synchronization between the underlying ontology and the wiki content.

3.4 Concluding Remarks

This paper has discussed the main functionalities of a software framework we have designed and implemented, called WikiFactory. We have exemplified two of its main features: the automatic ontology-driven deployment of a semantic wiki, and the run-time ontology-to-wiki synchronization. Through WikiFactory, domain experts from FAO case study described in Section 2.1 can browse their ontologies, and update it in a collaborative way. Ontology engineers collaborate with domain experts and all updates are immediately reflected on the ontology.

Ongoing work includes the experimental setting and execution of a user study to evaluate the actual improvement, with respect to existing technologies, on user satisfaction and economic impact, of collaborative creation, validation, and evolution of ontologies with WikiFactory. The user study is planned with the following characteristics:

- A team of AGROVOC [agr] experts from FAO will be split into two groups, say *A* and *B*.
- The OWL version of the AGROVOC [SLL$^+$05] ontology will be deployed on a semantic wiki by using WikiFactory.
- Team *A* will validate the OWL version of AGROVOC by using WikiFactory. They will directly implement their changes from the wiki pages and will be provided with support by ontology engineers using the same wiki as a means for collaboration, and possibly using other editors and tools with the domain ontology synchronized by WikiFactory.
- Team *B* will validate the OWL version of AGROVOC by using the workflow described in Section 2.1.1.
- A time interval *t* will be defined, after which the two teams will be swapped. After an additional *t*, the experiment will stop.

The user study will allow us to have a quality evaluation from users, based on the comparison between the two methods. Ontology engineers will validate the resulting modified ontologies, in order to evaluate the correctness of the changes. In addition, further future work includes:

- Realization of an environment for argumentation support with WikiFactory;
- The implementation of a repository of ontology design patterns, publicly maintained with WkiFactory;
- An extended support to more complex OWL constructs (e. g., complex OWL restriction types).

Chapter 4

Argumentation

Creating and designing an ontology is a complex task that requires the collaboration of domain and ontology engineering experts (cf. Section 2.1). For coming to a consensual model of a domain that is expressed by an ontology, the participants in the engineering process must discuss their different viewpoints in an efficient manner. Thus, discussions are an important part of collaborative ontology engineering.

In Section 4.1, we will present the idea of issue based information systems (IBIS) and how they provide benefits in discussion and decision taking processes. After that in Section 4.2 and 4.3, we will present the Compendium tool and the Cicero tool which both implement the idea of IBIS. As we will show in Section 4.4, they are best suited for complementary scenarios: Compendium has its strengths in documenting group meetings, Cicero is better suited for asynchronous collaboration.

4.1 Issue Based Information Systems (IBIS)

The original idea of Issue Based Information Systems was proposed by Horst Rittel and colleagues at the beginning of the 1970s (see [KR70] and [RW73]). The main purpose of IBIS is to support the decision making process for wicked problems. Wicked problems have the following properties:

- During problem formulation it is difficult to identify the requirements for a good solution and the relevant influencing factors. Only by developing possible solutions one gets a sufficient understanding of the problem for identifying the requirements and influence factors.
- There are no wrong or correct solutions but only better or worse solutions.
- It is not possible to try several solution proposals out and then select the best one (e. g. because of the costs of realizing a single solution).
- The problem is unique to the extent that previously developed solutions can not be adapted to the new problem.

During collaborative decision making processes, IBIS helps to structure the *issue* or problem and to simultaneously derive possible solutions with the help of discussions between the different stakeholders. A discussion in IBIS is centered around a topic for which a number of issues exist. While discussing an issue, the participants exchange their different perspectives on the problem and propose possible solutions.

In IBIS, a discussion starts with a first definition of the *issue* or problem and subsequently different solutions may be proposed by the participants. The solutions are then supported or objected by arguments. Furthermore, one may relate issues with each other. For example, one issue may generalize another issue or it may be a relevant analogy (i. e. arguments and solution proposals apply in an analogous way). By making such relations between issues explicit, the participants get a better understanding of the problem and the discussion is easier to follow.

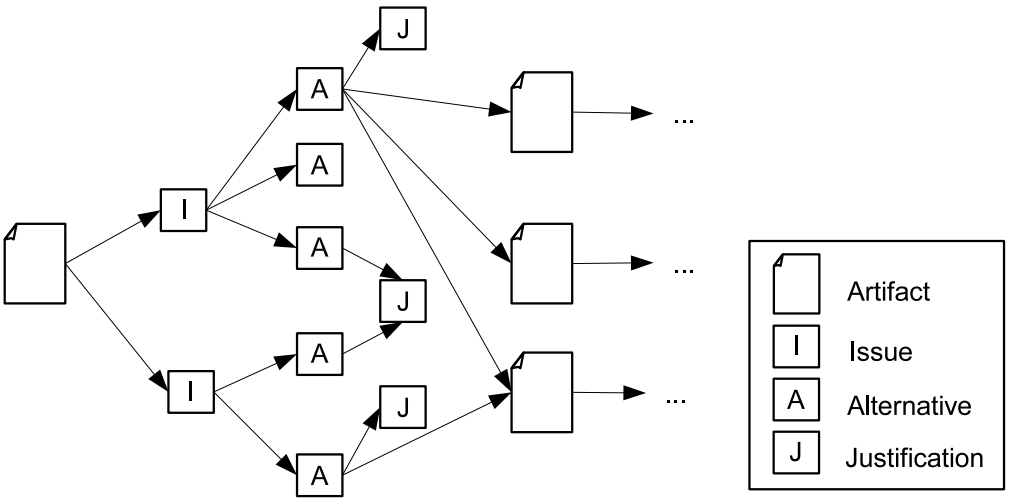

Figure 4.1: Documenting the design rationale with the Potts and Bruns model (adapted from [PB88]).

4.1.1 Extensions of IBIS

Originally, IBIS was designed for supporting the coordination and planning of political decision processes but the above listed properties of wicked problems also apply in other domains with project-oriented work. Thus, several extensions of the original IBIS approach were proposed, i. e. for adapting it to other domains or for extending its functionality.

In [PB88], Potts and Bruns adapt the IBIS approach to workflows in which a discussion usually leads to the creation or change of a concrete artifact. For example, in ontology engineering such an artifact might be an ontology element or a requirements document. In the Potts and Bruns model, the discussion serves as a connection between the old and the new version of a changed artifact (see Fig. 4.1). This helps to document the rationale of making certain changes to the artifact so that the changes are better understandable at a later point in time.

In [Lee90] and [Lee91], the quite simple Potts and Bruns model is extended by further elements to the *Decision Representation Language* (DLR). One important goal of the DLR was to keep the balance between human usability, machine usability and expressiveness. All three goals are addressed in DLR by introducing new elements that help to better express the structure of arguments. For example, DLR distinguishes supporting and denying justifications but it also introduces additional elements like *question* or *viewpoint*.

A further extension of IBIS is the *Procedural Hierarchy of Issues* (PHI) [McC79, McC91]. PHI supports the definition of a hierarchy of issues. In such a hierarchy, the solution of an issue depends on the solutions of its subissues. Also the arguments can be organized hierarchically. The hierarchical organization helps in dealing with a larger amount of issues and arguments in a single project. For example, in IBIS the users can typically handle 30-50 issues in a single project while in PHI usually 200-400 issues can be handled (cf. [DMMP06]).

In [MYBM91], MacLean and colleagues propose the *Questions, Options and Criteria* (QOC) approach that introduces *criteria* and *assessments*. The criteria are used for coming to a decision about which solution should be implemented for a certain issue. For example, such a criterion might be a desirable property of an ideal solution or a concrete requirement. Subsequently, all solutions of an issue are assessed in how far they fulfill the previously defined criteria. Criteria and assessments are in principle specializations of arguments in IBIS. However an important distinction between IBIS and QOC is that QOC requires the usage of the specialized arguments every time a decision is made. Thus, it helps the users to better structure the decision process to come to a decision more easily.

The *DILIGENT methodology* (see [PST04]) for collaboratively developing ontologies also proposes an argumentation framework that contains ideas from IBIS and the *Rhetorical Structure Theory* RST (see [MT87]).

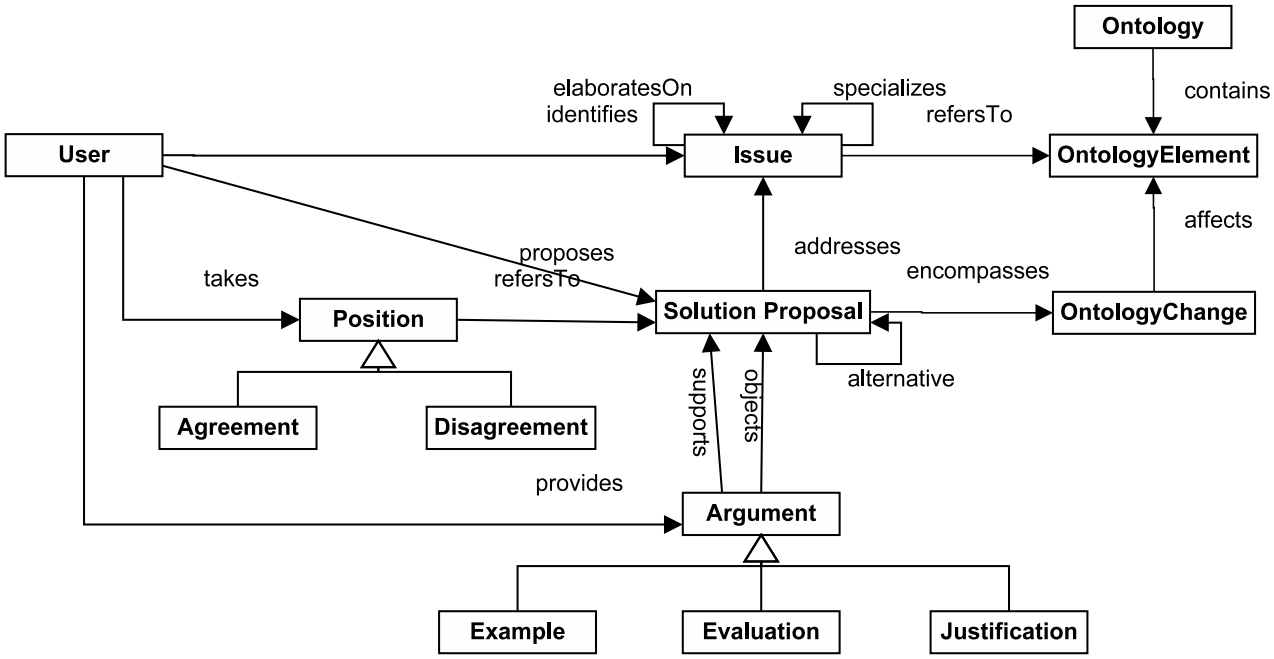

Figure 4.2: Adapted version of the DILIGENT argumentation model as it is used in Cicero.

From RST, the DILIGENT argumentation framework mainly takes a number of argument types. RST was originally used for annotating texts and for analyzing their argumentation structure. For this purpose, RST introduces different argument types like *justification*, *example* or an *elaboration* of previously said things.

In the context of a small case study, Pinto and colleagues showed in [PST04] first evidence that the argument types are to a varying extent useful for bringing a discussion to a successful conclusion. By asking the users of the argumentation framework to concentrate their discussion on the most effective argument types they were able to show that the participants of the case study needed less time for coming to a solution. For the Cicero argumentation tool (see Section 4.3), the DILIGENT argumentation framework was further simplified in order to make it more easily understandable and usable. The simplified argumentation model is shown in Fig. 4.2.

4.1.2 Benefits

Using IBIS or one of its extensions has benefits during as well as after the decision making process:

- **During the Decision Process**: According to the understanding of Rittel and his colleagues, the main benefit of IBIS is a better support of the decision making process (see [KR70]). For example, IBIS helps in structuring the current problem and thus in getting a better understanding of it. The better understanding may then lead to a better coverage of the different aspects of the problem and to additional solution proposals. Furthermore, IBIS and its extensions help in balancing the reasons for or against a certain solution proposal.

- **After the Decision Process**: A further benefit of IBIS and its extensions is the improved documentation of the decision making process. In conventional approaches often only the final decision is documented. All other alternative solutions proposals and the reasons for the decision are lost. But the knowledge about the other solutions and the reasons may be useful at a later point in time, e. g. if the requirements change and a new solution has to be devised (for an example from the software lifecycle see [BB06]).

Generally, one can distinguish more- and less-intrusive approaches for supporting decision making processes. The more-intrusive approaches like IBIS prescribe a certain structure for capturing a discussion which has to be followed by the user. Often, the more-intrusive approaches lead to a change in the way the users discuss with each other.

The goal of less-intrusive approaches is to restrict users in their discussions as little as possible. Instead of being prescriptive they are more descriptive. Thus, less-intrusive approaches are mainly suitable for documenting a discussion and their main benefit is after the decision process. In contrast the more-intrusive approaches they do not only document the discussion but they also try to increase the efficiency of the decision making process by changing the style of how the users discuss with each other (cf. [DMMP06]).

However depending on the current phase in the decision making process one can further differentiate. For example, in [HA06] they point out that during the initial collection of solution proposals a less focused or intrusive approach may be beneficial because otherwise there is an increased probability of overlooking possible solutions and concentrating on unimportant aspects of the problem. However when coming to a decision about which solution should be implemented, a more-intrusive approach is actually beneficial as it helps in structuring and evaluating the different arguments for and against the solutions.

Since the initial proposal of IBIS in [KR70], several tools have been implemented for supporting users in applying IBIS or one of its extensions in their decision making processes. An overview of tools that were developed in the 1990s is available in [DMMP06]. In the following, we will only highlight two tools: First, in Section 4.2 we will describe Compendium. Compendium can be seen as the successor of the tools that have been developed in the 1990s as it incorporates many lessons learned from these tools. Second, in Section 4.3 we will present Cicero. Cicero is a web-based tool developed in the context of NeOn that supports the user in applying the DILIGENT argumentation framework.

4.2 Compendium

Compendium is a Java based tool that helps in applying the IBIS or the QOC approach on decision making processes. It incorporates several of the lessons learned from other IBIS and QOC tools that have been developed in the 1990s. It is continuously under further development by the Compendium Institute that offers it as an open source software.[1] The Compenidum Institute also offers training material for download and it regularly organizes workshops.

Compendium may be used by a single person but it can also be used for collaborating in a team. It helps in collecting issues, solution proposals and arguments that are then represented as a graph in which the users can easily add relations between the different elements and quickly get an overview of the discussion. Furthermore, Compendium allows for linking to external documents like web sites or word documents, e. g. for referencing relevant text passages that support an argument (see Fig. 4.3). The way of using Compendium depends on whether it is used for personal organization or for documenting and reflecting discussions in a group (see below).

4.2.1 Personal Organization with Compendium

Compendium can be used by a single user for organizing his tasks and documents. For example, it is possible to drag any document from the computer and drop it in Compendium. The user can then establish links to other documents and annotate it with ideas, arguments or decisions. Furthermore, the documents can be annotated with keywords or tags for flexibly organizing and structuring all objects that are relevant for a certain topic. For the previous applications of Compendium, the IBIS approach isn't so much in the foreground. Instead, Compendium is used more like a hypertext system that offers additional functionality that may alternatively be found in e. g. Mind Mapping tools.

[1] Compendium can be downloaded free of charge from the web site http://www.compendiuminstitute.org/.

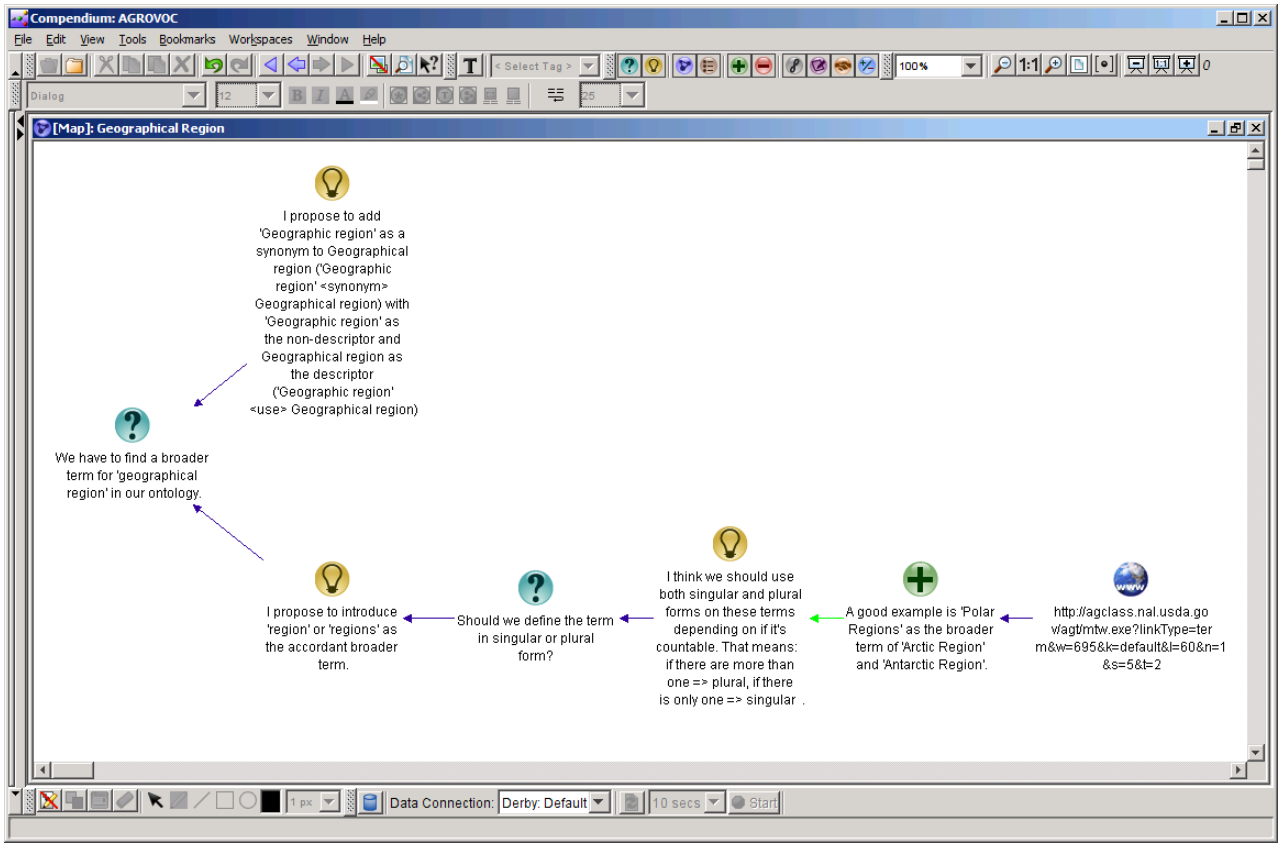

Figure 4.3: Example for a discussion in Compendium.

4.2.2 Documenting Group Discussions in Compendium

Compendium can also be used in a team for collaboratively developing solutions. Several scenarios exist for using Compendium in a team:

- **Synchronous Documentation of Group Meetings**: In this scenario, all participants of a discussion are together in one room or they may be connected via a video conference. A single, dedicated person, the *dialogue mapper*, is responsible for summarizing the discussed issues, solutions and arguments with the help of Compendium. For this purpose, he uses the structure that is proposed by IBIS. If a decision has to be made or if the meeting should be summarized, all participants recapitulate the most important points by looking at the documented discussion in Compendium (cf. [SSS+06]). This way, no idea for solving the issue is lost and decisions are made in a more objective manner.

- **Ex Post Documentation of Group Meetings**: The actual group meeting is recorded, e. g. its audio and/or video. After the meeting is finished, the most important issues, solutions and arguments are structured and summarized in Compendium. Each element in Compendium references to the relevant position in the recordings of the meeting so that one can quickly jump to it for recapitulating the details. In this scenario, Compendium has more the role of an index of the complete record of the meeting but it also provides a summary of the discussion.

- **Compendium as Asynchronous Groupware**: In this scenario, all participants run an instance of Compendium and the communication mainly takes place in Compendium. For this purpose, all instances of Compendium have read and write access on the same central database to which all changes are written, i. e. all instances work on the same copy of the discussion. It can be seen as a disadvantage of this scenario that each user can not only change and delete his own contributions

but also those of everyone else (e. g. if he thinks that a contribution is irrelevant or wrong). This might be problematic as it isn't possible to prevent the misuse of this feature within Compendium.

- **Compendium as Synchronous Groupware**: In this scenario, the communication between the participants of a discussion is also mediated by Compendium. If Compendium is used as a synchronous groupware, every participant works on his own copy of the discussion. They get notified by Compendium if another participant makes changes to his copy of the discussion. Upon receipt of such a change notification, each user has to explicitly agree that the change should also be applied in his copy of the discussion. Disadvantages of this scenario are: (1) all participants must be online at the same time for getting the change notifications and thus only synchronous collaboration is possible and (2) it is not ensured that every participant sees the same issues, solutions and arguments in his copy of the discussion. For example, the view on the discussion might differ if one participant didn't apply a change in his local copy.

4.2.3 Conclusion

Compendium is a mature tool that is based on many years of experience with capturing and documenting design and decision making processes. Furthermore, an active user community has formed around Compendium and it has been successfully used in different companies and organizations.[2]

All in all, Compendium allows for flexibly modeling a discussion. There is only very little guidance of the user by the tool on how to use it for applying the IBIS approach to capturing and documenting discussions. In order to use the tool in the intended manner one has to spend a larger effort for learning the usage of the tool (e. g. by working through the tutorials on the Compendium web site or by attending one of the Compendium workshops). Without the initially acquired knowledge about the IBIS or the QOC approach, one can compare the functionality of Compendium with hypertext systems or tools for creating Mind Maps.

4.3 Cicero

Cicero is a web-based tool that supports asynchronous discussions for collaborative decision making and design processes. The Cicero argumentation framework is further development of the DILIGENT argumentation framework described in Section 4.1.1. The Cicero argumentation framework is shown in Fig. 4.2. The goal of the further development was to simplify it without loosing its expressiveness. This way, the required learning effort for the users should be reduced. Besides the argumentation framework, Cicero also contains a functionality for actually deciding which of the solution proposals should be implemented.

To further reduce the required learning effort of new users, Cicero is developed as an extension of the MediaWiki and Semantic MediaWiki[3] that are already known to many users through Wikipedia. Thus, its look and feel as well as its use style should already be known to a larger group of users. Cicero is being developed as Open Source software by the ISWeb working group of the University Koblenz-Landau in the context of NeOn.[4]

In an installation of Cicero, several projects may be hosted in which issues can be discussed by the members of the project. For each issue and its related discussion, an overview page is created in Cicero that summarizes the issue and all currently proposed solutions (see Fig. 4.4). The content of the overview page is automatically generated and can not be edited by the users. For each overview page there also exists a discussion page to which new solution proposals or arguments can be posted (see Fig. 4.5).

[2] At http://www.compendiuminstitute.org/library/casestudies.htm several case studies with Compendium are available.

[3] http://ontoworld.org/wiki/Semantic_MediaWiki

[4] Further information about downloading and installing Cicero are available on the web site of the ISWeb working group at http://isweb.uni-koblenz.de/Research/Cicero/. Furthermore, a demo server is available at http://cicero.uni-koblenz.de/. The demo server can be used for exploring the functionality of Cicero as well as for hosting actual projects.

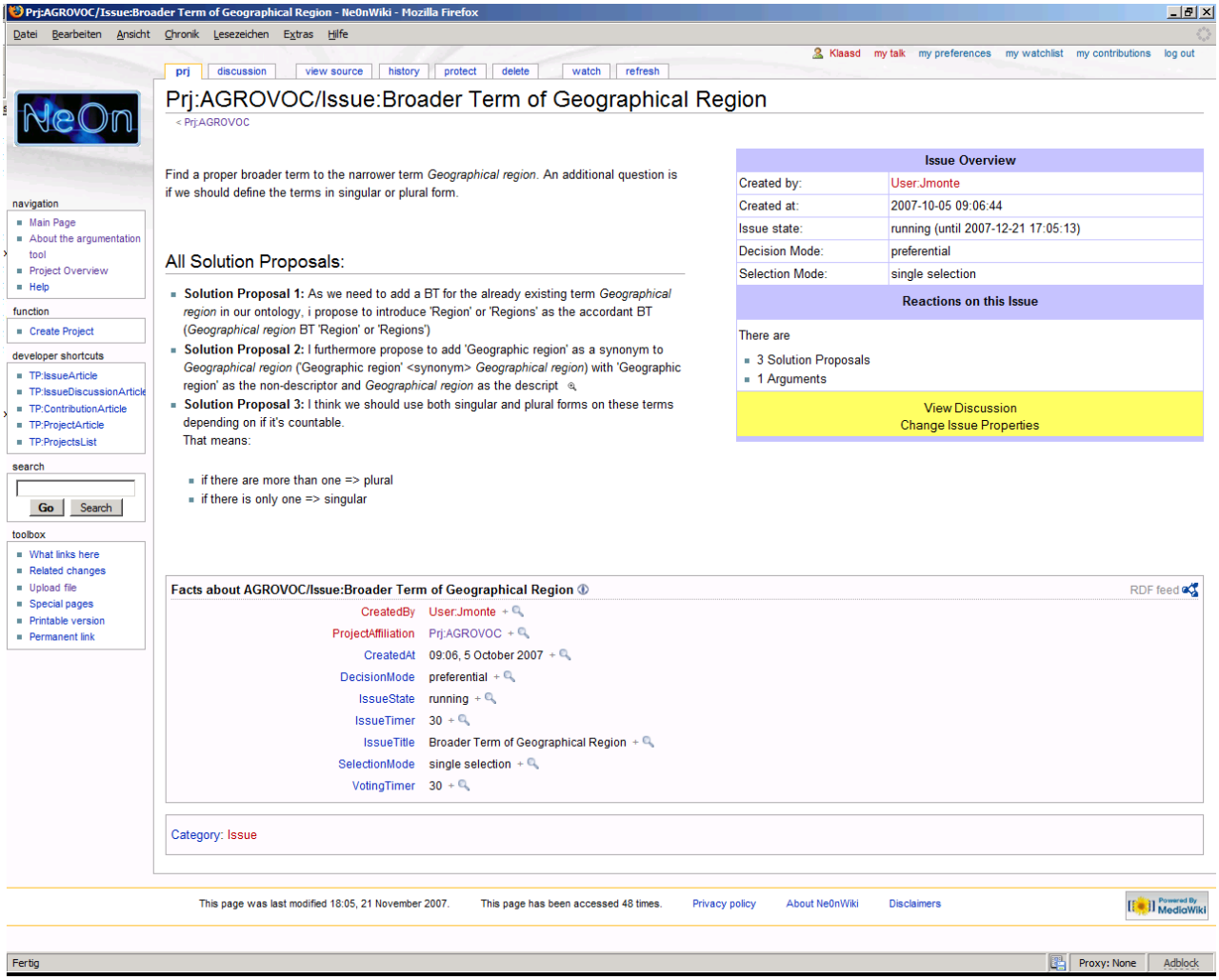

Figure 4.4: Overview page of an issue in Cicero.

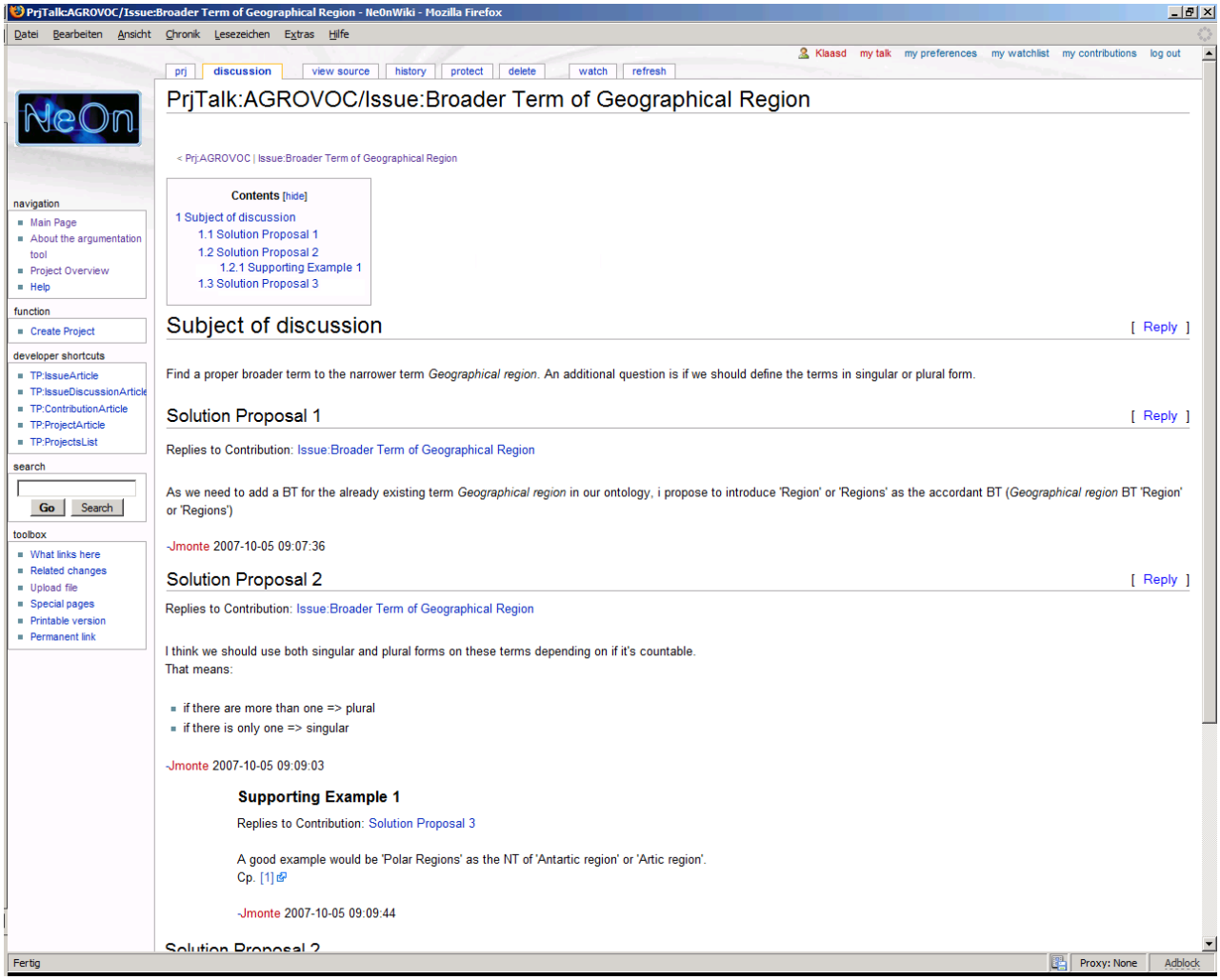

Figure 4.5: Discussion page in Cicero to which new solution proposals and arguments can be posted.

4.3.1 Asynchronous Group Discussions

In contrast with Compendium, Cicero is mainly designed for collaborative scenarios in which the participants of a discussion are separated in time and space. In these scenarios, regular meetings of all participants, possibly also in the same room, are in most cases no longer feasible. Instead, an asynchronous and computer-mediated communication, as it is provided by Cicero, is better suited.

Furthermore, Cicero allows for defining fine-grained access to the projects and discussions that are hosted on a single installation. For example, it is possible to give a certain group of users only read access to discussions while other groups may actively participate in the discussions or in the decision making process. The fine-grained access control allows for supporting different policies of user participation in a project. For example, in the case of developing an ontology, during which change requests and their possible solutions are discussed in Cicero, one may (1) give the users of the ontology only read access so that they are informed of ongoing changes or (2) one may give them the rights for actively participating in discussing possible solutions or (3) one may restrict all access to a smaller circle of developers.

Because Cicero is based on the MediaWiki software, it not only allows for discussing issues but it can also be used like a normal Wiki for collaboratively editing documents. If there is a need for discussion during the editing, then the extended functionality of Cicero may be used and the edited documents may be annotated with the discussions. It is also possible to integrate Cicero with other editing tools like the NeOn toolkit (see the following section).

4.3.2 Establishing Provenance Links

In [CGL+07b], the role of provenance in collaborative ontology engineering is described as “tracking the reasons why a change has occurred and to record the history of the design process” (p. 61), i. e. two main objectives of collecting provenance information can be identified:

- Documenting the history of changes to an ontology element.
- Documenting the design rationale of ontology elements.

The first objective of collecting provenance information is supported by version control systems like CVS, the framework for ontology evolution described in [KN03] or the ontology evolution framework proposed in WP1 of NeOn (see [PWHd07] for more details). The latter framework captures and logs ontology changes and stores them in an ontology change ontology. The collected information can be used for undoing or redoing previous operations and also for maintaining different variants of an ontology.

The second objective of documenting the design rationale can be supported by establishing a relationship between ontology entities and the corresponding discussion in Cicero that affected its design. For this purpose, a plugin for integrating Cicero and the NeOn toolkit is currently under development. The plugin will especially include the following functionality:

- Create issues from within the NeOn toolkit.
- Establish provenance links between issues created in the toolkit and the corresponding ontology elements.
- Allow for easily establishing provenance links between ontology changes made in the toolkit and solution proposals in Cicero.
- Allow for searching and filtering discussions:
  - Full text search in issues, solution proposals and arguments.
  - Filtering results for specific authors and dates.

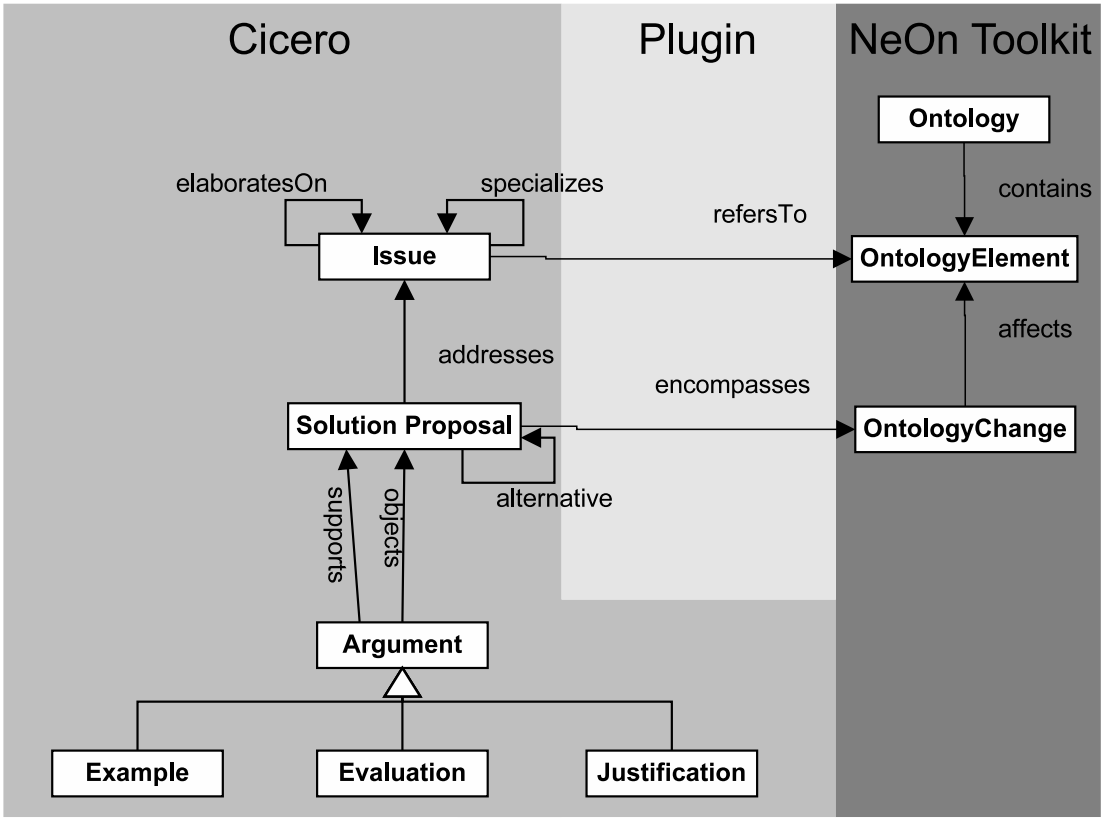

Figure 4.6: Mapping between tools and the parts of the argumentation model that they support.

- Filtering results for the status of the discussion (e. g. *running* or *decided*).
- Filtering results for discussions with a provenance link to specific ontology elements.

In Fig. 4.6, it is shown which parts of the Cicero argumentation model are supported by the respective tools. The actual discussions will be held in Cicero while editing and browsing the ontology will be supported by the NeOn toolkit. The plugin will be used for editing and browsing the provenance links between the ontology elements and the corresponding design rationale discussions in Cicero.

4.3.3 Conclusion

Cicero is mainly designed for collaborative scenarios in which participants are separated in time and space. It may for example be used for discussing the issues in a project, instead of e. g. mailing lists or discussion boards.

Compared to discussion boards, Cicero has the advantage that the specified structure of issues, solution proposals and arguments allows for more systematically developing solutions for a certain issue. Furthermore, the specified structure enables new users to more quickly get an overview of ongoing discussions, e. g. by reading the description of the issue and the proposed solutions first and then only reading arguments related to selected solution proposals. Finally, Cicero also offers mechanisms for coming to a decision e. g. by a voting of the participants or by only giving certain users the right to decide which solution should be implemented.

Compared to mailing lists, Cicero additionally has the advantage that it is easier to find a certain discussion topic and its related arguments and that the discussion isn't scattered in the inboxes of the discussion participants but centrally collected and documented.

	Compendium	Cicero
Approach	IBIS/QOC	DILIGENT
Main Application Area	Documentation of group discussions by a dialogue mapper.	Collaboration in teams that are separated in space and time.
Specific Features	Graphical representation of discussions. Linking to audio and video recordings of group meetings.	Flexible mechanism for access rights. Collaborative editing of documents.
License	Open Source	Open Source
Flexibility	++	○
Learnability	○	++
Communication Mode		
asynchronous	+	++
synchronous	○	not supported
group meeting	++	not supported
Provenance Links	no	yes

Table 4.1: Comparison of Compendium and Cicero.

4.4 Comparison of Compendium and Cicero

A summary of both tools is available in Tab. 4.1. Compendium and Cicero have their strengths in complementary collaboration scenarios. Thus, a decision for one of the tools should in the first place be dependent on the requirements of the existing scenario.

Compendium has its main application area in the synchronous or ex post documentation of group meetings. The documentation will usually be done by a specialized dialogue mapper. Compendium may also be used as a synchronous or asynchronous groupware but this is connected with certain restrictions and problems.

In contrast, Cicero has its main application area in scenarios where team members are separated in time and space and thus regular meetings are not feasible. The tool is then used for computer-mediated, asynchronous communication between the team members. Furthermore, its Wiki functionality may be used for collaboratively editing documents. In the future, an important advantage of using Cicero for collaborative ontology engineering will be its integration with the NeOn toolkit which leads to a better support of the ontology engineering lifecycle. Amongst others, the integration will reduce the required effort for establishing the provenance links between discussions of the design rationale and the affected ontology elements.

Compendium is all in all more flexible than Cicero with regard to modeling and capturing discussions. In contrast to Cicero, it doesn't enforce a certain approach e. g. that issues, solution proposals or arguments are related to each other in a certain way. The flexibility of Compendium can be seen as an advantage as well as a disadvantage: It can be seen as an advantage because the user isn't restricted in his possibilities and can thus be more creative in developing and documenting his ideas. But this can also be seen as a disadvantage because especially the IBIS approach (or one of its extensions) for structuring discussions should lead to a more systematical development and evaluation of solutions.

Compendium can also be used for applying the IBIS approach but the user isn't guided by the tool in its correct application. Instead, the user has to learn the approach either by attending a Compendium seminar or by working through the tutorials on the Compendium web site. Cicero in contrast offers more guidance for the users in applying the DILIGENT argumentation framework and in structuring the discussions. This helps to reduce the learning effort that a user has to initially spend but it also decreases the flexibility of the tool.

Chapter 5

Conclusion

In this deliverable, we identified a list of features that are required for supporting a collaborative ontology design process. The list is based on an analysis of the NeOn case studies and already existing tools with collaborative ontology design features. According to the feedback of the future users of the collaborative features, one can identify the following most important features:

- The annotation of ontology elements and ontology changes with e. g. scope notes or justifications for a specific change is one of the most important collaboration features. Annotations help to increase the awareness between the different editors of an ontology. Furthermore, they can be used for coordinating the collaborative workflow (e. g. by annotating status labels).
- Facilitating the communication between the editors of an ontology by means of asynchronous and/or synchronous communication media (e. g. forums or chats). An important objective of the communication support is to reduce the number of required face-to-face meetings for coordinating the collaborative editing. It is especially important if the editors of an ontology are spread all over the world, like in the FAO case study, or over a whole county, like in the invoicing case study.
- Furthermore, efficient means for logging ontology changes are required during collaborative editing of ontologies. Such a log can e. g. be used for maintaining different versions of an ontology and/or for undoing previously made changes. Especially an undo mechanism is expected to be very useful for the case study partners.

With the WikiFactory framework and the Cicero argumentation tool we also presented two concrete tools that address the collaborative editing, including mechanisms for logging and undoing changes, and the communication between the different editors of an ontology. Furthermore, we presented two concrete proposals of how to ease the process of attaching provenance annotations to ontology elements and ontology changes as well as how the social network of the editors can be analyzed.

Besides the techniques mentioned in this deliverable, there is also ongoing work in other NeOn deliverables that addresses collaborative aspects of ontology design. For example, in [PWHd07] a framework for ontology versioning and logging is presented as well as a framework for the propagation of ontology metadata annotations. Furthermore, in [QH08] it is proposed how to extend the collection of provenance information to ontology elements originating from an ontology learning step while in WP4 customized views and ontology access rights are treated.

Currently, plugins are under development that integrate the functionality of the WikiFactory framework and the Cicero argumentation tool into the NeOn toolkit. The WikiFactoryDeployer plugin[1] provides the automatic generation of semantic wikis (based on Semantic MediaWiki platform) from an ontology that is currently loaded in the NeOn toolkit. It can also deploy the ontology into an instance of the Cicero argumentation

[1] http://www.neon-toolkit.org/wiki/index.php/WikiFactoryDeployer

tool. For the Cicero plugin[2] it is currently planned to support the functionality described in Section 4.3.2 until August 2008.

[2] http://www.neon-toolkit.org/wiki/index.php/Cicero

Appendix A

Manual of the Cicero Argumentation Tool

A.1 About the Argumentation Tool

Cicero is a web-based tool which supports asynchronous discussions between several participants. This social software application is based on the idea of Issue Based Information Systems (IBIS) and the DILIGENT argumentation framework. The DILIGENT argumentation framework was adapted for Cicero in order to make it easier applicable on discussions and in order to reduce the learning effort by users.

The discussions in Cicero are organized in projects. It is possible to have different property values and access rights for the different projects. Thus, projects are really independent of each other. More details about how to create a project and define its properties and access rights is available in Section A.4.

In Cicero, a discussion starts with the issue that should be discussed. For this issue, several solutions can be proposed. The solutions proposals can then be discussed with the help of supporting or objecting arguments. Furthermore, Cicero also defines a workflow for coming to a decision. For this purpose, it offers different decision procedures like preferential voting or that a responsible person is allowed for making decisions. More details about the discussion and decision process is available in Section A.5

A.2 Installation and Configuration

A.2.1 Installing Cicero

To install Cicero you need to execute the following steps:

1. Install[1] or upgrade[2] to MediaWiki 1.11 (or higher).

   Note that Cicero 1.00 has been tested with MediaWiki 1.11 and doesn't work with versions of MediaWiki prior to 1.11; it is also NOT guaranteed that Cicero 1.00 will work with versions of MediaWiki higher than 1.11.

2. Install[3] the MediaWiki extension Semantic MediaWiki 1.0 RC2 (or higher).

   Note that Cicero 1.00 has been tested with Semantic MediaWiki 1.0 RC2 and doesn't work with versions of Semantic MediaWiki prior to 1.0 RC2; it is also NOT guaranteed that Cicero 1.00 will work with versions of Semantic MediaWiki higher than 1.0 RC2.

3. Extract the downloaded archive into the folder *<MediaWikiPath>/extensions/*. Note that after extracting the archive, you will have a new folder called *DILIGENTArgumentationTool* which you should NOT rename.

[1] See http://www.mediawiki.org/wiki/Installation
[2] See http://www.mediawiki.org/wiki/Manual:Upgrading
[3] See http://ontoworld.org/wiki/Semantic_MediaWiki

4. Add the following line at the end of *LocalSettings.php* of your MediaWiki-installation:

```
include_once("extensions/DILIGENTArgumentationTool/DAT_DILIGENTArgumentationTool.php");
```

5. By calling the page

```
http://<serverName>/<MediaWikiFolder>/index.php?title=Special:DAT_InstallForm
```

Note that you need a *Sysops*-account within MediaWiki to be able to access the installation form. Here you just have to click on the *Install Cicero 1.00*-button and the installation is executed automatically (see area 1 in Fig. A.1). Make sure that in the *LocalSettings.php* of MediaWiki a database user account is listed which has the right to create and alter tables.

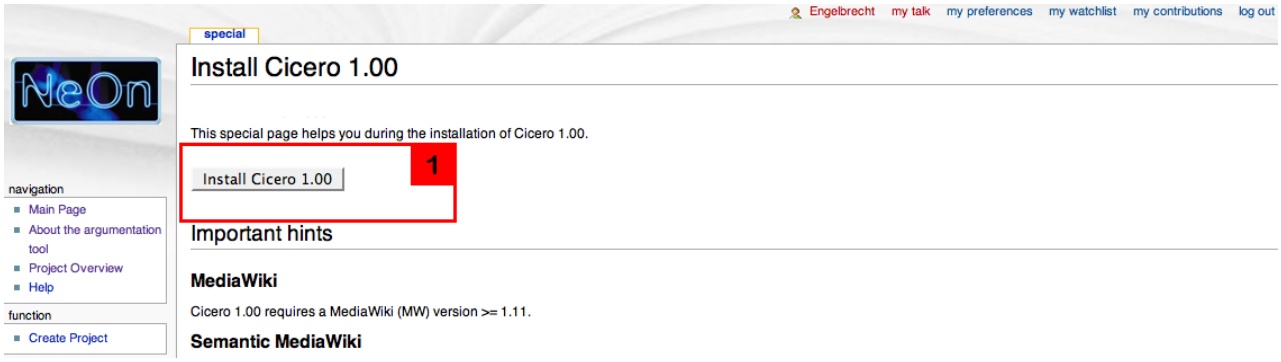

Figure A.1: The Cicero installation form.

Note that - if for some reason - the installation becomes incomplete or defective you can execute the installation procedure at any time. The installation procedure will add the missing parts of the installation and will repair the broken parts.
The user who installed the Cicero extension gets automatically assigned the role of a *Cicero administrator*. In Subsection A.2.2, more details are available about the *Cicero administrator*-role and how to assign it to further users.

A.2.2 Configuration

User groups

Predefined rights settings

The definition of the user groups used in Cicero are stored in the file *UserManagement/DAT_UserRights.php*. Usually, it will not be necessary to manually change this file. It is used for changing the predefined *public* and *private* rights-configurations described in Subsection A.4.2. They can be changed by adapting the values in lines 25-88. The following lines

```
$datGroupPermissions["project member"]['changeprojectproperties']['public'] = false;
$datGroupPermissions["project member"]['changeprojectproperties']['private'] = false;
$datGroupPermissions["project member"]['createissue']['public'] = true;
$datGroupPermissions["project member"]['createissue']['private'] = false;
...
```

show an excerpt. For example if you prefer that in the predefined configuration *public* a project member should not be able to create an issue, you need to change line 3 to

```
$datGroupPermissions["project member"]['createissue']['public'] = false;
```

$datGroupPermissions has the following structure

```
$datGroupPermissions[<roleName>][<rightName>][<predefinedConfigurationName] = {true or false}
```

true stands for *allowed* while *false* means *forbidden*. Be sure to keep the hierarchical structure of the user groups and the inheritance rule (for more details see Subsection A.4.2). Changes to this file will not have an effect on the user rights of already created projects.

Add new Cicero administrator

For registering a plain user as a *Cicero administrator*, the *User rights management* page of the MediaWiki can be used. The page can be accessed under the URL

```
http://<serverName>/<MediaWikiFolder>/index.php?title=Special:Userrights
```

Note that you need to be (at least) member of the *Bureaucrats*-group to access this page. There you can type the loginname of the respective user. After clicking on the *Edit User Groups*-button (area 1 in Fig. A.2) an interface appears below. It shows on the left the current memberships of the user (area 2). If the entry *Cicero admin* already exists, the user already is a cicero administrator. But if the entry *Cicero admin* appears on the right side (area 3), the user needs to be added to this group. You achieve this by selecting the entry *Cicero admin* on the right side and clicking on the *Save User Groups*-button (area 4).

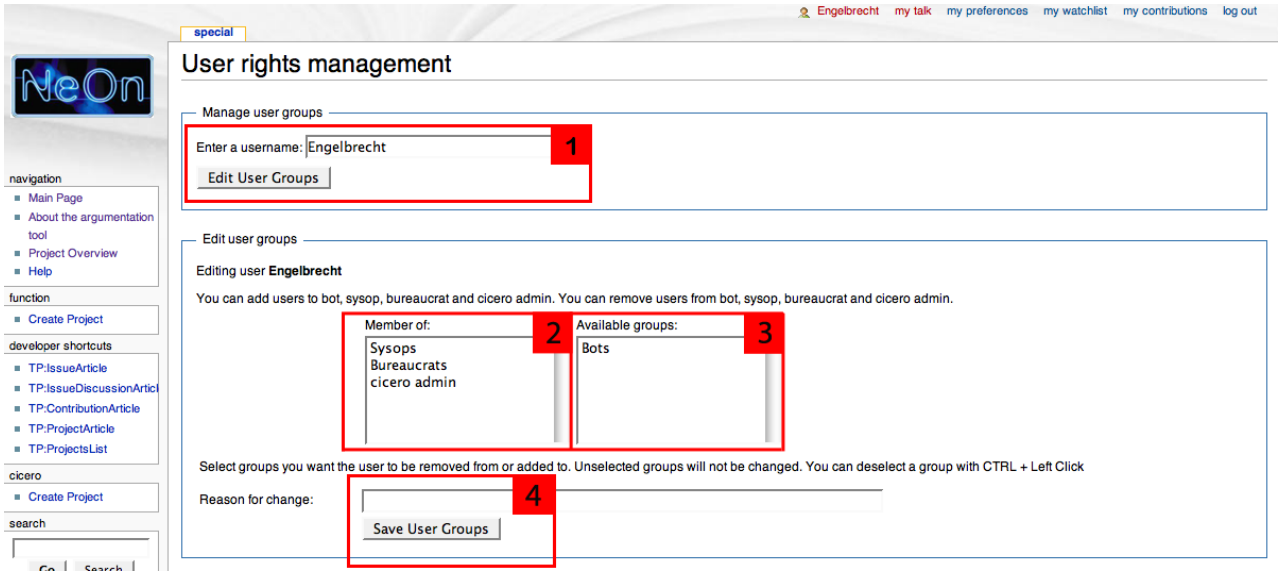

Figure A.2: Adding a new cicero administrator

Settings

The settings of the Cicero extension are mainly stored in the file *DAT_Settings.php*. You should normally NOT change any of the values. If you have other MediaWiki extensions installed besides of Cicero and *SemanticMediaWiki* which uses own namespaces, you probably need to change the following lines (34-37):

```
### NAMESPACES ###
// namespace numbers for this extension
$namespaceNr = 102;
$namespaceDiscussionNr = 103;
```

If another extension already uses the namespace numbers *102* and *103*, change the values in *DAT_Settings.php* into values unequal to these and bigger than *110*.

If you want to deactivate the creation of accounts by users, you just need to out-comment the following line (64) by adding // at the beginning:

```
$datGroupPermissions['*']['createaccount'] = true;
```

With this you restore the default behavior of your MediaWiki-installation.

If you want to change the default issue and voting timer within a project, you can change the following lines (193-194):

```
$defaultIssueTimer = 0;
$defaultVotingTimer = 7;
```

The predefined values are *0* for the default issue timer and *7* for the default voting timer. These values are shown in the interface for creating new projects (see Subsection A.4.1).

If you want to change the colors used on the project and issue article page, you can change the following lines (196-199):

```
### COLOR SETTINGS ###
$boxHeadlineBackground = "#C3C3FF";
$buttonAreaBackground = "#FFFF66";
$buttonAreaText = "#000000";
```

Note that you have to use the hexadecimal notation of the colors[4].

Activation of the email-notifications in MediaWiki

For activating email-notifications on your MediaWiki-installation you need to add the following lines to the *LocalSettings.php*:

```
#including external SMTP server
 $wgSMTP = array(
  "host" => 'examplehost.example.org',
  #"IDHost" => 'example.org',
  "port" => "25",
  #"auth" => true,
  #"debug" => false,
  #"username" => ,
  #"password" => ,
 );
```

Usually you only have to indicate the host-address and the port (normally 25) of your delivery SMTP server. Note that the lines with # are commented out and don't have an effect. Remove the leading # if the corresponding property needs to be set for configuring the access to your mail server.

A.3 Getting Access and Log In

After successfully installing MediaWiki, Semantic MediaWiki and the Cicero extension on your server, you can now start using the features of Cicero. The *Main page* of MediaWiki is the place where you get the first time in touch with the Cicero extension. Here, Cicero provides a dynamically updated list of all existing projects in its installation. This way it is easy to access the different projects.

To log into the Wiki press the link on the upper right corner of the main page. It will show you an interface to put in your user name and password. If you don't have an account yet, create one (click on the appropriate

[4] See http://tomheller.de/theholycymbal/html-farben.html

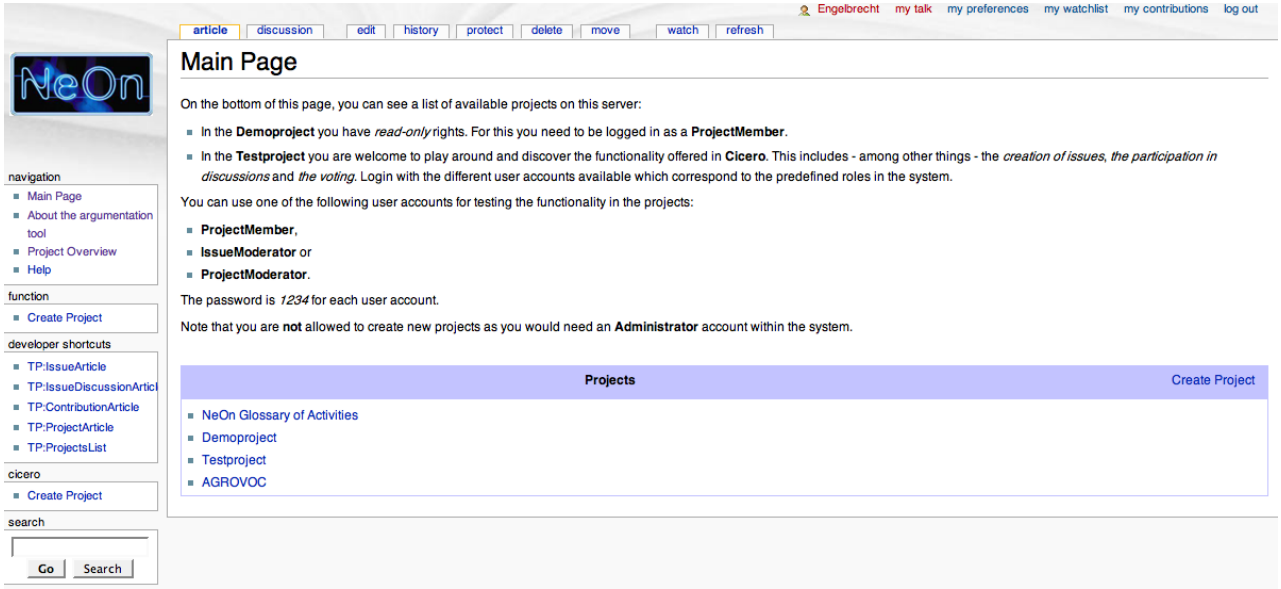

Figure A.3: Start page of Cicero.

link). While creating a new account you have to specify an email address. An email will be send to you containing a link for the verification of the new account. This address will also be used for notifications later on.

Note that in the default settings of MediaWiki users are not allowed to create their own user accounts. Normally a MediaWiki-user with the rights of a *Bureaucrat* needs to create them. In Cicero, this default setting of MediaWiki is overwritten, so that any user can create its own account. See Subsection A.2.2 if you want to restore the default behavior.

After you have successfully logged into the system you will be redirected to the main page. To log out use the corresponding link of the personal menu on the top of the page.

A.4 The Project Page

The overview page of a project can be reached from the main page of Cicero by using the links in the list of projects at the bottom of the main page (see Fig. A.3). The project overview page has three main areas that are shown in Fig. A.4:

1. A short introductory text that should summarize the project objectives. From here, also further pages with more detailed information may be linked. It should enable new users to get familiar with the project. The text may be changed in the settings of the project (see Subsection A.4.2).

2. An overview box with the most important properties of the project, e. g. who created the project, how many issues are currently attached to the project etc. At the bottom of the overview box, all functions that are available to the current user are shown. Functions for which the user does not have the necessary access rights are not shown (see Subsection A.4.2 for a summary of the access rights model of Cicero).

3. At the bottom of the page, two lists are available that contain the most recently modified issues and users participating in the project. For getting a list of all issues or users, the links in the yellow area have to be used.

The project overview page is automatically generated and updated by Cicero and need not be changed by the user. The introductory text of the project can be changed in the project properties (see Subsection A.4.2).

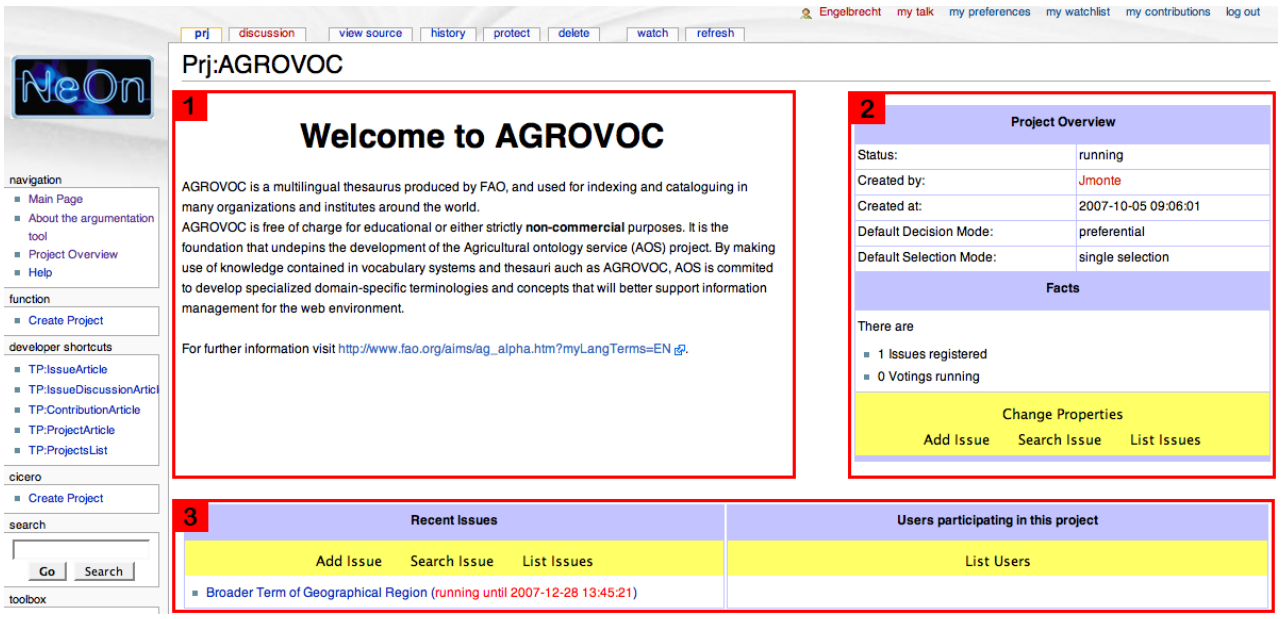

Figure A.4: Overview page of a project in Cicero.

A.4.1 Creating a Project

In Cicero, all issues and their discussions are related to a project. Thus, at least one project has to be created before one can start with creating and discussing issues. There are two possibilities to create a new project (see Fig. A.3):

- Using the *Create Project* link in the sidebar of Cicero. This link is accessible on each page of Cicero.
- Using the *Create Project* link next to the title of the project list on the start page of Cicero.

Only logged in users that have the *Cicero administrator* role can create projects (see Section A.2.1 for instructions how to assign users the *Cicero administrator* role). If a user has the sufficient rights for creating a project, he will see a form where the project can be configured. For more details on the configuration options of a project see Subsection A.4.2.

A.4.2 Managing the Project Properties

The properties of a project can be divided into two different blocks: (1) The description of the project, the advanced project settings, the default issue settings and (2) the management of the access rights of different user roles and the assignment of specific roles to the different users. The page for managing the project properties can be accessed from overview box on the overview page of the project (area 2 in Fig. A.4). The link is only visible to users with sufficient access rights (either an *Cicero administrator* or, in most cases, a *Project Moderator*).

Project Description, Advanced Project Settings and Default Issue Settings

In Fig. A.5, one can see the part of the project properties dialog in which the description of the project (area 1), the advanced project settings (area 2) and the default issue settings (area 3) can be edited. The description of the project will be shown on the overview page of the project (area 1 in Fig. A.4). It should give new participants in the project a short introduction to the project objectives etc. The text may contain Wiki markup for formatting, including links to subpages on which more details may be explained.

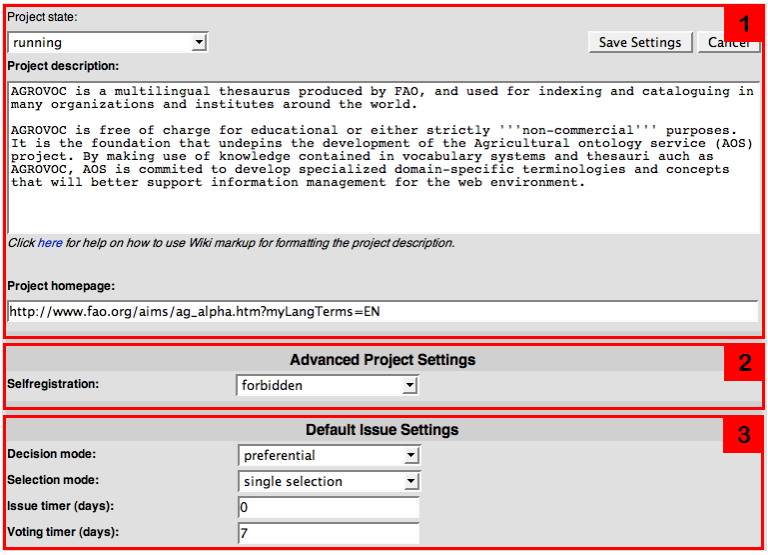

Figure A.5: Editing the project description and the advanced settings.

The advanced project settings contain the *Self-registration* option. If it is set to the value *allowed* then users that are currently not participating in the project may register himself for participation. Self-registered users are automatically assigned to the role *Project Member* (see below). This function is deactivated if the option is set to the value *forbidden*.

In the default issue settings, one may change the default values that are used if a new issue is created, i. e. which decision mode should be used, how many solution proposals may be selected during a decision and the values of the issue and the voting timer. The details about these settings are available in Subsection A.5.5 about managing the issue properties.

Managing Access Rights and User Roles

In Fig. A.6, one can see the part of the project properties dialog in which the user roles in the project can be edited (area 1) and assigned to the different participating users (area 2). The access rights in the different projects hosted on a single Cicero installation are independent of each other. Besides the *Cicero administrators* of the Wiki, in each project exist four predefined roles, to which different access rights can be assigned: The *Project Moderator*, the *Issue Moderator*, the *Project Member* and the *Anonymous User*.

User roles are ordered hierarchically. This means, a project moderator always has equivalent or more rights than an issue moderator and the issue moderator always has equivalent or more rights than a project member and so on. A Cicero administrator automatically has all access rights to a project and its related issues.

If a user can register himself for participating in a project, he is always assigned to the role of a project member (see Subsection A.4.2 for instructions how to activate the self-registration of users). Alternatively, one can explicitly assign the roles to different users (see area 2 in Fig. A.6).

Depending on the access and participation policies that should be implemented in a project, one can assign different access rights to the four roles. The following access rights exist in Cicero:

- **Read Discussion**: Allows access to the subpages of a project as well as to the overview pages of an issue and its discussions. Without this access right, a user can only see the start page of a project with the general information. Note that the title of issues are also shown to users which don't have the *Read Discussion*-right within a project.

- **Participate in Discussion**: Allows for actively participating in discussions of an issue, i. e. to provide solution proposals and arguments.

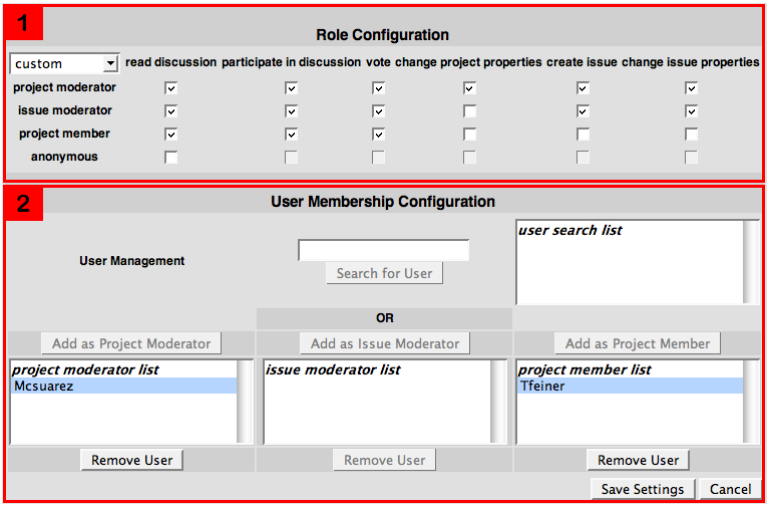

Figure A.6: Editing the access rights and user roles.

- **Vote**: Allows for participating in the decision taking process of an issue, i. e. to either participate in a voting or to select a solution proposal for implementation (see Subsection A.5.5 for more details on the decision taking process).
- **Change Project Properties**: Allows for managing the project properties as they are described in this section, including the access rights and user roles. Thus, this right should usually only be given to a very small and trusted group of users.
- **Create Issue**: Allows for raising new issues, i. e. creating a new issue overview page (see Subsection A.5.1).
- **Change Issue Properties**: Allows for accessing the issue properties as they are described in Subsection A.5.5).

Note that for the *Anonymous User* only the *Read Discussion*-right can be activated or deactivated. All other rights are not selectable in the interface and are automatically set to *deactivated*.
There exists two predefined configurations of the access rights that can be selected with the drop-down list in the upper-left corner of the role configuration panel (area 1 in Fig. A.6):

- **Public**: In a public project, all registered users are allowed for actively participating in discussions and the decision taking procedure as well as in raising new issues. Managing the project and issue properties are restricted to users that have assigned the role *project moderator* or *issue moderator*, respectively.
- **Private**: In a private project, the plain *project members* only have read access to the whole project and the related issues. Only *issue moderators* are allowed for participating in discussions and decision taking procedures as well as in raising new issues.

It is possible to use one of the pre-configured role configurations as the basis for a customized role configuration. Just select one of the pre-configurations and adapt it by adding or removing the marks in the checkboxes. Note that during checking or unchecking a box also the inherited access rights are automatically set for the more and less capable user groups (see above for details about the hierarchy of user groups). So if a right for a user group is activated, it will automatically be activated for all more capable user groups. Analogously, if a right for a user group is deactivated, it will automatically be deactivated for the less capable user groups.

In area 2 in Fig. A.6, the panel for assigning roles to the users of a project is shown. To assign a role to a user, one has to first search for his account in the database of the Wiki. A user can be searched by either his Wiki account name, his real name or his e-mail address. The latter two are only optional information that need not to be given by all users during their registration.

The search results are then shown in the *user search list* where one ore more users can be selected and then assigned to one of the three predefined roles by clicking on e. g. the *Add as Project Member* button. Users can also be removed from the different lists by clicking on the *Remove User* button under the respective list.

Note that the interface in the current version doesn't avoid that a user is added to more than one user group. You could for example add a user *McSuarez* to the *project moderator*- and the *issue moderator*-group at the same time. But the Cicero will only store the highest group membership, which means that for the user *McSuarez* only the *project moderator*-membership is considered, because a project moderator always has the same or more rights than an issue moderator.

In order to avoid the administrative overhead of manually adding all users that are plain *Project Members*, it is possible to enable the self-registration of users in a project (see Subsection A.4.2). They are then automatically added as project members. By default, the self-registration is deactivated.

A.5 The Issue Page

For each issue, an overview page and a discussion page exists. The overview page of an issue can be reached from the project overview page: Either the list of the 10 most recently added issues can be used for accessing an issue (see area 3 in Fig. A.4) or the options *Search for Issues* and *List all Issues* can be used (see area 2 and 3 in Fig. A.4).

The issue overview page has three main areas that are shown in Fig. A.7:

1. A description of the issue as it was entered during the creation.
2. A list of all solutions for the issue that were proposed up to now. If the description of a solution proposal is too long only the first 30 words are initially shown. One can expand the description to its full length by using the lens at the end of the corresponding description. This area also shows which solution is selected for implementation once a decision is taken (see Subsection A.5.4).
3. An overview box with the most important properties of the issue, e. g. who created the issue, how many solutions are proposed and how many arguments are given. Furthermore, one can access from here the full discussion associated with the issue or change the properties of the issue. In order to see the *Change Issue Properties* option, one needs to have the corresponding access right (see Subsection A.4.2).

The issue overview page is automatically generated and updated by Cicero and need not be changed by the user. The description and the settings of the issue can be changed in the issue properties (see Subsection A.5.5).

A.5.1 Creating an Issue

New issues can be added by using the corresponding option on the project overview page (see area 2 and 3 in Fig. A.4). In order to see the *Add New Issue* option, one needs to have the corresponding access right (see Subsection A.4.2).

Creating a new issue is a simple task in Cicero and can be done very fast. In the form for creating a new issue (see Fig. A.8) one only needs to enter a unique title for the issue and an initial description. The description may contain Wiki markup for formatting, including links to related issues or web pages.

The issue is then create by clicking on the *Save Issue* button. The settings with regard to the decision taking procedure and the issue and voting timer are set to the default values as they are specified in the project

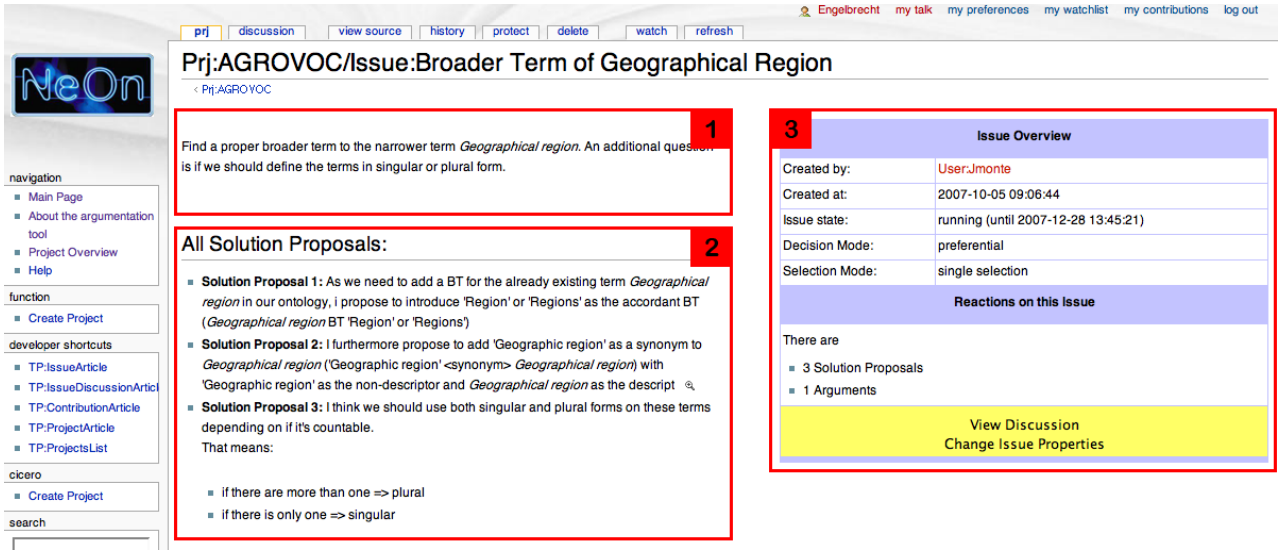

Figure A.7: Overview page of an issue in Cicero.

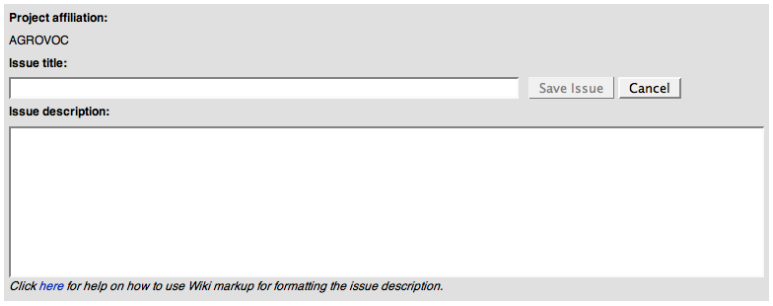

Figure A.8: Form for creating a new issue in Cicero.

settings (see Subsection A.4.2). The settings and the text describing an issue can later be changed on the page for managing the issue properties (see Subsection A.5.5).

A.5.2 Issue States

During its lifetime, an issue passes through four different states. Depending on the state, different changes to the issue are allowed. The states are summarized below and in Fig. A.9:

- **Running**: During the *running* state, all users with the corresponding access rights are allowed for making changes to the issues like adding further solution proposals or arguments.
- **Locked**: An issue can reach the *locked* state only if the dictator mode is chosen for decision taking (see Subsection A.5.4). During the *locked* state, no changes to the issue are allowed. Only a user with the corresponding access rights is allowed for deciding which solution proposal should be implemented as a response to the issue. As soon as the decision is taken, the state automatically changes to the *decided* mode.
- **Voting**: An issue can reach the *voting* state only if a preferential voting is chosen for decision taking (see Subsection A.5.4). During the *voting* state, no changes to the issue are allowed. All users with the corresponding access rights are allowed for casting their ballot. The voting is finished either after the time span set in the voting timer or it is manually finished by a user with the corresponding access rights. As soon as the decision is taken, the state automatically changes to the *decided* mode.
- **Decided**: As soon as a decision is taken which solution proposal should be implemented in response to the issue, the issue changes into the *decided* state. In this state, no changes to the issue are possible. If it should be further discussed, the issue has to be set back to the *running* state by a user with the corresponding access rights.

A.5.3 The Discussion Page

The discussion page is – as the name says – the place where the discussion of a certain issue is stored. It can be reached from the issue page either through the tab-bar at the top or through the *View Discussion* link in area 3 of the issue overview page (see Fig. A.7).

At the top of the page a table of content of the whole discussion is shown for quickly accessing specific solution proposals or arguments (see area 1 in Fig. A.10). Directly below the table of contents, the subject of discussion, i. e. the description of the issue, is repeated from the overview page.

Below the subject of discussion, the different solution proposals and their supporting or objecting arguments are listed. To make a contribution to the discussion, one has to use *Reply* link next to the corresponding heading to which it should refer. Two different kinds of contributions can be distinguished:

- **Solution Proposal**: As the name says, it proposes a possible solution of the current issue. During taking a decision, one can select one or more solution proposals for being implemented as a response to the issue (see Subsection A.5.4).
- **Argument**: In principle, an argument can either support or object a specific solution proposal. Three different types of arguments exist:
  - **Example**: An example corresponds to a pattern that should or should not be imitated (depending on whether its a supporting or objecting example). They are used for illustrating similar cases that may serve as a model for the solution proposal to which they reply.

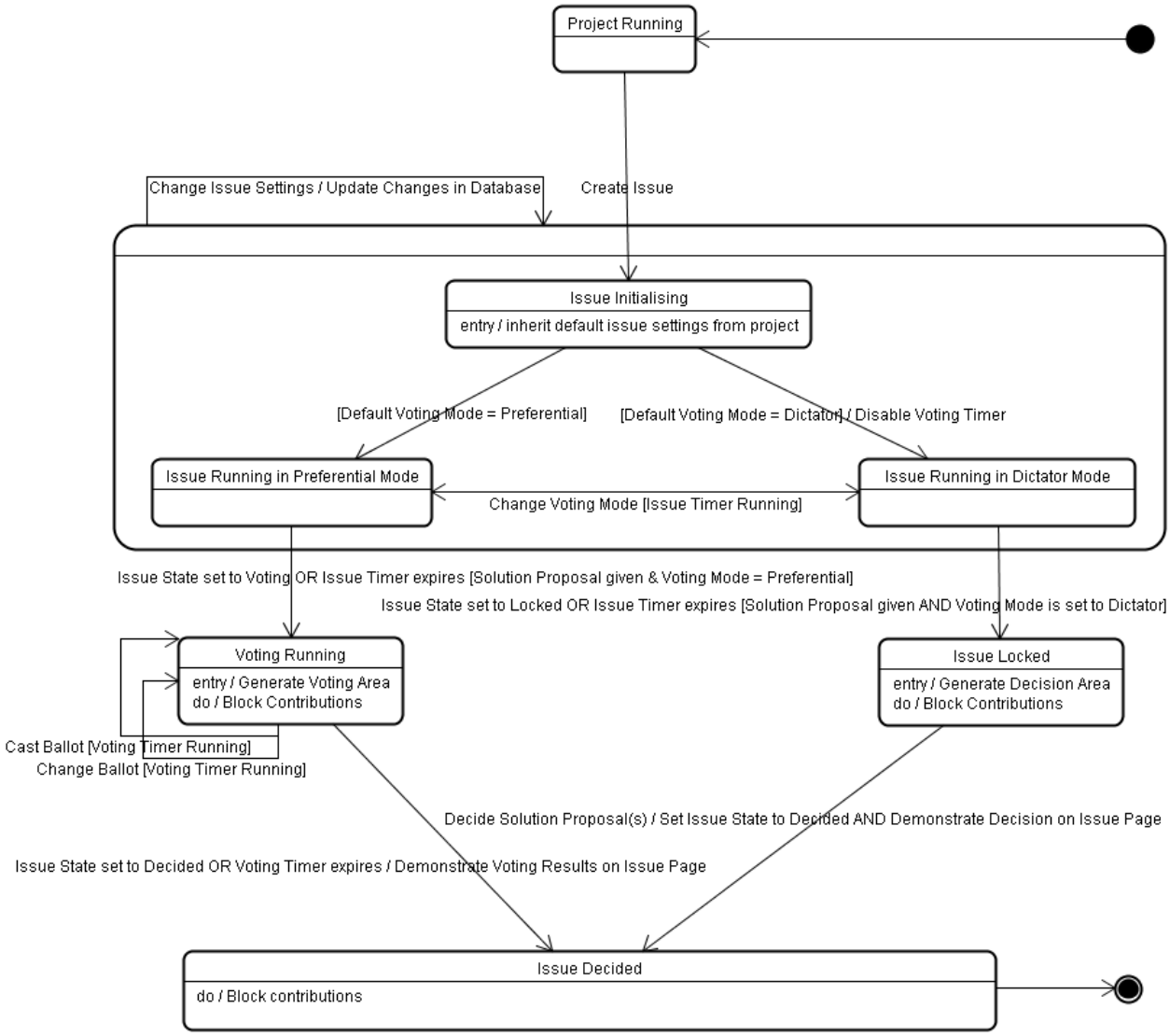

Figure A.9: Issue States in Cicero

Figure A.10: Discussion page of an issue in Cicero.

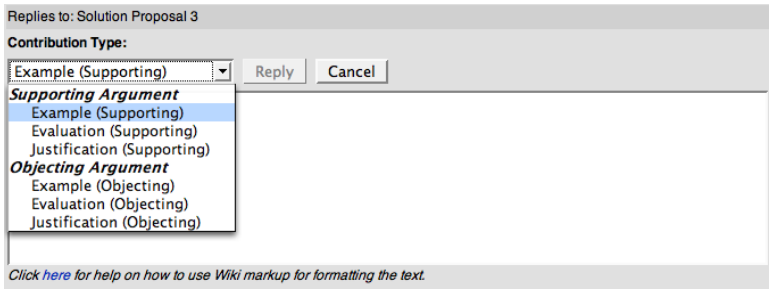

Figure A.11: Form for replying to a solution proposal.

– **Evaluation**: An evaluation gives criteria which help to assess the strengths and weaknesses of a solution proposal.

– **Justification**: A justification describes the relevant circumstances that help to understand why a certain solution is supported or objected by the author of the argument.

In Fig. A.11, the form for adding an argument to a solution proposal is shown. In the top left drop down list, one can select the argument type and whether it supports or objects the solution proposal to which it replies. In the box below, the argument text can be entered. The text may contain Wiki markup for formatting, including links to external resources or files uploaded to the Wiki.

The different kind of contributions and how they are related to each other can also be seen in Fig. A.12. One can see that solution proposals can only directly reply to the issue while the arguments can only directly reply to a solution proposal. This results in a very flat hierarchy, showing the arguments with a small indent to their solution proposals.

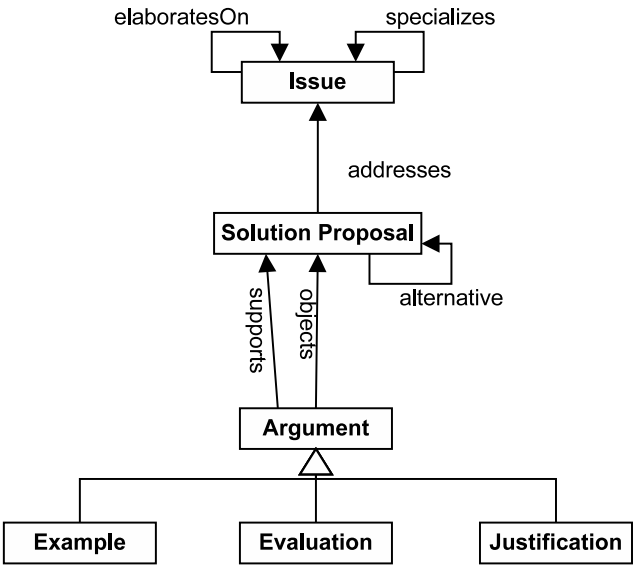

Figure A.12: Relations between issues, solution proposals and arguments.

A.5.4 Taking a Decision

If the decision taking procedure for an issue should be started, the state of the issue has to be changed from *running* to *voting* first. Under the precondition that there exists at least one solution proposal on the discussion page of an issue, two different ways exist how this state transition may take place:

1. In the issue settings, an automatic issue timer can be set that triggers that transition of the issue state from *running* to *voting*. By default, the issue timer is deactivated but in the project or issue settings a specific number of days may be given after which this transition takes place (see Subsection A.4.2 and A.5.5).
2. A user with the necessary access rights (see Subsection A.4.2) edits the issue properties and manually starts the voting phase for an issue.

Furthermore, two basic modes for taking a decision can be distinguished:

- **Preferential Mode**: In the voting phase of the preferential decision mode, all users with the corresponding access right can cast their ballot. Either automatically, by means of the voting timer determined in the issue settings (see Subsection A.5.5), or manually by an authorized user, the voting phase is closed after some time and the solution proposal with the most votes is marked as the decided solution. In case of a draw between two or more solution proposals a run-off ballot will start automatically. As soon as a final decision is available, the results are shown on the issue overview page and the state of the issue automatically changes to decided.
- **Dictator Mode**: In this mode, a user with the corresponding access rights locks the issue (see Subsection A.5.5). After that, he may go back to the issue overview page where a link to the page for taking a decision is shown. Once, the user made his decision he returns to the overview page where the result is shown and the issue state automatically changes to *decided*. Thus, the decision is only made by a single user.

Depending on the *selection mode* of the issue (see Subsection A.5.5), the users can either select only a single solution proposal during the decision taking phase or multiple solution proposals. In Fig. A.13 shows how the look of the issue overview page during a running preferential voting. By clicking on the button in the upper-left box, the user can change to the page shown in Fig. A.14 and cast his ballot. For the dictator mode both pages look very similar.

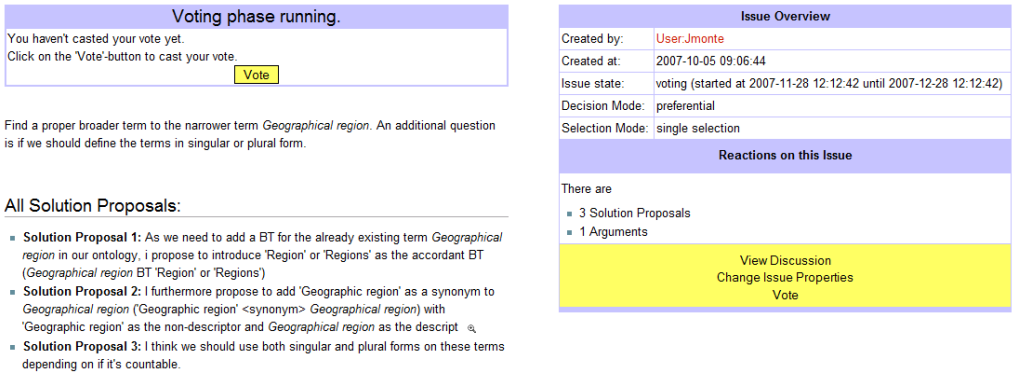

Figure A.13: Issue overview page during a running preferential voting.

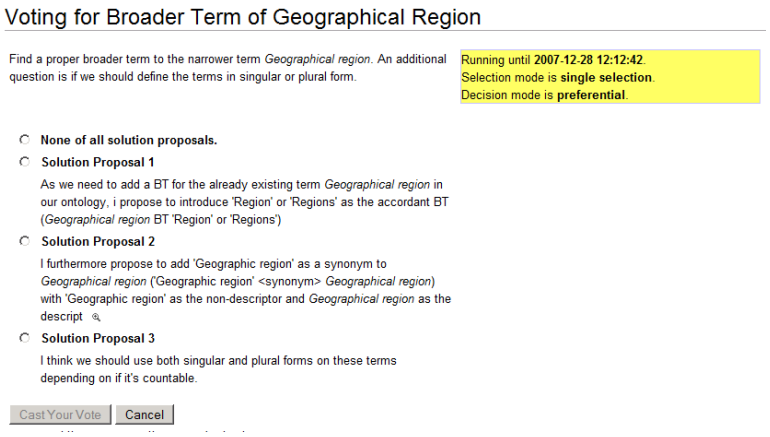

Figure A.14: Form in Cicero for casting the ballot during a voting.

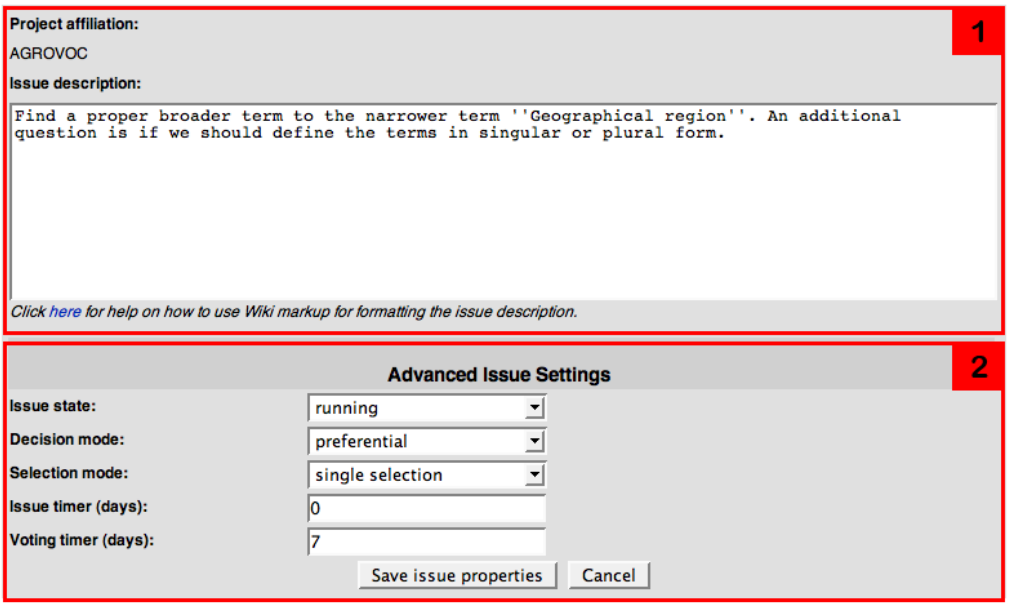

Figure A.15: Editing the issue properties in Cicero.

A.5.5 Managing the Issue Properties

The properties of an issue can be changed by choosing the corresponding option on the issue overview page (area 3 in Fig. A.7). This option is only available for users with the corresponding access rights. In the issue properties (see Fig. A.15), one can change the description of the issue (area 1) as well as its advanced settings (area 2). The description may contain Wiki markup for formatting, including links to related issues or web pages.

The values of the advanced issue settings are inherited from the project settings during the creation of the issue (see Subsection A.4.2). More details on the meaning of the different settings are available in the sections about the different state of an issue and the available decision taking procedures (see Subsection A.5.2 and A.5.4).

The issue timer and the voting timer can be used for automatically triggering issue state transitions. The issue timer gives the number of days after which an issue should automatically change from the *running* state into either the *voting* or *locked* state, depending on the chosen decision mode. The voting timer gives is only activated if preferential voting is chosen as the decision taking procedure. In this case, it gives the number of days after which the voting is automatically closed. Setting either of both time spans to a value of 0 corresponds to deactivating the automatic state transition, i. e. a user with the corresponding access rights has to manually change the state with the help of the *issue state* drop-down list.

Bibliography

[AB06] Ben Adida and Mark Birbeck. RDFa Primer 1.0 Embedding RDF in XHTML. Technical report, World Wide Web Consortium, May 2006. http://www.w3.org/TR/xhtml-rdfa-primer/.

[Ack01] M.S. Ackerman. . In John Carroll, editor, *HCI in the New Millennium*, 2001.

[ADR06] S. Auer, S. Dietzold, and T. Riechert. OntoWiki-A Tool for Social, Semantic Collaboration. *International Semantic Web Conference*, 4273:736–749, 2006.

[agr] Agrovoc. http://www.fao.org/agrovoc.

[ARJ96] Farquhar A., Fikes R., and Rice J. The ontolingua server: a tool for collaborative ontology construction. In B. Gaines, editor, *Proceedings of Knowledge Acquisition Workshop, Banff, 1996*, 1996.

[BB06] Janet E. Burge and David C. Brown. Rationale-based support for software maintenance. In Allen H. Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 273–296. Springer, 2006.

[BG06] Michel Buffa and Fabien Gandon. Sweetwiki: semantic web enabled technologies in Wiki. In *WikiSym '06: Proceedings of the 2006 international symposium on Wikis*, pages 69–78, New York, NY, USA, 2006. ACM Press.

[BRKT96] Swartout B., Patil R., Knight K., and Russ T. Toward distributed use of large-scale ontologies. In B. Gaines, editor, *Proceedings of Knowledge Acquisition Workshop, Banff, 1996*, 1996.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1*. W3C, 1.1 edition, March 2001. URL: http://www.w3c.org/TR/wsdl.

[CFB00] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. In *Proceedings of the 9th World Wide Web Conference (WWW9)*, pages 137–157, 2000.

[CGL+06] C. Catenacci, A. Gangemi, J. Lehmann, M. Nissim, and V. Presutti. Design rationales for collaborative development of networked ontologies. State of the art and the Collaborative Ontology Design Ontology. Deliverable d2.1.1 of the neon project, NeOn project, 2006.

[CGL+07a] Carola Catenacci, Aldo Gangemi, Jos Lehmann, Malvina Nissim, Valentina Presutti, Gerardo Steve, Nicola Guarino, Claudio Masolo, Holger Lewen, Klaas Dellschaft, and Marta Sabou. Design rationales for collaborative development of networked ontologies – state of the art and the collaborative ontology design ontology. Deliverable D2.1.1, NeOn Project, 2007.

[CGL+07b] Carola Catenacci, Aldo Gangemi, Jos Lehmann, Malvina Nissim, Valentina Presutti, Gerardo Steve, Nicola Guarino, Claudio Masolo, Holger Lewen, Klaas Dellschaft, and Marta Sabou. Design rationales for collaborative development of networked ontologies Ű- state of the art and the collaborative ontology design ontology. Deliverable D2.1.1, NeOn Project, 2007.

[Cha01] H.E. Chandler. The complexity of online groups: a case study of asynchronous collaboration. *ACM Journal of Computer Documentation*, 25(1):17–24, 2001.

[DMMP06] Allen H. Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech. Rationale management in software engineering: Concepts and techniques. In Allen H. Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 1–48. Springer, 2006.

[Dom98] J. Domingue. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, April 18th-23rd. Banff, Canada*, 1998.

[DS08] Klaas Dellschaft and Steffen Staab. Unterstützung und Dokumentation kollaborativer Entwurfs- und Entscheidungsprozesse. Technical Report 4/2008, Universität Koblenz-Landau, Arbeitsgruppe ISWeb, 3 2008.

[DSW+00] AJ DUINEVELD, R. STOTER, MR WEIDEN, B. KENEPA, and VR BENJAMINS. WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 52(6):1111–1133, 2000.

[DWM01] A. Das, W. Wand, and D.L. McGuinness. Industrial Strength Ontology Management. In *Proceedings of the International Semantic Web Working Symposium*, 2001.

[FFR97] A. Farquhar, R. Fikes, and J. Rice. Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computers Studies*, 46(6):707–727, 1997.

[GHM+06] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. Soap version 1.2 part 1: Messaging framework. Technical report, W3C, 2006.

[GM03] Aldo Gangemi and Peter Mika. Understanding the Semantic Web through Descriptions and Situations. In Robert Meersman, Zahir Tari, and Douglas Schmidt et al., editors, *On The Move 2003 Conferences (OTM2003)*. Springer Verlag, 2003.

[GP07] Aldo Gangemi and Valentina Presutti. A Grounded Ontology for Identity and Reference of Web Resources. In *Idenitity, Identifiers, Identifications – Entity-Centric Approaches to Information and Knowledge Management on the Web (I3), WWW2007 Workshop, Banff, Alberta Canada*, May 2007.

[HA06] John Horner and Michael Atwood. Effective Design Rationale: Understanding the Barriers. In Allen Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 73–90. Springer, 2006.

[HBS05] M. Hepp, D. Bachlechner, and K. Siorpaes. OntoWiki: Community-driven Ontology Engineering and Ontology Usage based on Wikis, 2005.

[Inc] JotSpot Inc. Jotspot beta: the application wiki. http://www.jotspot.com/.

[JMLSSJ] Jugel Matthias L. and Schmidt Stephan J. Snipsnap: the easy weblog and wiki software. http://www.snipsnap.org/space/.

[KEE] Kim E. E. Purplewiki. http://purplewiki.blueoxen.net/cgi-bin/wiki.pl.

[KGLCA07] Alexander Kubias, Laurian Gridinoc, Angel Lopez-Cima, and Carlos Buil Aranda. The role of access rights in ontology customization. Deliverable D4.4.1, NeOn Project, 2007.

[KKIM02] K. Kozaki, Y. Kitamura, M. Ikeda, and R. Mizoguchi. Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of" Role" and" Relationship. *Proc. of EKAW2002*, pages 213–218, 2002.

[KKPS02] J. Kahan, M.R. Koivunen, E. Prud'Hommeaux, and RR Swick. Annotea: an open RDF infrastructure for shared Web annotations. *Computer Networks*, 39(5):589–608, 2002.

[KN03] M. Klein and N. Noy. A component-based framework for ontology evolution. In *Proceedings of the Workshop on Ontologies and Distributed Systems*, 2003.

[Kol96] P. Kollock. . In *Proceedings of the Harvard Conference on Internet and Society*, 1996.

[KPS+06] A. Kalyanpur, B. Parsia, E. Sirin, B.C. Grau, and J. Hendler. Swoop: A Web Ontology Editing Browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):144–153, 2006.

[KR70] Werner Kunz and Horst Rittel. Issues as elements of information systems. Working Paper 131, Institute of Urban and Regional Development, University of California, Berkeley, California, 1970.

[KSKM07] K. Kozaki, E. Sunagawa, Y. Kitamura, and R. Mizoguchi. Distributed Construction of Ontologies Using Hozo. In *Proc. Workshop on Social and Collaborative Construction of Structured Knowledge colocated with WWW2007 in Banff*, 2007.

[Lee90] Jintae Lee. SIBYL: A Qualitative Decision Management System. In Patrick Winston and Sarah Shellard, editors, *Artificial Intelligence at MIT – Vol. 1: Expanding Frontiers*. MIT Press, 1990.

[Lee91] Jintae Lee. Extending the potts and bruns model for recording design rationale. In *ICSE '91: Proceedings of the 13th international conference on Software engineering*, pages 114–125, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[Lu03] Y. Lu. Roadmap for tool support for collaborative ontology engineering. Master thesis, XiAn Transportation University, Department of Computer Science, 1994 (2003). `http://www.cs.uvic.ca/ chisel/thesis/YilingLu.pdf`.

[McC79] Raymond McCall. *On the structure and use of issue systems in design*. PhD thesis, University of California, Berkeley, 1979.

[McC91] Raymond McCall. PHI: A conceptual foundation for design hypermedia. *Design Studies*, 12(1):30–41, 1991.

[med] *The MediaWiki Website*. http://www.mediawiki.org/wiki/MediaWiki. Viewed on November 30th, 2006.

[MIN] MINDSWAP Research Group, Univesity of Maryland. "SWOOP: A Hypermedia-based Featherweight OWL Ontology Editor". http://www.mindswap.org/2004/SWOOP/. Viewed on November 29th, 2006.

[MS06] C. Mancini and S.B. Shum. Modelling discourse in contested domains: A semiotic and cognitive framework. technical report kmi-06-14. Technical report, Open University, 2006. Final version submitted to International Journal of Human-Computer Studies.

[MT87] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. In Livia Polanyi, editor, *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, N.J., 1987.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. Owl Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2004. W3C Recommendation, W3C, February 2004.

[MYBM91] Allan MacLean, Richard Young, Victoria Bellotti, and Tom Moran. Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6:201–250, 1991.

[NCA07] N.F. Noy, A. Chugh, and H. Alani. The CKC Challenge: Exploring Tools for Collaborative Knowledge Construction. 2007.

[NCLM06] N.F. Noy, A. Chugh, W. Liu, and M.A. Musen. A Framework for Ontology Evolution in Collaborative Environments. In *Proceedings of The Semantic Web - ISWC 2006*, volume 4273, pages 544–558. Springer-LNCS, 2006.

[PB88] Colin Potts and Glenn Bruns. Recording the reasons for design decisions. In *ICSE*, pages 418–427, 1988.

[PS05] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF, 2005.

[PST04] Helena Sofia Pinto, Steffen Staab, and Christoph Tempich. DILIGENT: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In Ramon López de Mántaras and Lorenza Saitta, editors, *ECAI*, pages 393–397. IOS Press, 2004.

[PWHd07] Raul Palma, Yimin Wang, Peter Haase, and Mathieu d'Aquin. Propagation models and strategies. Deliverable D1.3.1, NeOn Project, 2007.

[QH08] Guilin Qui and Peter Haase. Improved neon formalisms for context representation. Deliverable D3.1.3, NeOn Project, 2008.

[RW73] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169, June 1973.

[Sch06] Sebastian Schaffert. Ikewiki: A semantic wiki for collaborative knowledge management. In *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA'06)*, Manchester, UK, June 2006.

[Sea06] S.B. Shum and et al. Co-opr: Design and evaluation collaborative sensemaking and planning tools for personnel recovery. Technical report kmi-06-07, Open University, UK, 2006.

[Sem] Semantic Wiki Wiki Web. http://c2.com/cgi/wiki?SemanticWikiWikiWeb.

[SLL+05] D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering thesauri for new applications: Agrovoc example. *Journal Of Digital Information*, 4, 2005.

[Sou] SourceForge. http://www.sourceforge.net/. Viewed on May 15th, 2007.

[SPKR96] B. Swartout, R. Patil, K. Knight, and T. Russ. Ontosaurus: a tool for browsing and editing ontologies. *9th Banff Knowledge Aquisition for KNowledge-based systems Workshop*, 1996.

[SSN93] T.J.M. Sanders, W.P.M. Spooren, and L.G.M. Noordman. Coherence relations in a cognitive theory of discourse representation. *Cognitive Linguistics*, 4(2):93–133, 1993.

[SSS+06] Simon Buckingham Shum, Albert Selvin, Maarten Sierhuis, Jeff Conklin, Charles Haley, and Bashar Nuseibeh. Hypermedia Support for Argumentation-Based Rationale: 15 Years on from gIBIS and QOC. In Allen Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech, editors, *Rationale Management in Software Engineering*, pages 111–132. Springer, 2006.

[Sta] Stanford Medical Informatics. *Protégé Ontology Editor*. http://protege.stanford.edu/. Viewed on November 29th, 2006.

[STV+06] Y. Sure, C. Tempich, D. Vrandecic, S. Pinto, E. Paslaru Bontas, and M. Hefke. Sekt methodology: Initial framework and evaluation of guidelines. Deliverable d7.1.2, SEKT project, 2006.

[Sun] Sun Microsystems. Javadoc Tool. http://java.sun.com/j2se/javadoc/. Viewed on May 11th, 2007.

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

[TMN06] G. Tummarello, C. Morbidoni, and M. Nucci. Enabling Semantic Web communities with DBin: an overview. *International Semantic Web Conference*, pages 943–950, 2006.

[TN07] Tania Tudorache and Natasha Noy. Collaborative Protégé. In *Proc. Workshop on Social and Collaborative Construction of Structured Knowledge colocated with WWW2007 in Banff*, 2007.

[Vea06] D. Vrandecic and et al. Sekt methodology: Initial lessons learned and tool design. Deliverable d7.2.1 of the sekt project, Institut AIFB, Universitaet Karlsruhe (TH), Germany, 2006.

[VO06] Max Völkel and Eyal Oren. Towards a Wiki Interchange Format (wif). In Max Völkel and Sebastian Schaffert, editors, *Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics*, 2006.

[W3C] W3C. Resource Description Framework (RDF): Concepts and abstract syntax. http://www.w3.org/TR/rfd-concepts/. Recommendation 10 February 2004.

[Web] Wiki Wiki Web. Cunningham and Cunningham Inc. http://c2.com.

[Wik] WikiMedia. “Semantic MediaWiki". http://meta.wikimedia.org/wiki/SemanticMediaWiki. Viewed on November 30th, 2006.