NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D2.3.2 Practical Methods to Support Collaborative Ontology Design (V2)

Deliverable Co-ordinator: **Wim Peters, Aldo Gangemi**

Deliverable Co-ordinating Institution: **USFD/CNR**

Other Authors: Valentina Presutti (CNR); Dunja Mladenic (JSI); Raul Palma (UPM); Klaas Dellschaft (UKO-LD); Alessandro Adamou (CNR), Enrico Daga (CNR), Holger Lewen (UKARL), Michael Erdmann, Anne Becker (ONTO)

Document Identifier:	NEON/2009/D2.3.2/v1.1	Date due:	January 30, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	September 15, 2009
Project start date:	March 1, 2006	Version:	v1.1
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

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

Open University (OU) – Coordinator Knowledge Media Institute – Kmi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: glesi.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: glesi@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: gle@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 – 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 1 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

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

CNR

JSI

USFD

UKO-LD

UPM

ONTO

UKARL

Change Log

Version	Date	Amended by	Changes
0.1	06-01-2009	Wim Peters	TOC
0.2	09-02-2009	Wim Peters	Added section 4.7
0.3	12-02-2009	Wim Peters	Added section 4.5 and 4.6
0.4	20-02-2009	Wim Peters	Added section 4.3
0.5	21-02-2009	Wim Peters	Added section 4.3
0.6	27-03-2009	Aldo Gangemi	Added new plugin descriptions, plugin models, and partly filled sections 1,2,3,5
0.7	01-04-2009	Wim Peters	Editorial integration
0.8	10-04-2009	Alessandro Adamou, Wim Peters	Added improved version of section 5
0.9	14-04-2009	Wim Peters, Aldo Gangemi	Added section 4.8 and figures
0.10	15-04-2009	Wim Peters, Aldo Gangemi	Changes figures
0.11	16-04-2009	Wim Peters	Minor editorial improvements
0.12	19-04-2009	Aldo Gangemi	Revision of figures, styles, table of contents; added COAT figure and N3 code; some text revision
0.13	14-09-2009	Wim Peters, Aldo Gangemi	Improvements as requested by reviewers, also reinstating changes as in 0.12 that were missing in version sent to reviewers (0.11)

Executive Summary

This deliverable describes practical methods for collaborative ontology design support in the form of a number of tools. It introduces two new tools for collaborative design that have been recently added to the NeOn palette in WP2 research development activities (COAT and SocialOnto). It also contains lightweight formal descriptions of five existing tools for collaborative design, which come from heterogeneous requirements, approaches, and NeOn work packages. These lightweight descriptions receive formal owl definitions in C-ODO Light format (see Deliverable 2.1.2 for a detailed description). Their alignment with the C-ODO Light framework enables a uniform interface for tool selection and workflow definition, implemented by the Kali-ma plug-in. This will equip the NeOn Toolkit with a design-oriented, rather than language-oriented user interface, and aims at facilitating the integration of existing NeOn Toolkit plug-ins in anticipation of their use in a collaborative context, for which Kali-ma will provide an environment, which is only sketched here (Section 5). Kali-ma architecture, implementation, and testing will be detailed in the forthcoming deliverable D2.3.3.

Table of Contents

List of tables

List of figures

1. Introduction

Navigating the sea of requirements, specifications, and capabilities of software components is a difficult task. It encompasses the know-how of good software engineers, and has been implemented by many different languages and methods. As an important disclaimer, we are not going to propose a new one.

On the other hand, much less has been developed for ontology design requirements and descriptions, and C-ODO is a set of ontologies that attempts to provide a vocabulary to talk about that. When dealing with ontology design, however, most of the activities are carried out with the help of software tools, so that the worlds of ontology design and software design have a reasonable overlap, which goes well beyond the need of good software tools in order to perform good ontology design. As the recent history of the Web of Data shows, the development of RDF datasets, their reengineering practices, the usage of OWL vocabularies to reason on them, and now also their visualization and interaction aspects are becoming prominent and they largely overlap web design.

It is no surprise then that after the first release of C-ODO in 2006 (D2.1.1, [23]), which was concentrated on describing the practices of ontology design as a mainly human-centred set of activities, we started realizing that more effort should be involved in describing the actual pieces of software that accompany those activities, and the actual data ("knowledge types") that are managed computationally.

This deliverable is based on the results of the new developments in the definition of C-ODO Light, as they result from its application to the description of several design tools developed in NeOn as plugins to the NeOn Toolkit.

2. Goal of this deliverable

This deliverable has two main objectives.

First, to introduce new tools for collaborative design that have been recently added to the NeOn palette in WP2 research development activities.

Second, to gather a lightweight formal description of existing tools for collaborative design, which come from heterogeneous requirements, approaches, and NeOn work packages. This second objective aims at a clear match between ontology design functionalities and the possible implementations of those functionalities in NeOn. A practical outcome of this second objective is constituted by the ongoing work on the Kali-ma tool, whose architecture is outlined at the end of this deliverable. This plug-in will be able to help users overcome the problem of locating available functionalities within the NeOn Toolkit framework, and provides new ways of composing implemented functionalities according to explicit design needs.

3. Collaborative Ontology Design and C-ODO Light

This deliverable has dependencies with regard to the deliverable D2.1.2 (Model for collaborative design of networked ontologies (v2)), where the C-ODO network of ontologies has been updated and reconstructed as C-ODO Light, which deals with new requirements:

1. Ability to formalize tool descriptions in terms of input/output data, functionalities, interface objects and interaction patterns;
2. Alignments to existing vocabularies such as DOAP, FOAF, OMV, etc.
3. Lighter axiomatization (e.g. no anonymous classes in restrictions)
4. Patterns-based design, which reuses the ontologydesignpatterns.org practices

C-ODO Light (Fig. 1) is briefly summarized in the following section.

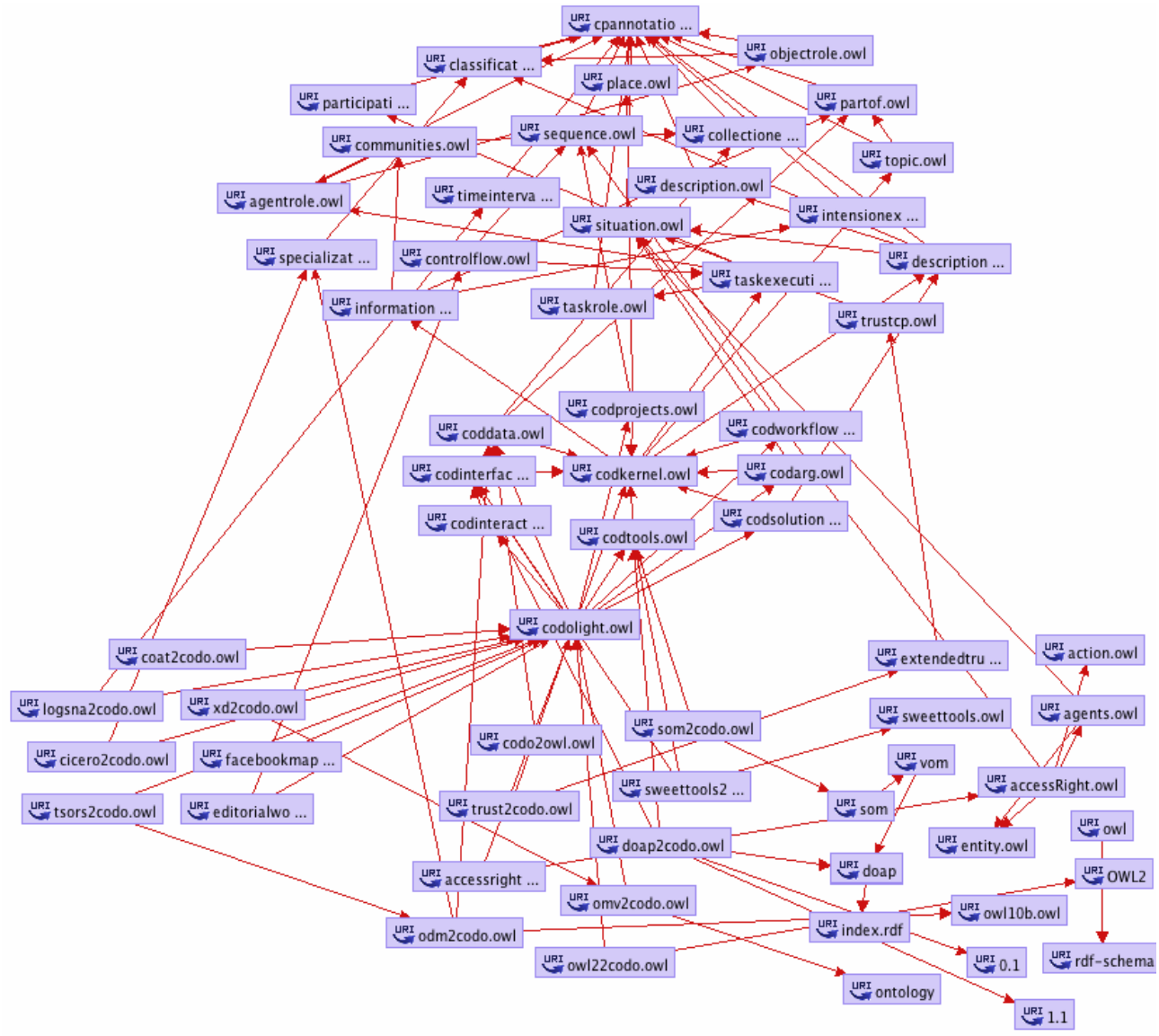

Figure 1: The C-ODO Light network

3.1 C-ODO Light

The C-ODO Light network of ontologies is currently organized according to 4-layer architecture:

1. *Pattern layer*: it contains reusable content ontology design patterns [22], such as *sequence, partof, situation, collectionentity,* etc. The patterns (Fig. 2) are reused in the design of the ontologies constituting the "corolla" architecture of C-ODO Light.

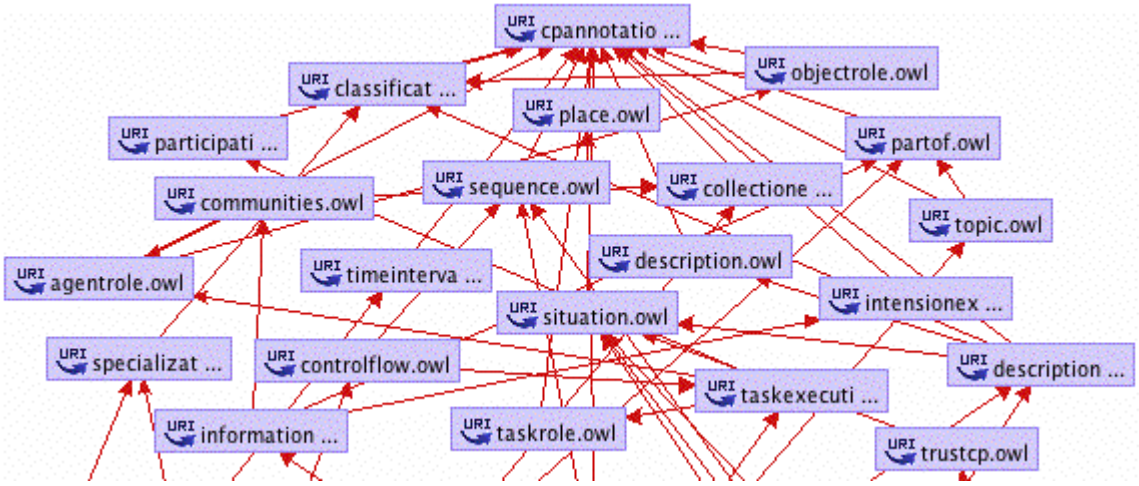

Figure 2: **The *pattern* layer in the C-ODO Light network**

2. *C-ODO layer*: it contains the nine modules of the C-ODO light core network of ontologies, organized as in a corolla, with *codkernel* module in the centre, and the modules: *coddata, codprojects, codworkflows, codarg, codsolutions, codtools, codinterfaces, and codinteraction* importing codkernel (Fig. 2).

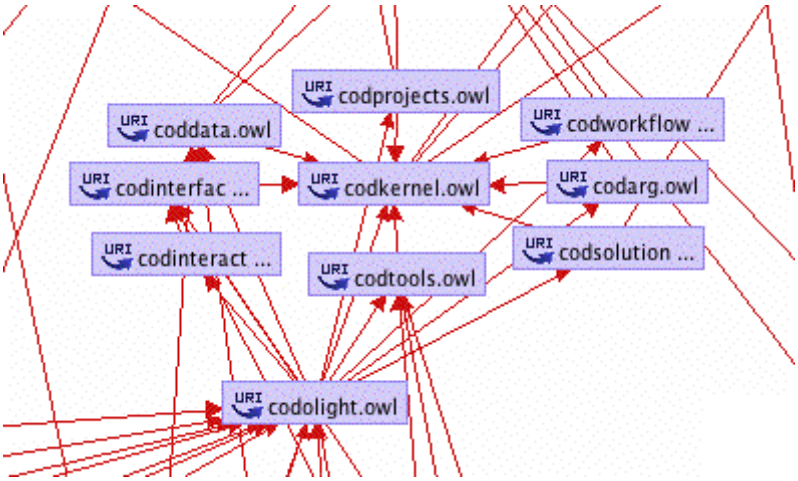

Figure 3: The *core* layer in the C-ODO Light network

3. *Alignment layer*: it consists of the modules containing mapping axioms between C-ODO Light and related vocabularies, currently: OMV, DOAP, FOAF, NeOn Trust ontology, NeOn Access Rights ontology, NeOn OWL1.0 metamodel, NeOn OWL2 metamodel, the RDF-OWL datamodel, and the SweetTools vocabulary (Fig. 3).

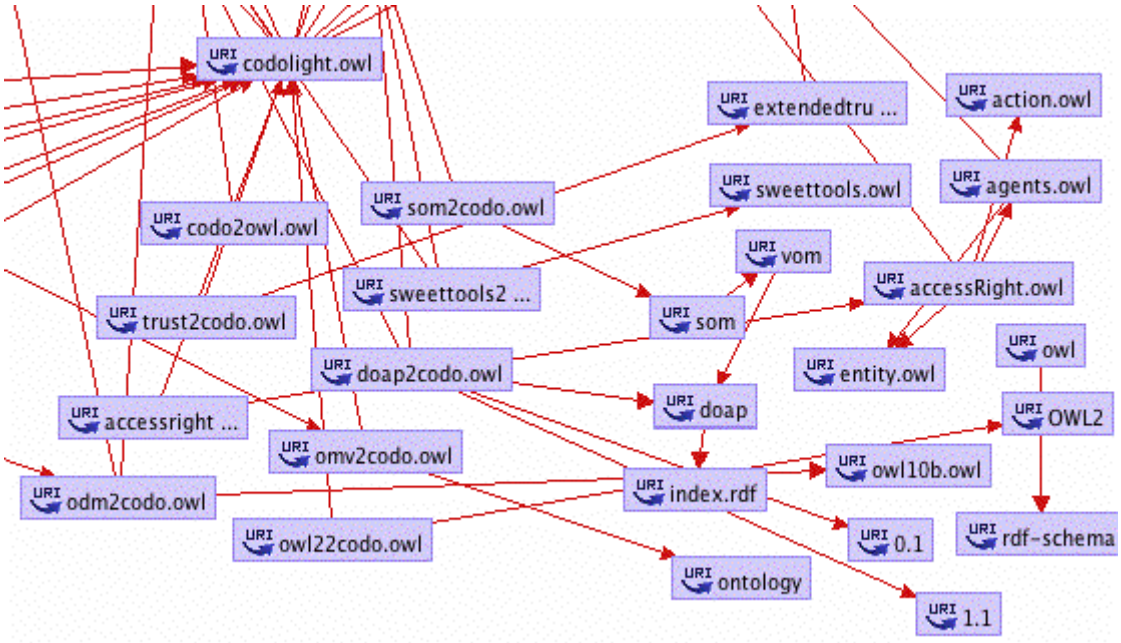

Figure 4: The *alignment* layer in the C-ODO Light network

4. *Plugin layer*: it consists of the modules containing the descriptions of the NeOn plugins related to ontology design, formalized in OWL by reusing the C-ODO vocabulary and some of the alignment modules (Fig. 4).

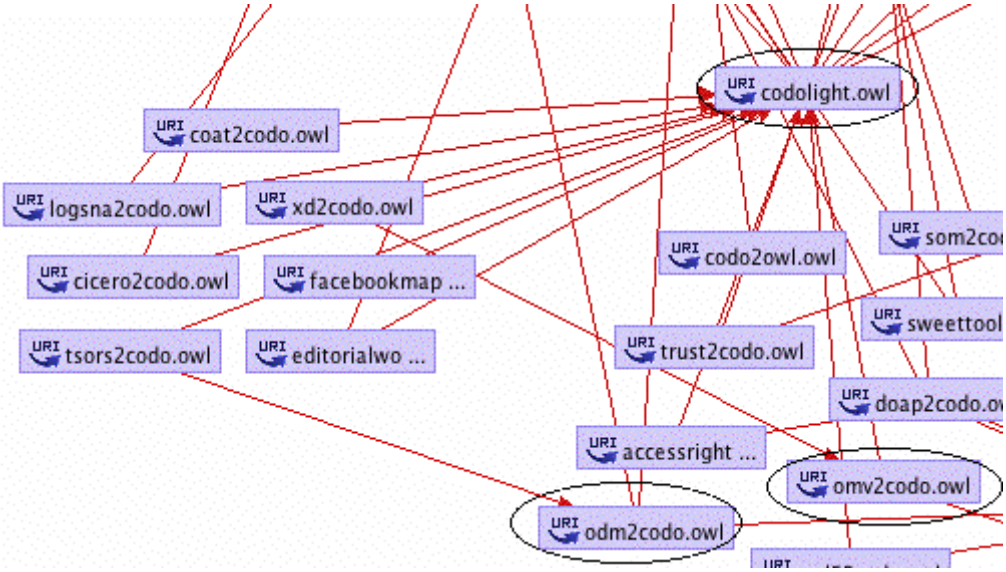

Figure 5: The *plugin* layer in the C-ODO Light network

The transitive closure of all modules in the four layers is loadable through the OWL ontology: http://www.ontologydesignpatterns.org/cpont/codo/allcodomappings.owl, which only contains `owl:import` axioms. An example of selected imports in the network is shown in Fig. 6.

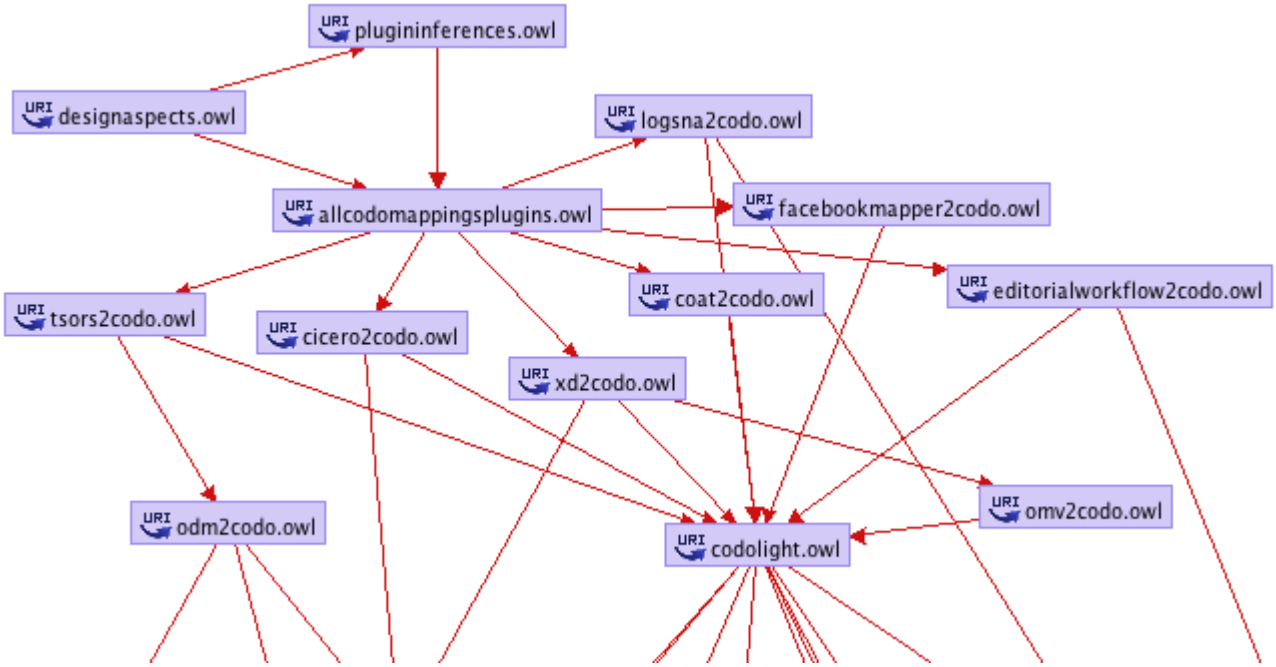

Figure 6: The C-ODO Light network

A relevant fragment of C-ODO is depicted in Fig. 7.

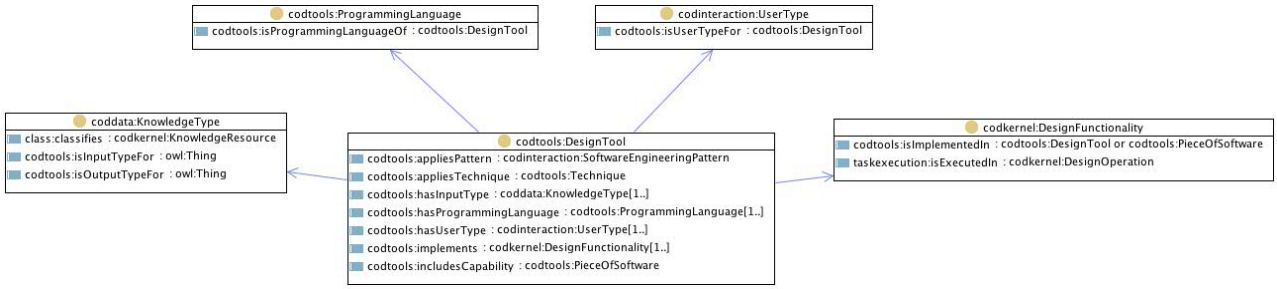

Figure 7: The vocabulary for the class *DesignTool* in C-ODO Light

4. Description of the tools

A number of tools are described in various ways in the following sections: free text annotated with C-ODO concepts, C-ODO lite UML diagrams, etc., and for each one there is an ontology that instantiates C-ODO Light concepts for their functionalities, pieces of software, knowledge types, etc.

4.1 Workflow Support (UPM)

The main purpose of the **Workflow Support** feature is to manipulate changes proposed to ontologies according to a well-defined process that coordinates who, when and how the ontology changes. In particular, the workflow feature supports (i) *the identification of users to the system*, (ii) *the management of the workflow activities* and (iii) *the user-interaction with the workflow.* For the workflow management, the component (i) takes care of enforcing the constraints imposed by the collaborative workflow (described in D1.3.1), (ii) creates the appropriate action individuals of the workflow ontology (described in D1.3.2) and (iii) registers them into the distributed registry Oyster [24]. Hence, whenever a new workflow action is performed, the component performs the following tasks:

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

To support the user's activities we rely on the ontology editor of NeOn Toolkit for the edition of ontology elements. Additionally, the NeOn Toolkit is extended with a set of views that allow editors to (i) see the appropriate information of ontologies in the editorial workflow and (ii) perform the applicable workflow actions (approve, reject, etc.), depending on their role. There are four views:

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

4.1.1 The Functionality of workflow support in CODO Light terms

This section describes the functionality of the workflow support implemented as a NeOn Toolkit Feature. The elements from the CODO Light ontology used in this description are in italics.

The workflow feature implements a *CollaborativeOntologyDevelopmentFunctionality* and it is an instance of a *CollaborativeOntologyDevelopmentTool*. In a nutshell, it supports ontology editors with different roles (instances of *UserType*) to propose changes, which are validated or rejected later by an authorized user. The workflow feature takes care of enforcing the constraints imposed by a *CollaborativeWorkflow* which coordinates who, when and how an ontology changes. Hence, the workflow defines the *actions* that *ontology editors* can perform according to their *role* and depending on the *state* of the ontology elements (for additional information we refer the reader to D1.3.1 and D1.3.2). Furthermore, users are also able to visualize depending on their role, the proposed changes.

The functionalities supported are (*CollaborativeOntologyDevelopmentFunctionality*):

- ***User Identification***: Identifies users to the system and log them in. The user information includes user first name and last name and user role.
- ***Workflow Management*****:** Based on the predefined collaborative workflow, it performs several *tasks* (*CollaborativeWorkflowTask*): enforces the workflow constraints, create appropriate workflow instances (of *workflow ontology*) and register them into the *ontology registry*.
- ***User Interaction in the workflow*****:** Support the following *tasks* (*CollaborativeWorkflowTask*): visualization of the appropriate information of ontologies in the editorial workflow and (ii) execution of the applicable workflow actions (approve, reject, etc.), depending on their role.

Figure 8 below illustrates the CODO-based conceptual design of the workflow support.

Figures 9-11 illustrate the editorialworkflow2codo.owl model (see Appendix for n3 format), which describes the basics of the tool in terms of C-ODO Light vocabularies. The figures represent views of editorialworkflow2codo.owl for functionalities included in workflows (Figure 10), precedence relations between functionalities (tasks, Figure 11), and input/output knowledge types for a functionality (Figure 9).
Fig. 9 illustrates the basic axioms and the input and output types for the workflow support tasks. Fig. 10 illustrates basic axioms about functionalities such as ChangeLogging, WorkflowVisualization and WorkflowManagement.
Fig. 11 presents a view on the basic axioms combined with precedence relations between tasks that capture the workflow restrictions of the tool.

Boxes represent classes, aligned to codarg.owl vocabulary, while triangles represent individuals (design tools, workflows, functionalities, knowledge types, user types).

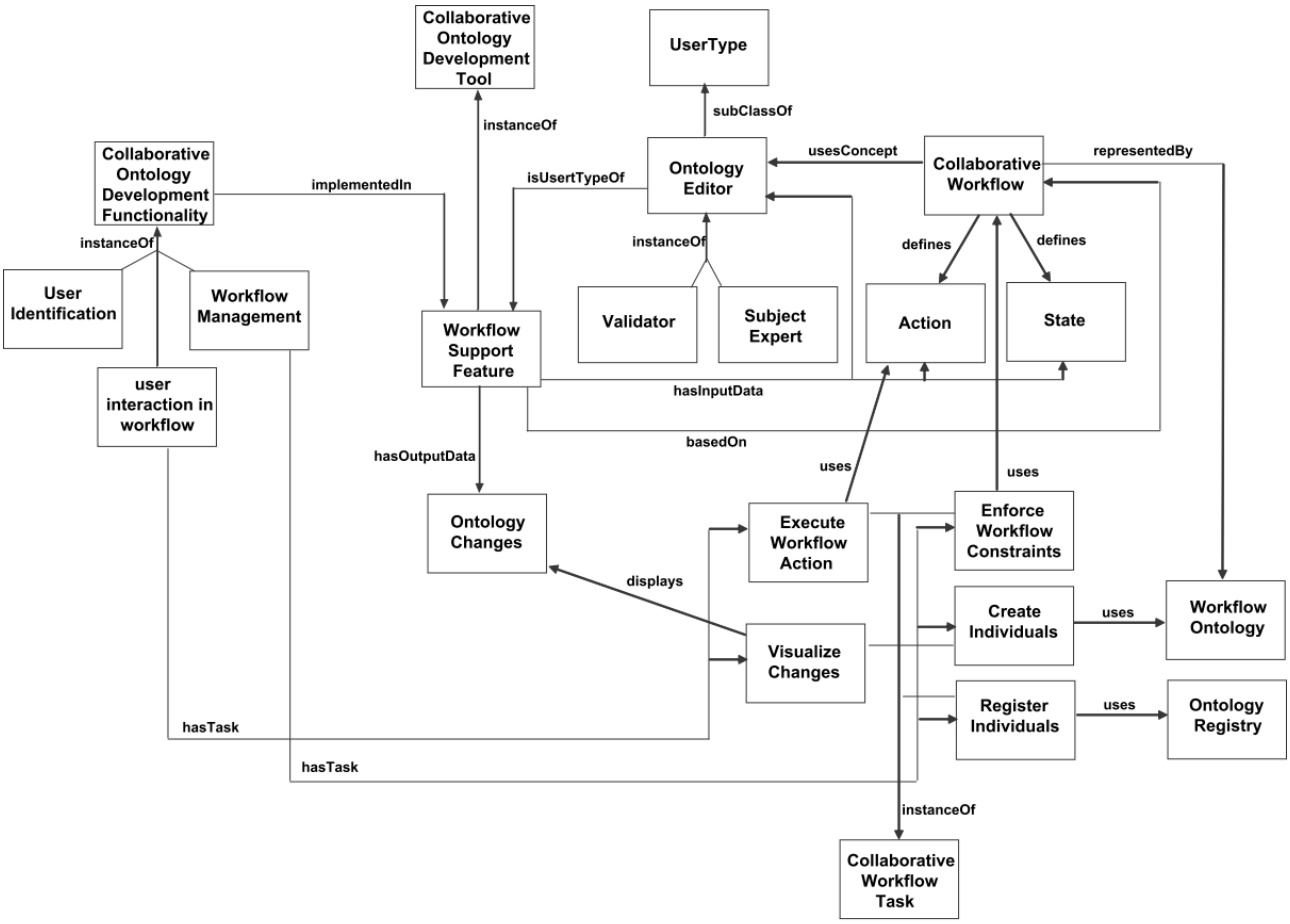

Figure 8: CODO Light model of Workflow Support

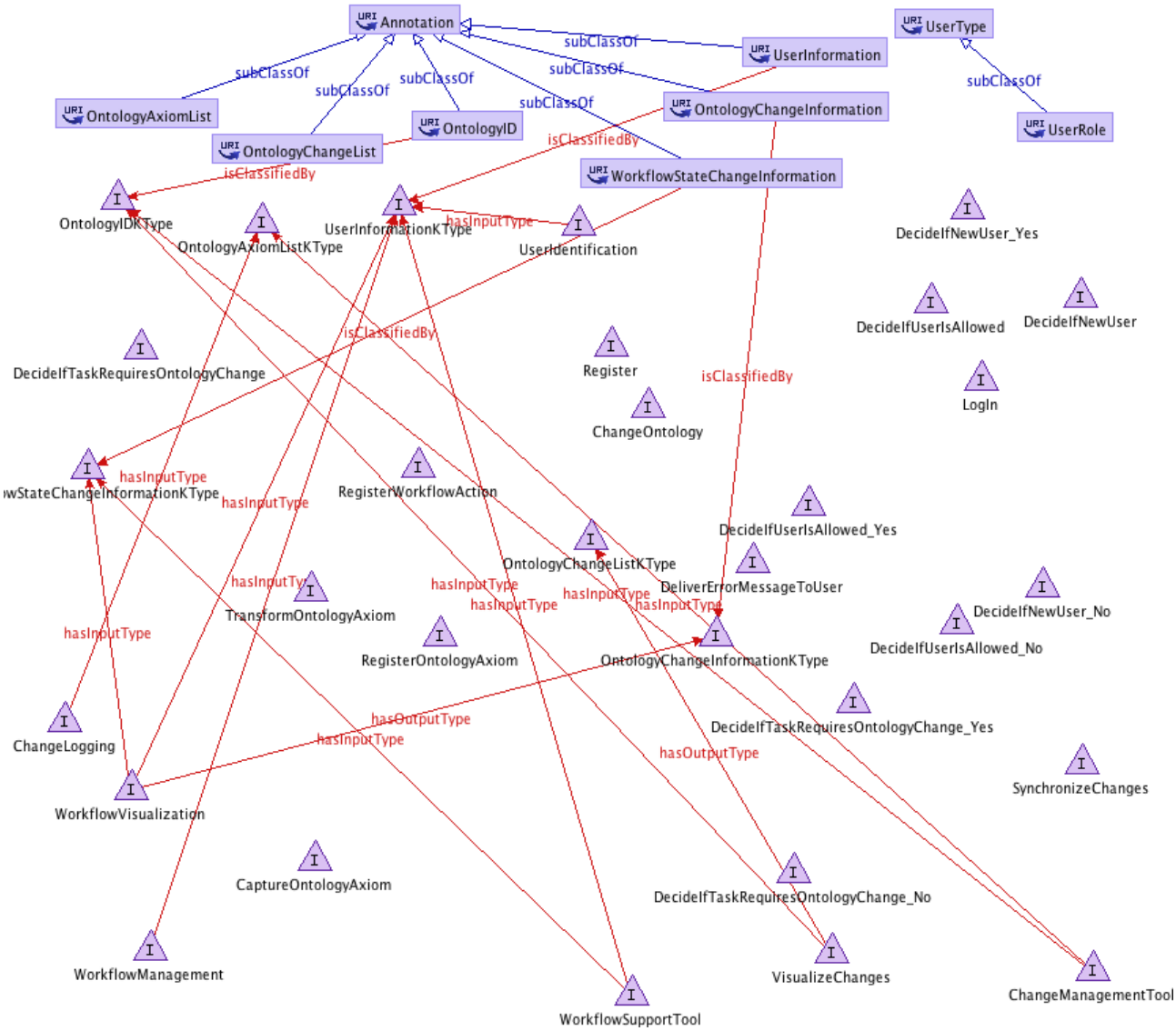

Figure 9: Workflow Support in C-ODO Light: basic axioms + input/output knowledge types wrt tasks declared for the tool

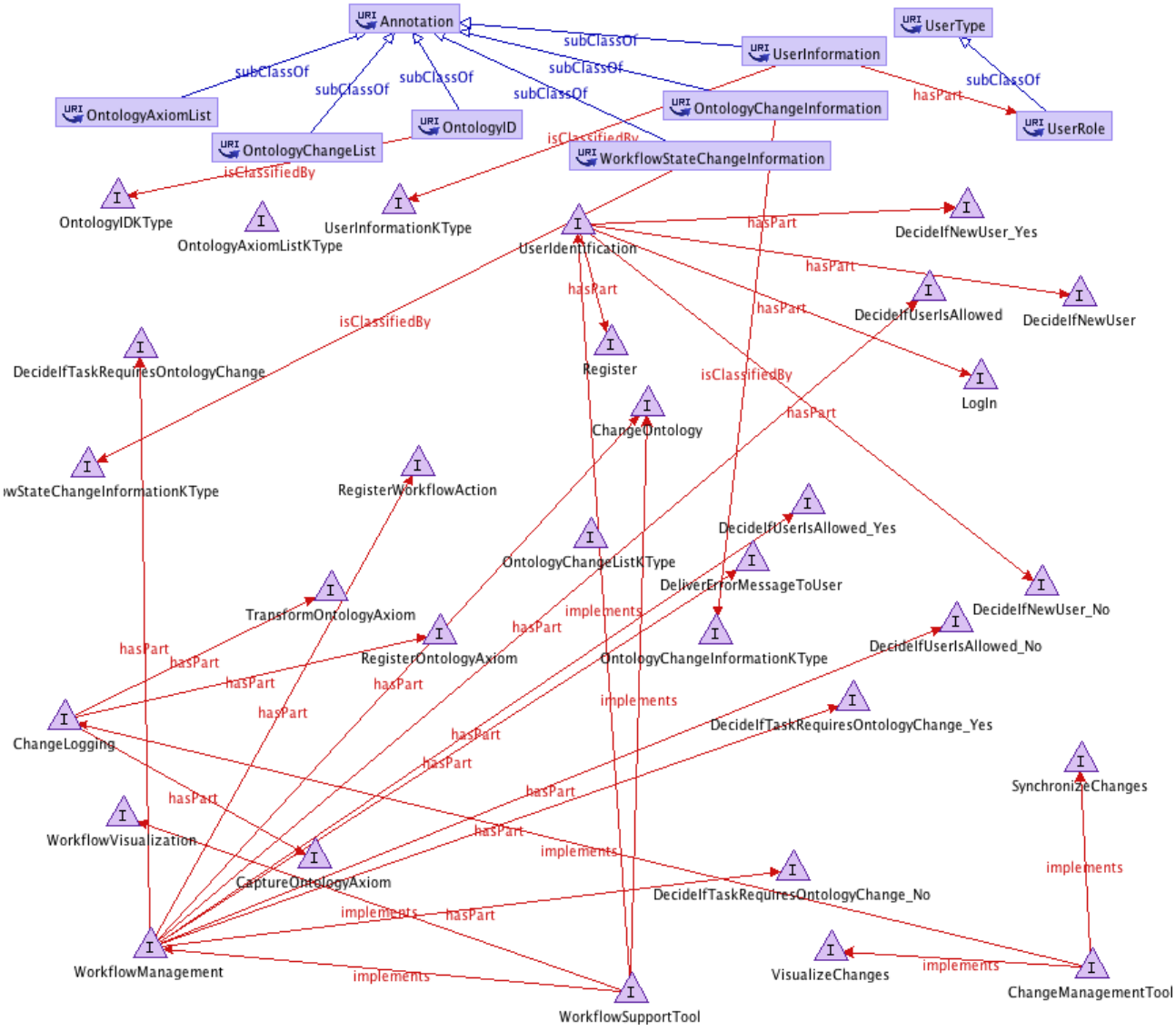

Figure 10: Workflow Support in C-ODO Light: basic axioms + implemented tasks declared for the tool

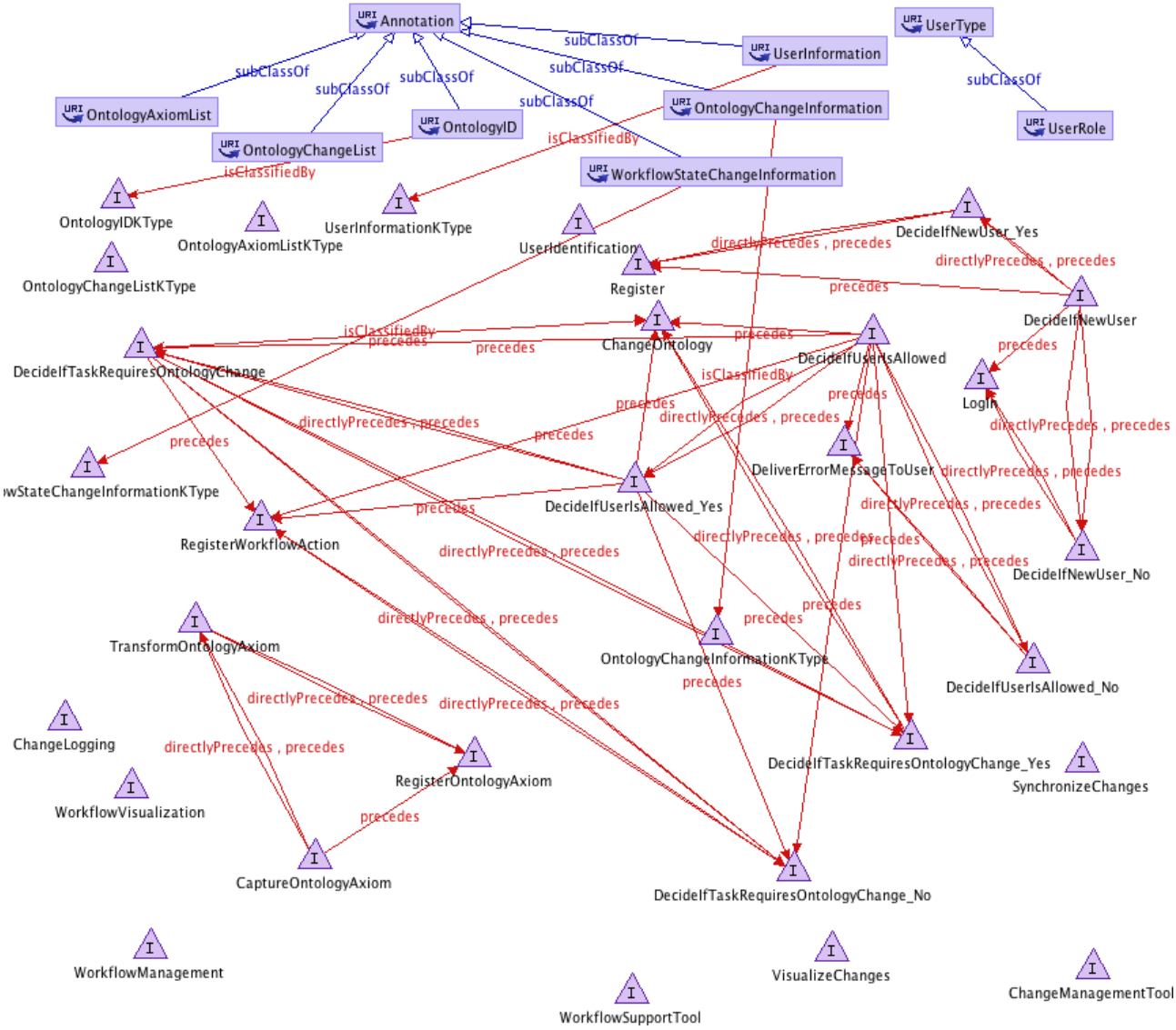

Figure 11: Workflow Support in C-ODO Light: basic axioms + precedence relations between tasks

4.2 Integration of Open Rating System, Oyster, Alignment Server and Watson (UKARL)

The Open Rating System can be used to collaboratively review content, in the case of NeOn, ontologies. Being able to break the reviews down into to bits on certain properties of the ontology (e.g. Domain Coverage, Usability, Reusability) enables reviewers to only review the aspects they are most comfortable reviewing. Ideally, the reviews are assigned based on the expertise of the reviewer, similar to the peer review process currently used in science. But, the reviews do not have to be coordinated, just by reviewers entering reviews, they help making the system better and provide added value to the users. It does not matter if there are multiple reviews for the same property of an ontology, since reviews are ranked based on the trust in a reviewer and this is user-specific. So on a global scale, all reviewers collaborate to provide a more detailed evaluation of the ontologies and to provide added value to the users.

Users can then choose which reviewer(s) to trust and get their reviews and ontologies ranked based on an underlying trust network. For details on the algorithms see D2.2.1.

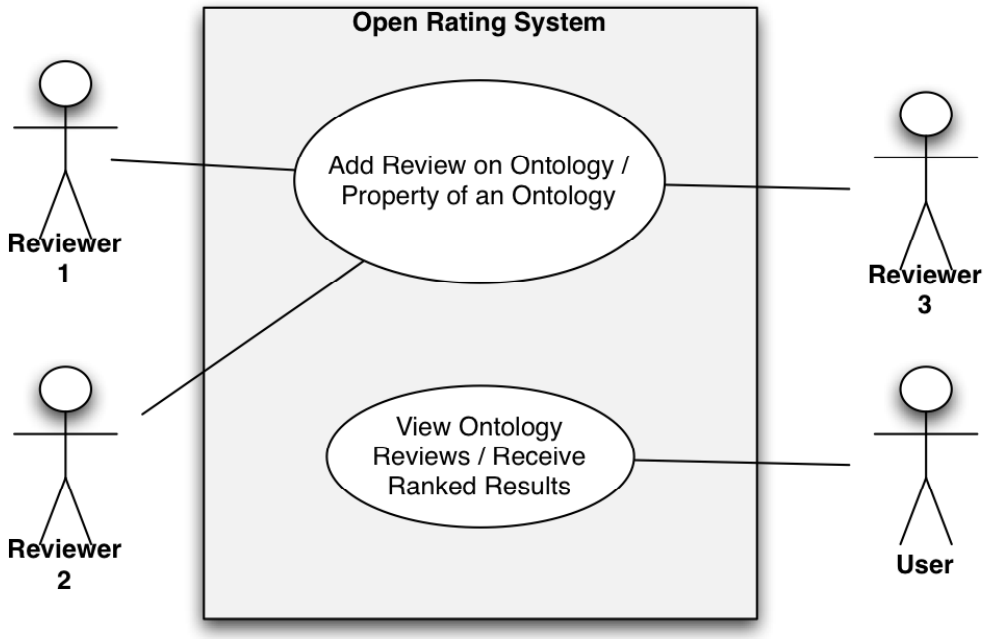

Figure 12: Schematic overview of Open Rating System

4.2.1 Workflow support TS-ORS Light model

Figure 13 below illustrates the functionalities covered by the Collaboration Server along the same lines as in the previous section: basic axioms, input/output types, implementation and precedence relations.

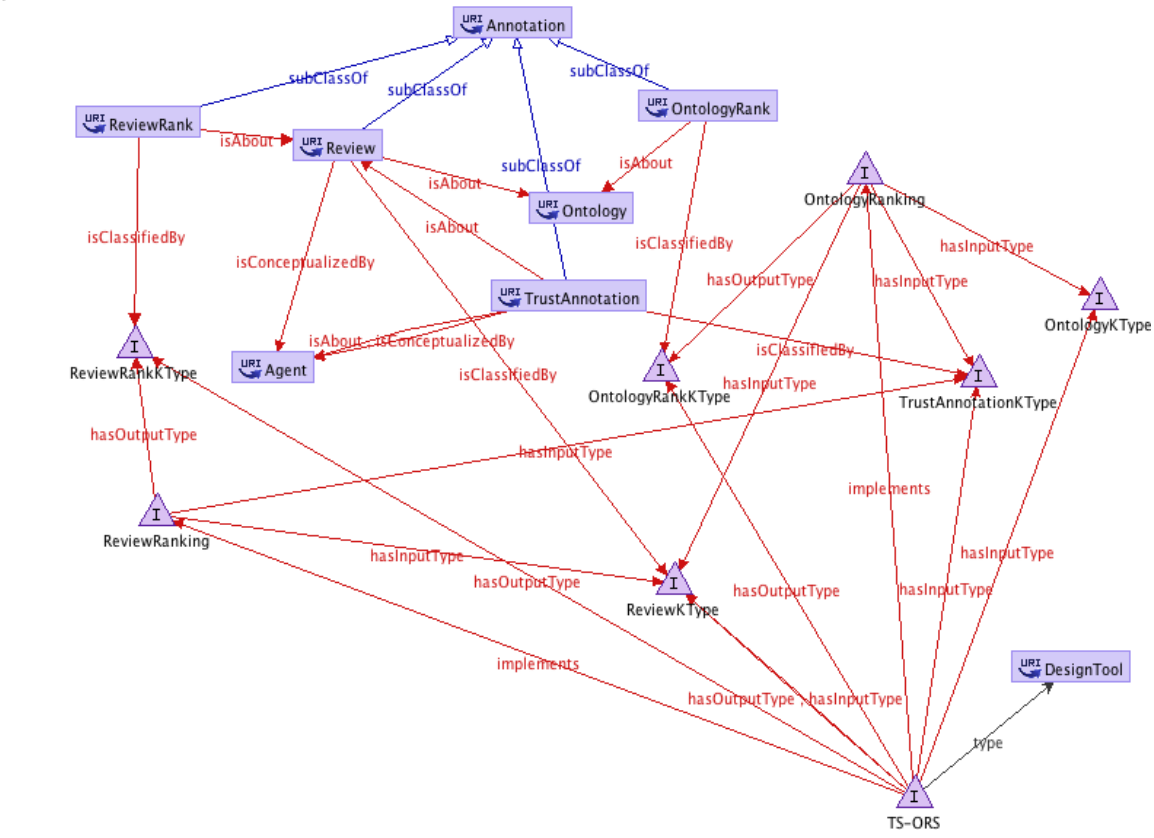

Figure 13: ORS in C-ODO Light

4.3 Argumentation Support on a Semantic Wiki (Cicero) (UKO)

Creating and designing an ontology is a complex task that requires the collaboration of domain and ontology engineering experts. 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.

The Cicero tool facilitates an asynchronous discussion and decision taking process between participants of an ontology engineering project. Two main objectives of capturing discussions in Cicero can be distinguished:

- *Higher Efficiency*: Cicero supports its users in discussing the design rationale of ontologies. The whole discussion including the pro and contra arguments is recorded, leading to fewer redundancies in disputes. It has been shown that the applied discussion methodology facilitates efficient discussions and accelerates convergence to a solution.
- *Enhanced Documentation*: The captured discussions reflect the design rationale of an ontology. By attaching a discussion to the entities in the ontology, it is possible later to understand why certain elements are modelled as they are. Furthermore, prior discussions can easily be resumed if e.g. new requirements have to be taken into account.

The first objective is accomplished by the Cicero tool itself.[1] Its underlying argumentation model and discussion workflow are described in two of the attached diagrams. The second objective is accomplished by means of the Cicero plug-in that integrates functionality of Cicero into the NeOn toolkit.

4.3.1 Cicero Wiki

In Cicero, there exist four different predefined roles (Cicero Administrator, Project Moderator, Issue Moderator and Project Member; see **Error! Reference source not found.**0). The different access rights of these roles can be configured for each project. **Error! Reference source not found.**0 shows the default access rights of each role and which actions can be triggered.

[1] It can be downloaded from http://isweb.uni-koblenz.de/Research/Cicero/.

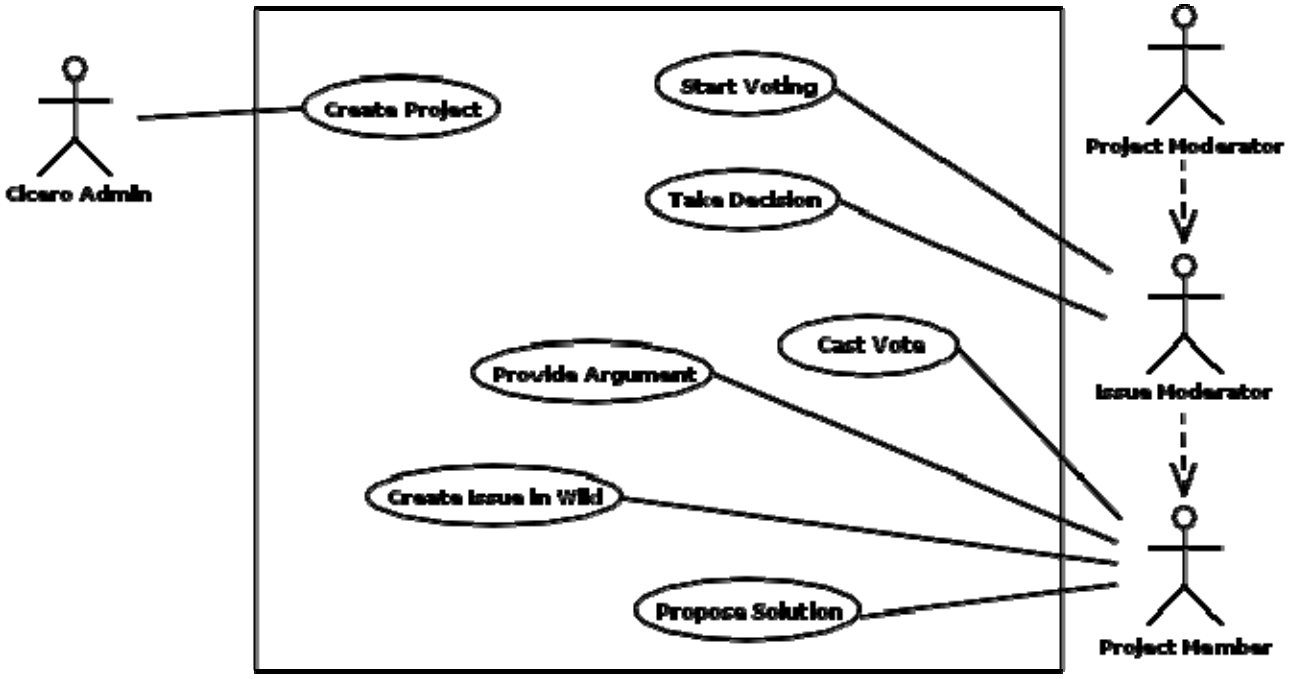

Figure 14: Use case diagram of the Cicero Wiki

The Cicero Wiki supports the following functionality:

1. *Create Project*: A Cicero Administrator creates a new project within the Wiki. The title and a short description have to be given as input. A project groups all discussions related to an ontology project.
2. *Create Issue*: A Project Member creates a new issue within an existing discussion project. The title and a short description have to be given as input.
3. *Propose Solution*: A Project Member proposes a solution for a previously created issue. A description of the solution proposal is required as input.
4. *Provide Argument*: A Project Member provides arguments in favour or against a specific solution proposal. A description of the argument is required as input.
5. *Start Voting*: An Issue Moderator starts a preferential voting for an argumentation thread (consisting of the issue, solution proposals and arguments). Having a preferential voting about the solutions is optional and depends on the participation policies in the ontology project.
6. *Cast Vote*: A Project Member casts his vote for one of the previously proposed solutions. It is required that a preferential voting is started first.
7. *Take Decision*: An Issue Moderator decides which of the previously proposed solutions should be implemented in the ontology project. This decision is based on the discussions among the project members but needn't take into account the result of the (optional) voting phase.

A more detailed description of the Cicero Wiki and how to use its functionality is available in its online manual: http://cicero.uni-koblenz.de/wiki/index.php/Help:Contents

4.3.2 Cicero Plug-in for NeOn Toolkit

The Cicero Plug-in for the NeOn Toolkit can be used for annotating ontology elements with related issues in the Cicero Wiki.[2] For using the plug-in, a login for the Cicero Wiki is required that has at least the access rights of a Project Member for the discussion project from which issues should be annotated. 11 shows which actions can be triggered with the Cicero Plug-in.

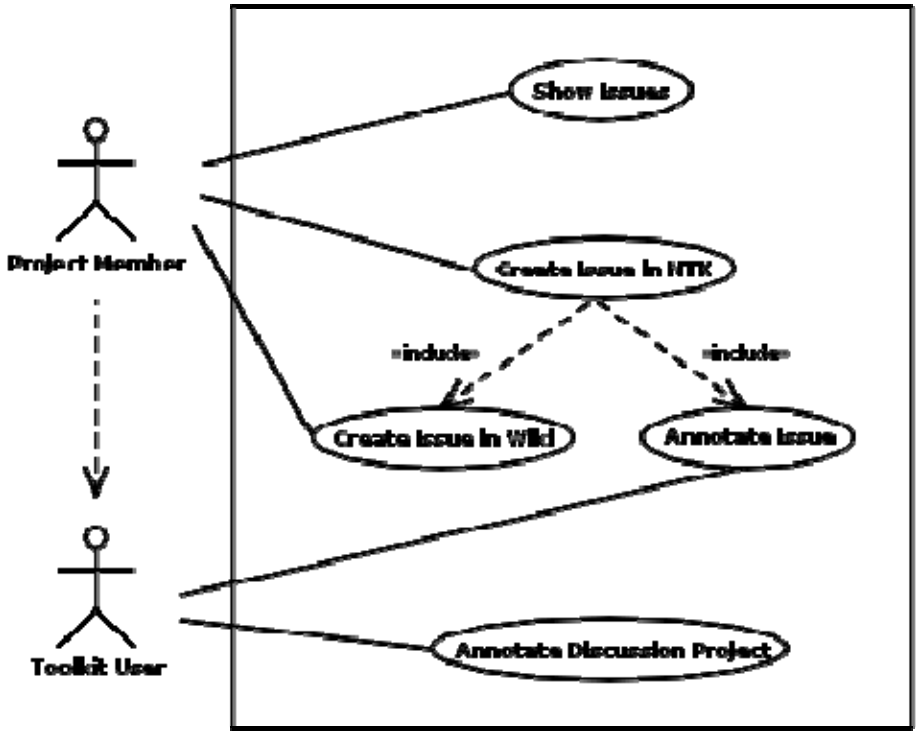

Figure 15: Use case diagram for the Cicero Plug-in

All in all, the Cicero Plug-in supports the following functionality:

1. *Annotate Discussion Project*: A user of the toolkit can annotate the URL of a discussion project to the currently opened ontology. In this discussion project, all issues will be created.
2. *Create Issue in Toolkit*: A toolkit user, who is at least a Project Member of the annotated discussion project, can create a new issue from within the NeOn Toolkit. The new issue is created in the wiki and automatically annotated to the currently selected ontology elements.
3. *Annotate Issue*: A toolkit user, who is at least a Project Member of the annotated discussion project, can annotate ontology elements with the URL of an already existing issue.
4. *Show Issues*: A toolkit user, who is at least a Project Member of the annotated discussion project, can select one or more ontology elements and retrieve the related issues from the Cicero Wiki.

A more detailed description of the Cicero Plug-in and how to use its functionality is available in the online help of the plug-in and in the wiki of the NeOn Toolkit: http://www.neon-toolkit.org/wiki/index.php/Cicero

[2] The Cicero Plugin can be downloaded from http://cicero.uni-koblenz.de/wiki/index.php/Download_(plugin)

4.3.3 Cicero C-ODO Light model

cicero2codo.owl (see Appendix) describes two complementary tools: CiceroWiki and CiceroToolkitPlugin, in terms of codolight vocabularies.
The model is visualised in the figures below, which represent views of cicero2codo.owl for functionalities included in workflows, precedence relations between functionalities (tasks), and input/output knowledge types for a functionality.

Boxes represent classes, aligned to codarg.owl vocabulary, while triangles represent individuals (design tools, workflows, functionalities, knowledge types, user types).
For example, fig. 16 illustrates the basic axioms and the input and output types for Cicero tasks. Fig. 17 shows how the individual CiceroWiki (a DesignTool) implements several functionalities: ProposeSolution, CreateIssue, CastVote, etc. A functionality can be the task for a user type, e.g. CiceroAdministrator.
Fig. 18 presents a view on the basic axioms combined with precedence relations between tasks that capture the workflow restrictions of the tool.

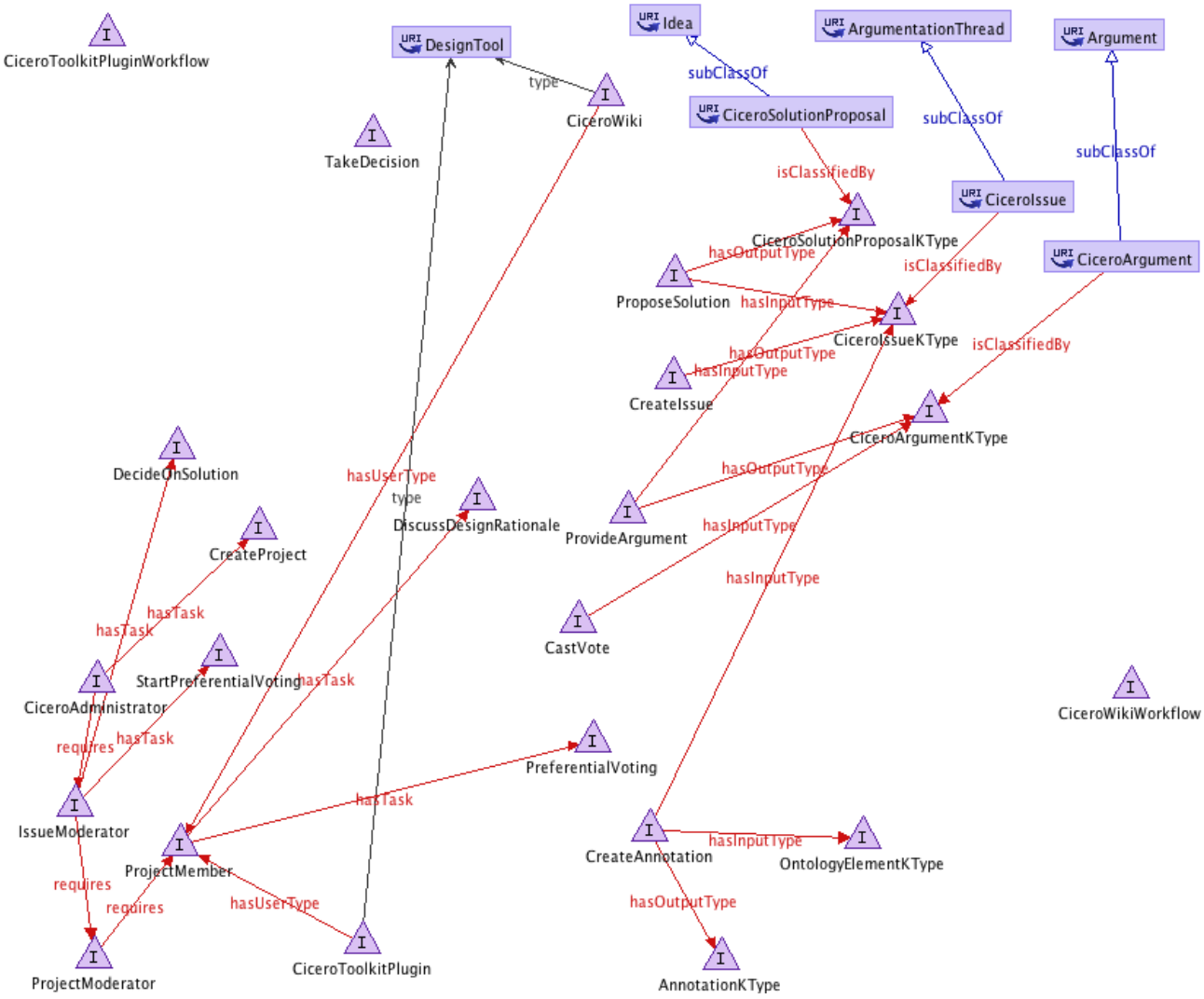

Figure 16: Cicero in C-ODO Light: basic axioms + input/output knowledge types wrt tasks declared for the tool

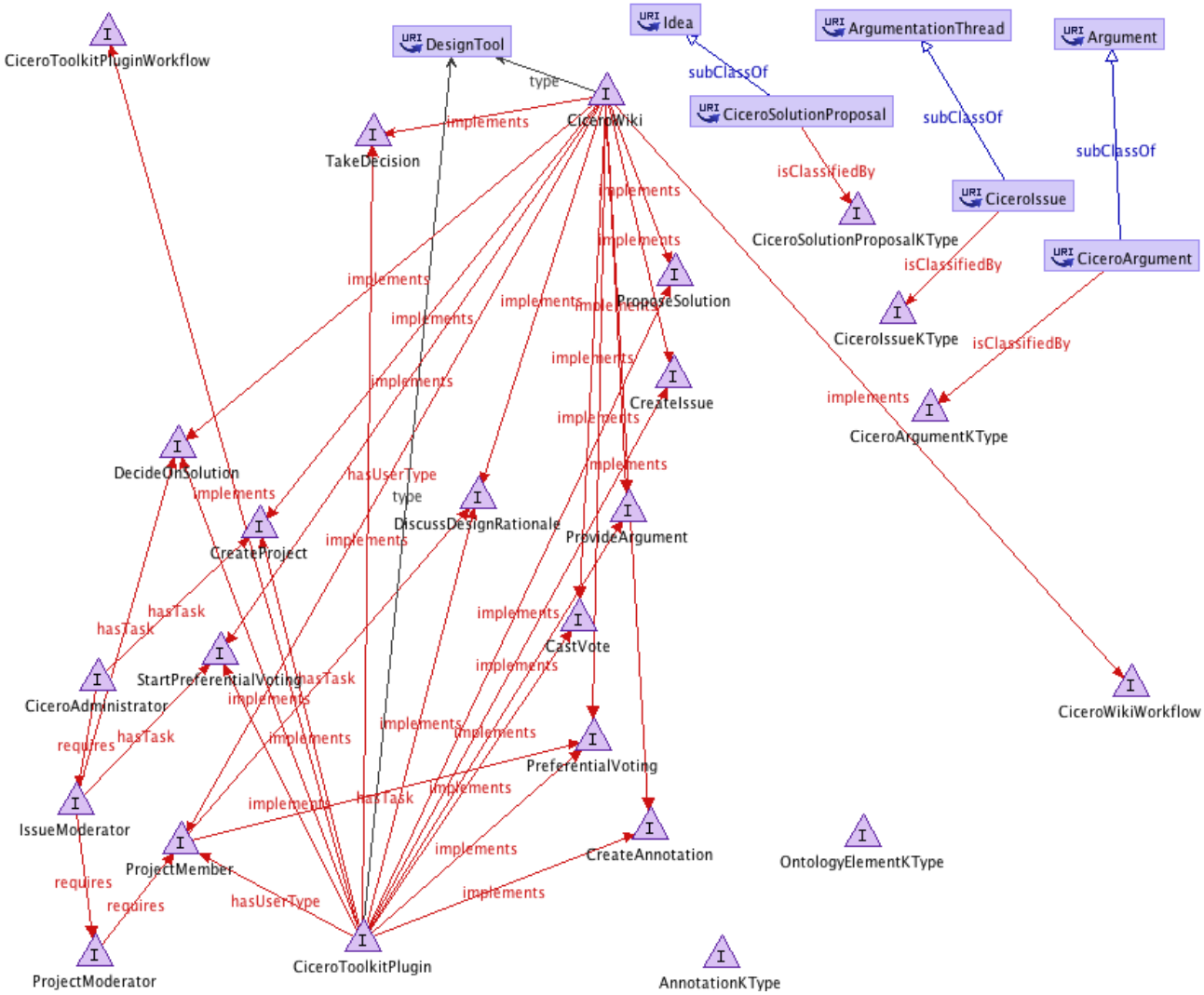

Figure 17: Cicero in C-ODO Light: basic axioms + implemented tasks declared for the tool

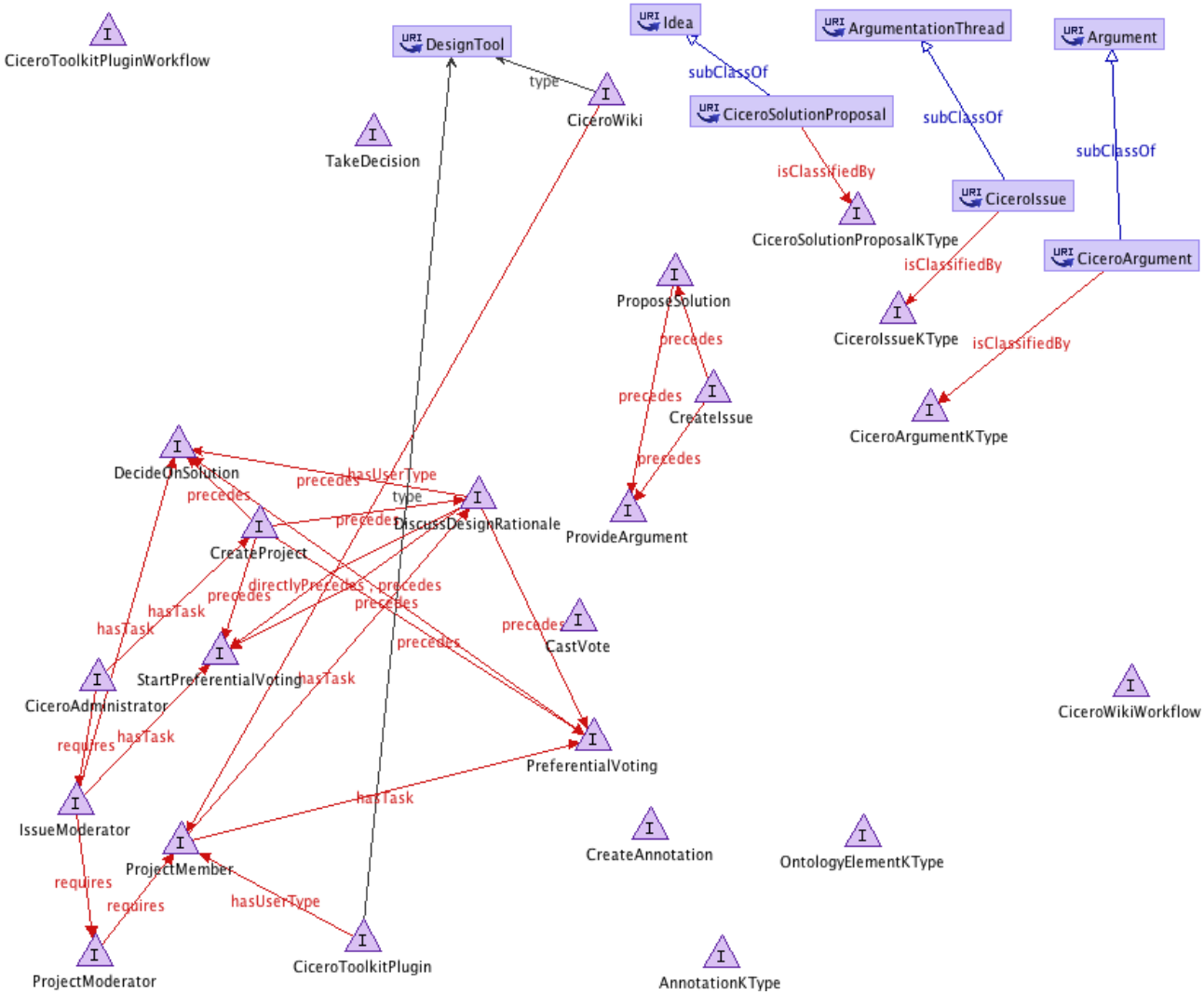

Figure 18: **Cicero in C-ODO Light: basic axioms + precedence relations between tasks**

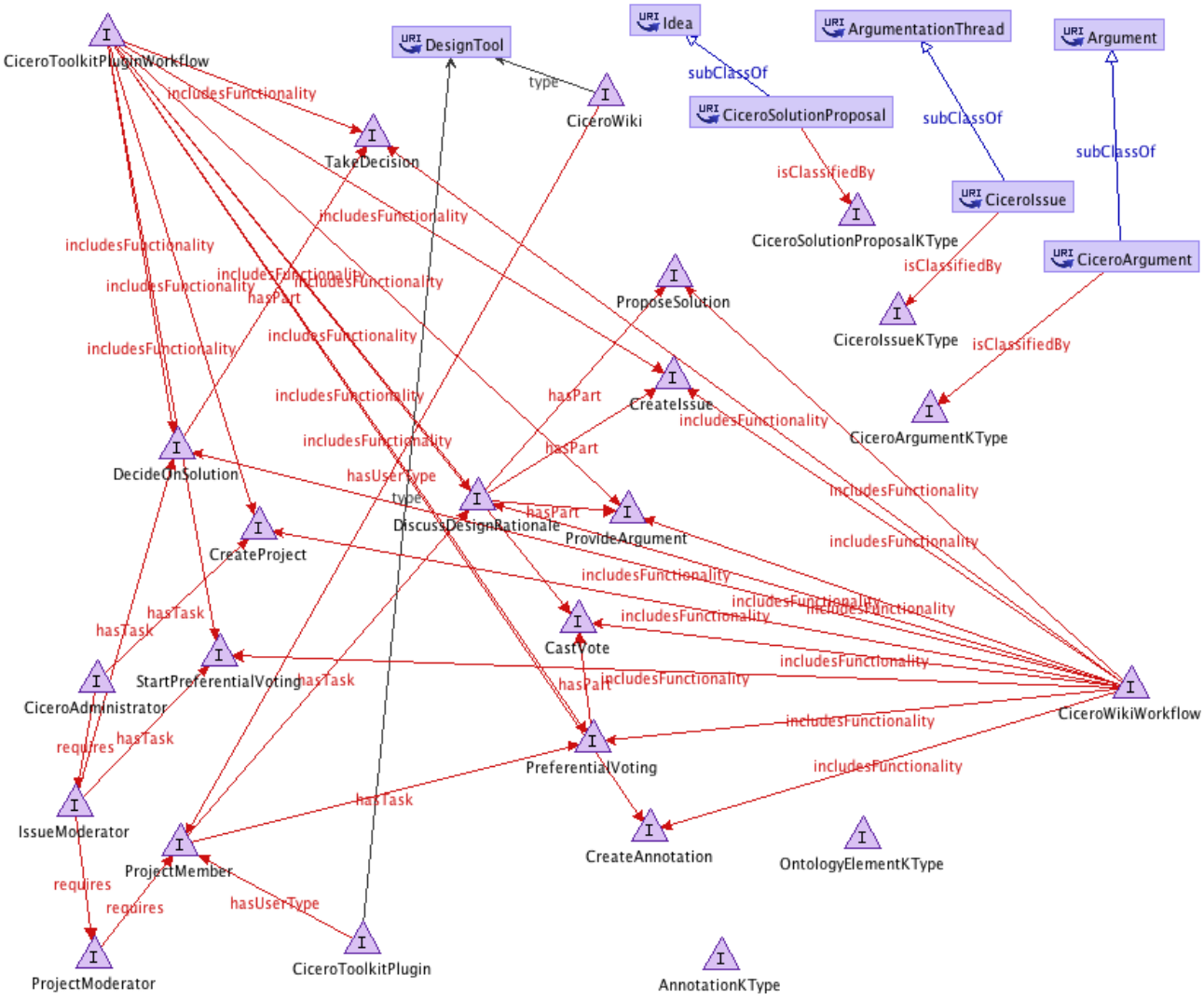

Figure 19: Cicero in C-ODO Light: basic axioms + tasks (functionalities) included in the tool

4.4 Collaboration server (ONTO)

The OntoBroker Collaboration Server acts as a central server with remote access to allow for the distributed usage, management, and editing of ontologies and for true concurrent and collaborative modeling via the NeOn Toolkit. The Collaboration Server is a full-fledged implementation of the KAON2 datamodel.

To use the Collaboration Server with NeOn Toolkit, the user has to choose “Collaboration Server” as a data model instead of the default setting “RAM model”. Since the Collaboration Server is a *remote* data model the user has to properly define the host and port of the machine on which the Collaboration Server is running. In this way a single machine can hold the data model for different clients.

By using the Collaboration Server the NeOn Toolkit supports collaborative ontology editing, allowing for the distributed modeling by any number of editors. This means that the ontologies are

accessible concurrently by many clients. Each of these clients establishes its own session with the server and thus can access available ontologies. All transactions of a client occur within this session. The Collaboration Server maintains a single KAON2 connection internally and thus ensures the overall integrity of the complete data model.

Clients may register for update events. If one client creates or changes an ontology entity or an axiom on the Collaboration Server the ontology is updated. The changes are propagated to all registered clients that can act accordingly, e.g. by updating the views within NeOn Toolkit. For transactional integrity the Collaboration Server locks ontologies to prevent concurrent writing.

The Collaboration Server supports OWL as well as F-logic and RDF. For each of them a single Collaboration Server *connection* is required. A connection to the server corresponds to an "ontology project" of the Neon Toolkit. The Collaboration Server supports all functionalities provided by the KAON2 API, e.g. modeling actions like creating and opening ontologies, as well as reading, writing and editing entities and axioms. Furthermore, searching within the ontology, querying and reasoning tasks are supported.

4.4.1 Example Scenario: Collaborative Ontology Engineering with the Collaboration Server

We assume that there are two clients both willing to perform modeling actions on the same ontology. At the beginning client1 connects with the Collaboration Server. After the Collaboration Server has established the connection, the project is shown to client1. Successively client one creates an ontology ontology1 and a concept named concept1 as well as a subconcept of concept1 named concept2.

When client2 registers to the Collaboration Server he is capable to open ontology1, which was formerly created by client1 and is opened there, too. The Collaboration Server enables client2 to get the ontology elements created by client1, namely concept1, concept2 and the subconcept relation.

If several clients are connected with the same Collaboration Server instance, the server sends all events to each of them. If client2 creates a new concept concept3 and defines it as a subconcept of concept1, the Collaboration Server sends a message about the creation to client2. Additionally the Collaboration Server sends a message to client1. The newly created ontology elements are displayed to all connected users.

4.4.2 Collaboration Server C-ODO Light model

The following figures illustrate the functionalities covered by the Collaboration Server along the same lines as in the previous section.

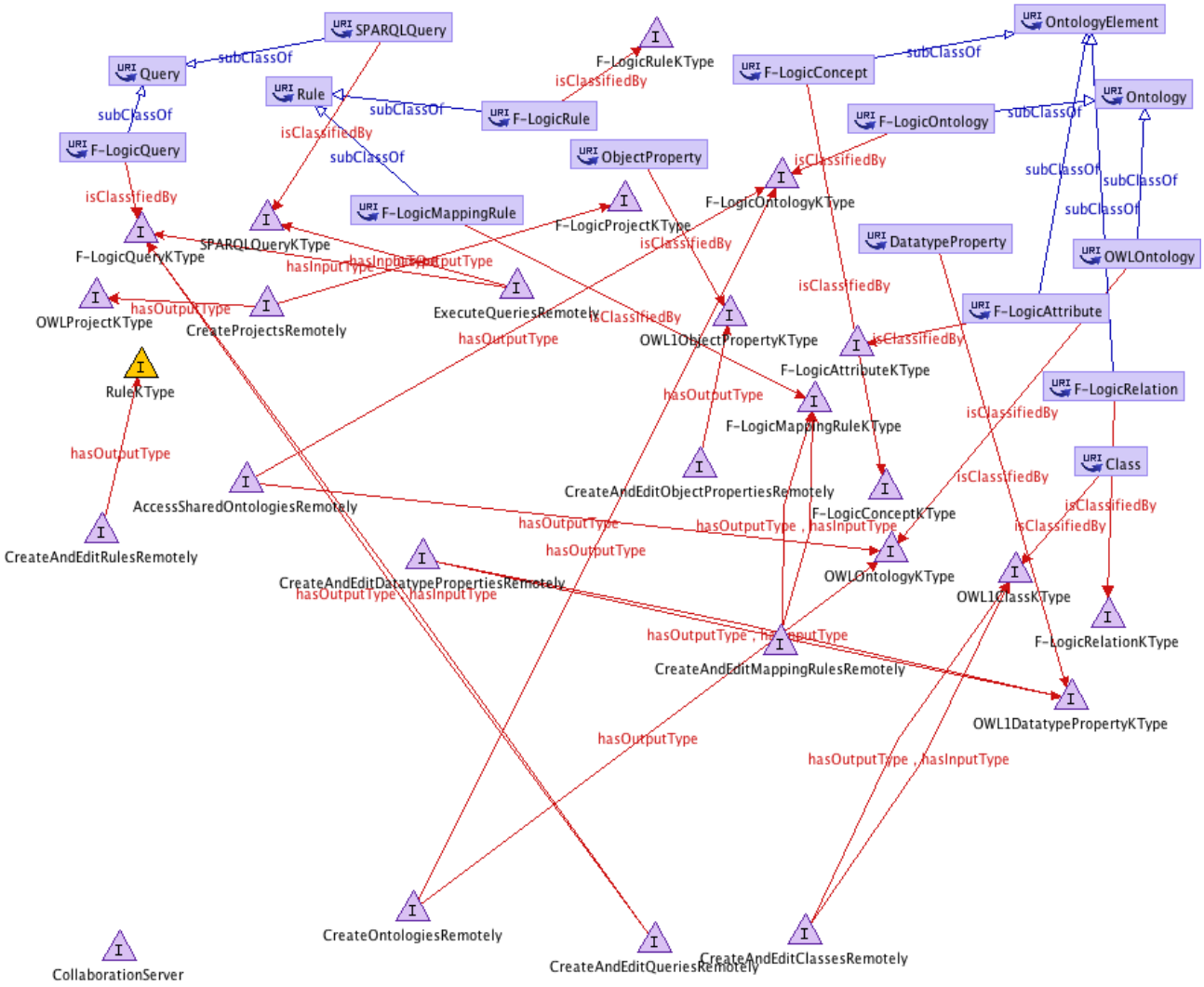

Figure 20: Collaboration Server in C-ODO Light: basic axioms + input/output knowledge types wrt tasks declared for the tool

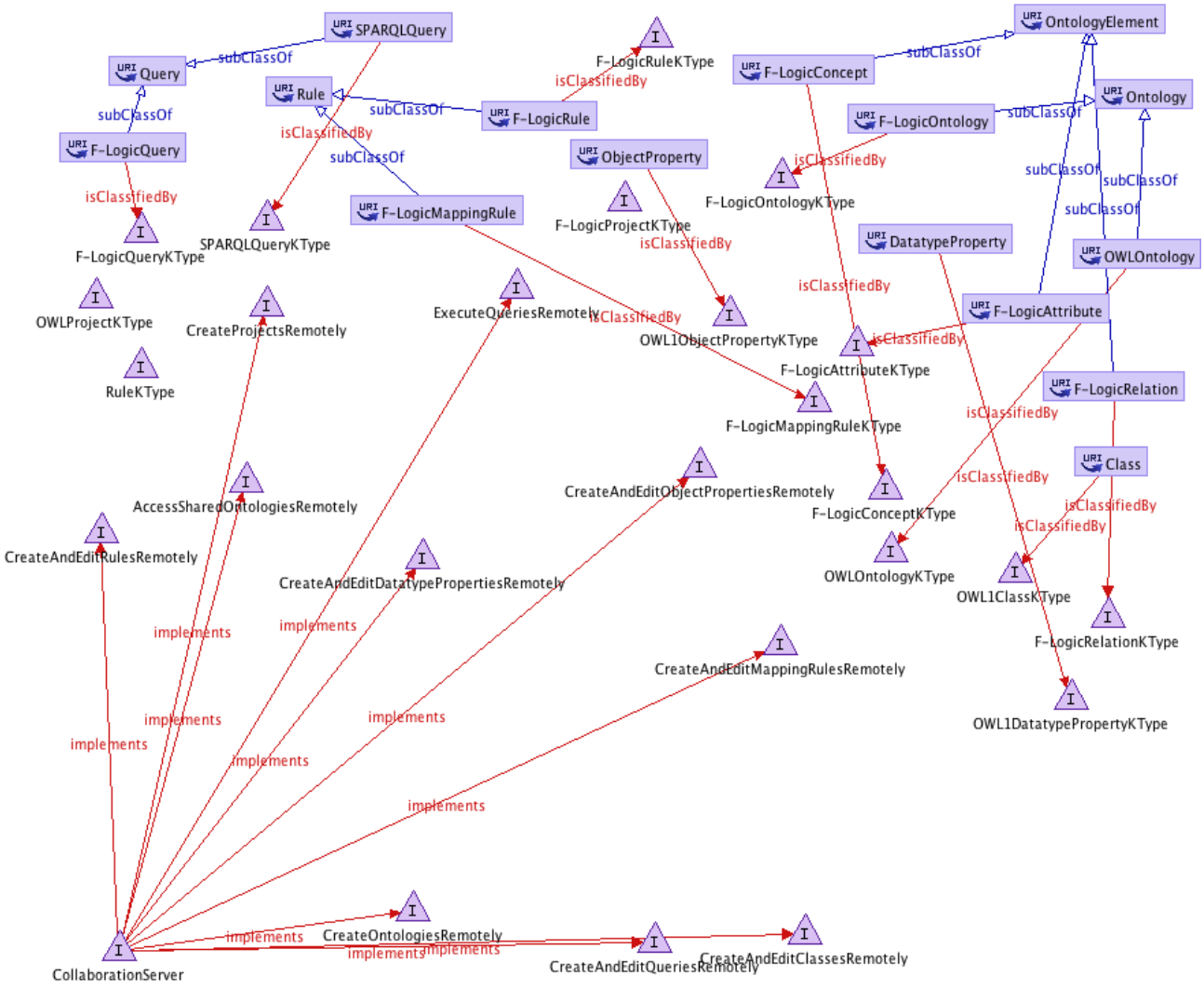

Figure 21: Collaboration Server in C-ODO Light: basic axioms + implemented tasks declared for the tool

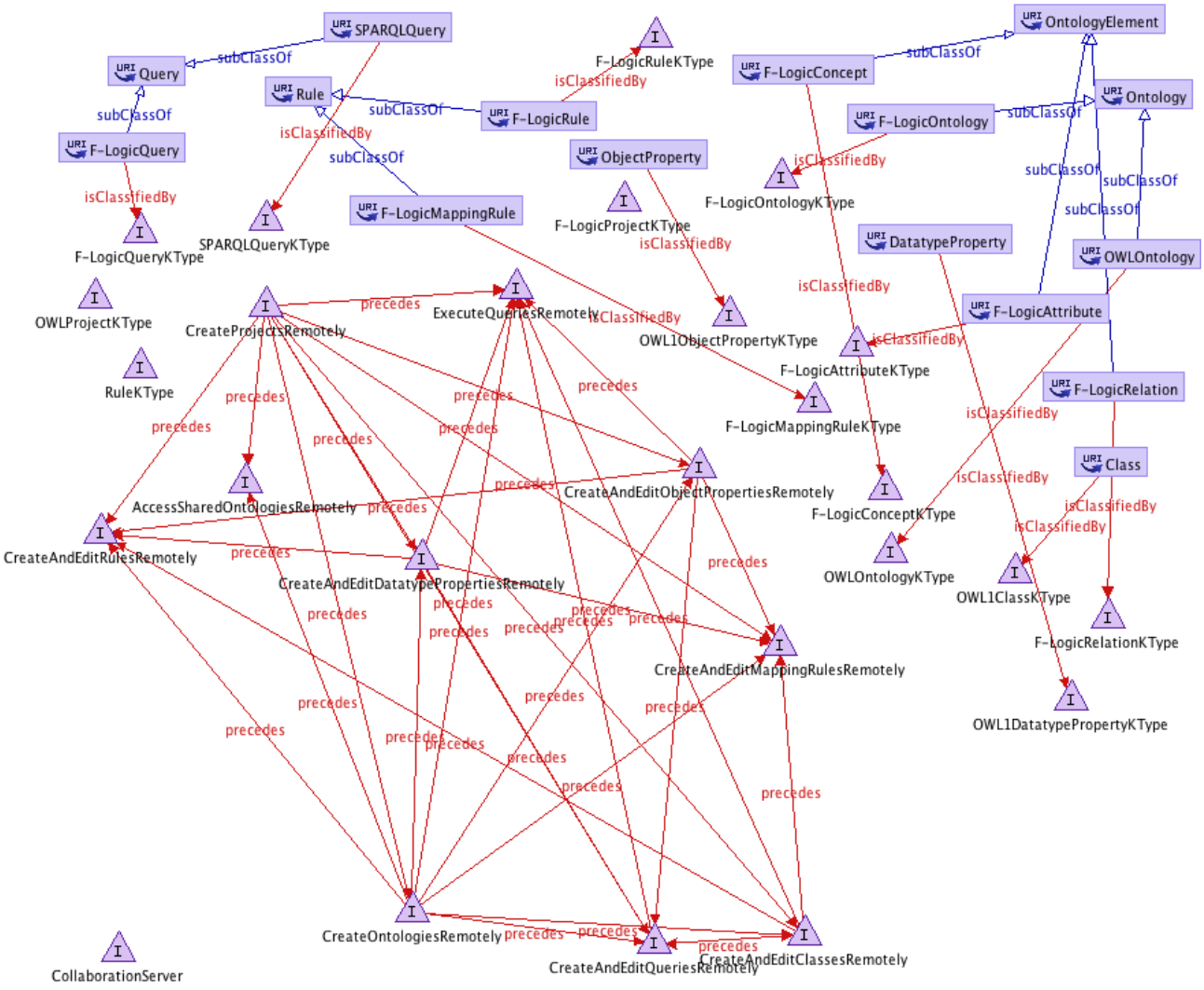

Figure 22: Collaboration Server in C-ODO Light: basic axioms + precedence relations between tasks

4.5 OntoConto (JSI)

OntoConto (NeOn D4.5.2) is an application for visualizing, editing and evaluating alignments and ontologies. There are two main goals of this application.
The first one is to support collaborative editing on networked ontologies. Each human ontology editor can edit an ontology with the clear visualization of networked ontologies and relations. Thus while each user edits a separate ontology, all users cooperate in the sense of a networked ontology.
The application enables the creation and editing of an ontology and results with an ontology constructed in the context of other ontologies.
The second main goal of this application is to support an evaluation of network ontologies (used in NeOn D3.4.1), mainly by helping evaluate alignments between pairs of ontologies. A human evaluator is provided with the visualization of alignments obtained from several different methods provided by the Alignment server. The application provides an evaluator with means to visualize and compare different alignments.
The application enables the evaluation of an ontology and results with an alignment evaluation.

4.5.1 Tool description

OntoConto, developed in D4.5.2, is a NeOn Toolkit plug-in that enables contextualized visualization of ontologies and ontology networks. The idea is to enable the user to browse through an ontology inside a context of related networked ontologies as defined in D3.2.2. Visualization consists of two parts. First, a pair of ontologies is visualized in an intelligent way and second, alignment of these two ontologies is visualized as links between the related concepts. Integration inside the NeOn Toolkit provides an easy access to the loaded ontologies, as well as possibility to edit them. The software incorporates the NeOn Alignment server (D3.3.2), which enables the user an access to several different alignments between ontologies of current interest. OntoConto is also used for evaluation of mappings in D3.4.1.
The screen is divided into two parts, both serving as placeholders for an ontology, within which the user can visualize each ontology. This visualization is based on the underlying tree (forest) structure defined with SubConceptOf relation between concepts. Since ontologies can be too big to visualize all at once, only four layers are visualized at the time and a visualization method for enhancing center of the screen is used (lens). The user can easily traverse the ontology by selecting any concept as the next root node of visualization. Any concept can also be deleted, edited, and new concepts can be added.
After the user loads both ontologies, he/she is enabled to request any mapping generated by the alignment server. All the mappings between currently visible concepts are drawn real time on the screen, connecting the two ontologies. Mappings can also be added, edited and deleted.

4.5.2 OntoConto support C-ODO Light model

Figure 23 below illustrates the functionalities covered by OntoConto in terms of C-ODO Light.

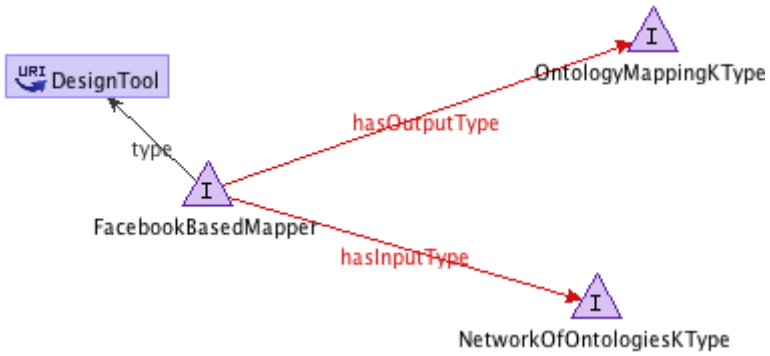

Figure 23: OntoConto in C-ODO Light

4.6 SocialOnto (JSI)

Social network analysis methods in combination with machine learning can be used to analyze collaboration between the users in the networked ontology setting [2]. For the purpose of the analysis, we can assume two distinct scenarios of collaborative work on networked ontology: several experts constructing/editing related ontologies and several experts editing the same ontology. The developed SocialOnto tool focuses on the latter.

In the first scenario, several experts are independently constructing ontologies from similar domains, each expert working on a separate ontology. We can support collaboration between the experts by exchanging information between them, as they are working on similar domains. More precisely, we propose a practical method supporting the experts in ontology construction [1, 3, 4, 5, 6, 7] by showing his/her data in the context of the ontologies constructed by the other experts. This work is performed in collaboration with WP3, providing support for including context in the ontology construction process (NeOn deliverable D3.2.1, D3.2.2) following ideas from [8] and

adjusted for our task to support collaboration via setting the context of other experts. The support lies in visualizing the data in the context of the ontologies generated by other experts, and suggesting the naming of concepts according to the context. In our practical setting, the mapping between the existing ontologies and ontology, which an expert is constructing, is obtained from the alignment server (NeOn deliverable D3.3.2).

In the second scenario, several experts are constructing the same ontology by independently editing different parts of the ontology. In order to support their collaboration, we propose to perform collaboration analysis based on the information on the actions of each expert. The assumption is that for each user a record exists of all the actions the user has performed on the ontology, including the location in the ontology (concept or relation) where the action was performed. In our practical setting, we assume that the tool (e.g., NeOnToolkit or AGROVOC Concept Server Workbench) produces a log-file of the ontology engineering process, in which each line contains information on a specific ontology change including: user id, time, concept or relation id, action performed by the user. In this scenario it is not really important whether the activity is synchronous or not, as we are performing analysis on past activities. This scenario fits the requirements of FAO collaborative ontology engineering in the situation when there are several ontology experts (or several domain experts) changing the ontology. The assumption we have here is that all the used tools are able to produce a log-file containing the needed information. The analysis enables several insights into the collaborative ontology editing based on (1) visualization of the ontology to show what parts were changed by which expert and (2) representing the community of experts as a social network where the connections reflect the experts' activity in the same part of the ontology (enabling calculation of some standard measures such as, computing centrality of the expert, identifying components, etc).

The SocialOnto tool has been developed for this deliverable for processing log files of ontology editing and performing social network analysis on them.

4.6.1 Formal description of SocialOnto by means of the C-ODO ontology

The main goal of the SocialOnto tool is social network analysis, taking into account the activity of different users who are editing the same ontology. It operates within a situation comprising a number of users editing an ontology, and a log file recording the activities of the users.
The social network analysis tool SocialOnto takes log-file of ontology editing as input, where each line of the log-file contains at least timestamp, userId, conceptId or relationId (that the user has edited). It creates a social network (vertex=userId, link=linking similar users, weight on the link = similarity of two users) and performs analysis on the social network in terms of degree centrality, cohesion etc. The tool analyzes the behavior of the users when they edit an ontology, and produces new insights into their collaboration.

4.6.2 Example of Social Network Analysis on Semantic Wiki pages

The SocialOnto tool created for this deliverable has been used for the analysis of interactions between the users while they are editing the same ontology represented by Semantic Wiki pages. The available data is given in change logs as history of changes of the Wiki pages. These change logs are publicly available at http://semanticweb.org/wiki/ Main_Page[3].

Our goal is to provide some insights into the changes of wiki pages by presenting the change logs as two orthogonal graphs. The first is a graph of users connected if they have changed the same page. The second is a graph of pages that are connected if the same user has changed them. In addition we define a bipartite graph connecting users and Wiki pages. The resulting visual representations of the change logs clearly shows several dimensions of the users' activity including

[3] Thanks to Markus Krötzsch and Peter Haase for providing the data.

the most active users, a grouping of users based on the access of the same pages, the most central users and the most frequently edited pages.

4.6.2.1 Data Pre-processing

The data with change logs of semantic Wiki pages as of the end of 2008 were given in an XML form, presented as a sequence of revisions for each page in the Wiki, containing a timestamp and either a username with an internal ID code or an IP of the person performing the page revision. Consecutive changes in page content were also included in the file, but we have discarded them in the pre-processing phase as they are not used in our further analysis. There was a total of 36,078 page edits. Most of those edits (75.5%) were made by the registered users. We have identified 617 registered users and 2,512 different IPs of the anonymous (non-registered) users. For the purposes of further analysis, each unique IP was considered a separate user giving us a total of 3,129 users. We have also performed some merging of user IDs in the process of data cleaning, since there were some cases of people using several usernames when logging in, for instance MaxVölkel and Max Völkel obviously refer to the same person.

The most “collaborative” users can be seen in Figure 15. By collaborative we mean that they are frequently (at least 5 times) editing the same pages as (at least 3) other users. In our visualization, the size of the circle reflects the number of changes the user has made on all the pages, while the color shows if the user is registered (red) or anonymous (blue). We can see that the most active users are registered (the right top part in Figure 15), a few of them are exceptionally active compared to the other users (e.g., the most active user has made 6980 page edits, the second most active 2982, while the 10th most active made 589 page edits).

After a brief inspection, it was determined that there were many very similar pages in the Wiki, for instance: WikiSpammer, WikiSpammers, Talk:WikiSpammer. Since it was conjectured that the same user or group of users with similar knowledge and skills is likely to be editing all of these very similar pages, it was deemed useful to first group together such similar pages into one concept and form an aggregate concepts representation of the data.

When combining the pages we used additional information that we have on the data, namely that the pages represent concepts of the ontology on semantic web represented as Semantic Wiki. However, in the presented work no information about the structure of the original concept ontology was used. The concepts were grouped based on syntactic similarity of their names i.e. page titles and the shortest title was used to label the aggregated concept. Short words that were too common in the concept names were added to the exception list in order to contain a reasonable maximum group size. Some of these words include: Property, Talk, Category, Template, etc. The number of concepts was thus reduced from the original 7,369 to 5,500.

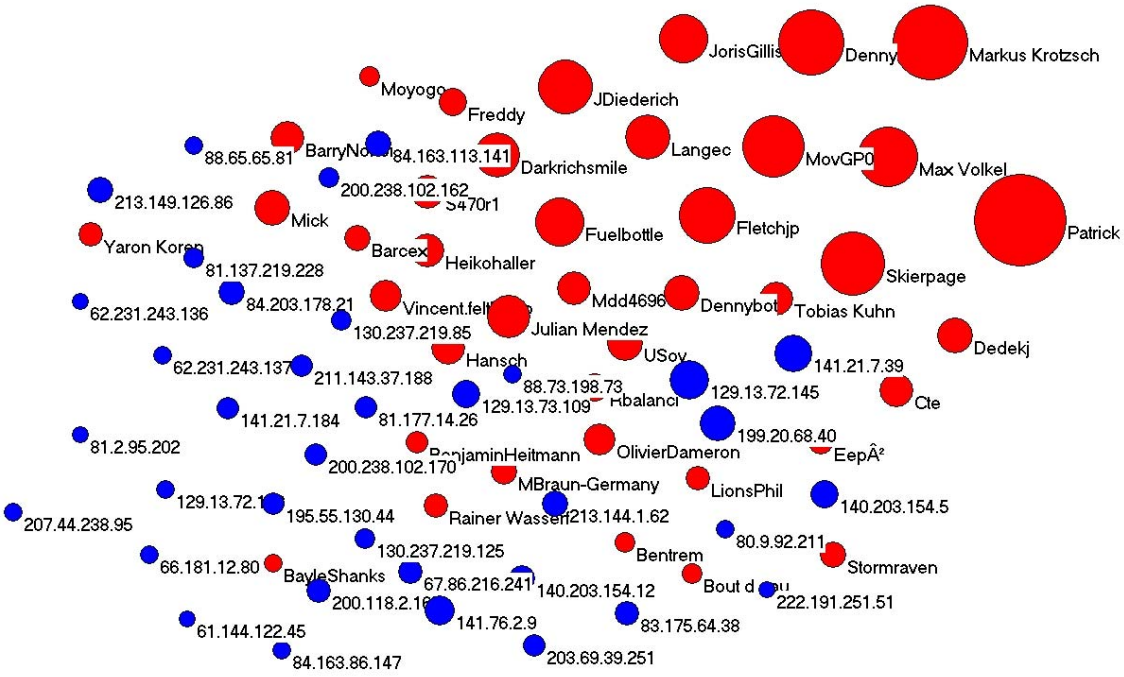

Figure 24: A subset of users that have edited the same pages as at least three other users (min. vertex degree = 3) and share at least five pages with them (min. edge weight = 5).

4.6.2.2 Searching for Relations

After the preprocessing, three different graphs were extracted from the data: a graph of users, a graph of concepts and a combined bipartite graph of concepts and users. These were given as input to social network analysis system Pajek [17] that was used here for data analysis including graph visualizations provided in Figure 24 and Figure 25.

Anonymous revisions accounted for 24.5% of the total number of page edits, so it would not have been beneficial to simply disregard them. Even though the average number of revisions per anonymous user was only 3.5, the maximum number was 251, which was not negligible. The overall most active user was Patrick accounting for 19.3% of the total number, by performing 6,980 page edits in the Wiki. User graph was formed by connecting the users who contributed to the same aggregated concepts, i.e. who had edited pages belonging to the same group of pages. The edges were associated with weights, corresponding to the number of different concept groups that were edited by the same two users. The user graph has 86,401 edges, which is less than 1% of the maximum possible number of edges. Hence, it can be considered a sparse graph. Only about 5% of these edges have a weight of more than one. About 9.7% of the graph consists of isolated nodes. The pages that were edited by these users were not edited by any other user. Most of them were pages about the users themselves (so, most of these users didn't participate in building the wiki any more than making an entry about themselves or their coworkers). However, there were also some exceptions, such as concepts Ontology learning, RELAX SEO Services and Category: Czech person. These were only edited by users Dmanzano, Webmissile and Tom, respectively. In the user graph there are 30 connected components having at least two nodes, but the biggest component comprises most of the users (84%). Average distance among reachable pairs in the graph is 2.23.

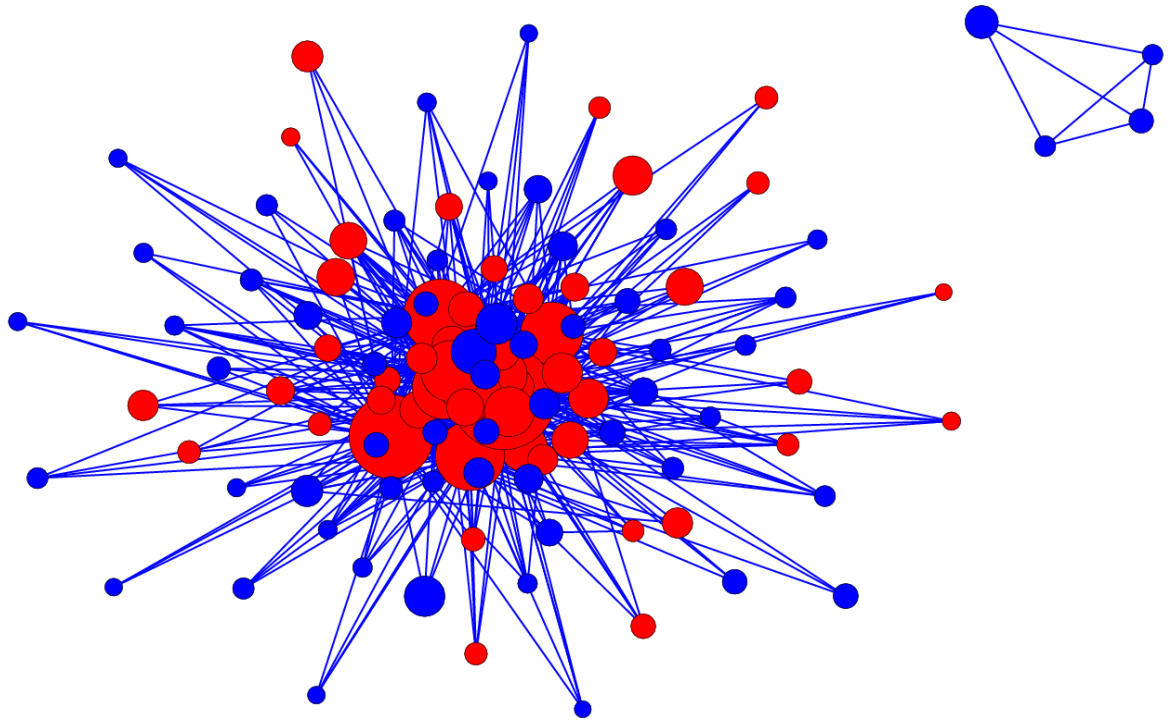

Figure 25: Graph of the most collaborative users (min. edge weight=4, min. Vertex degree=3) showing an isolated group of four anonymous users actively working on a group of 9 wiki pages (top right).

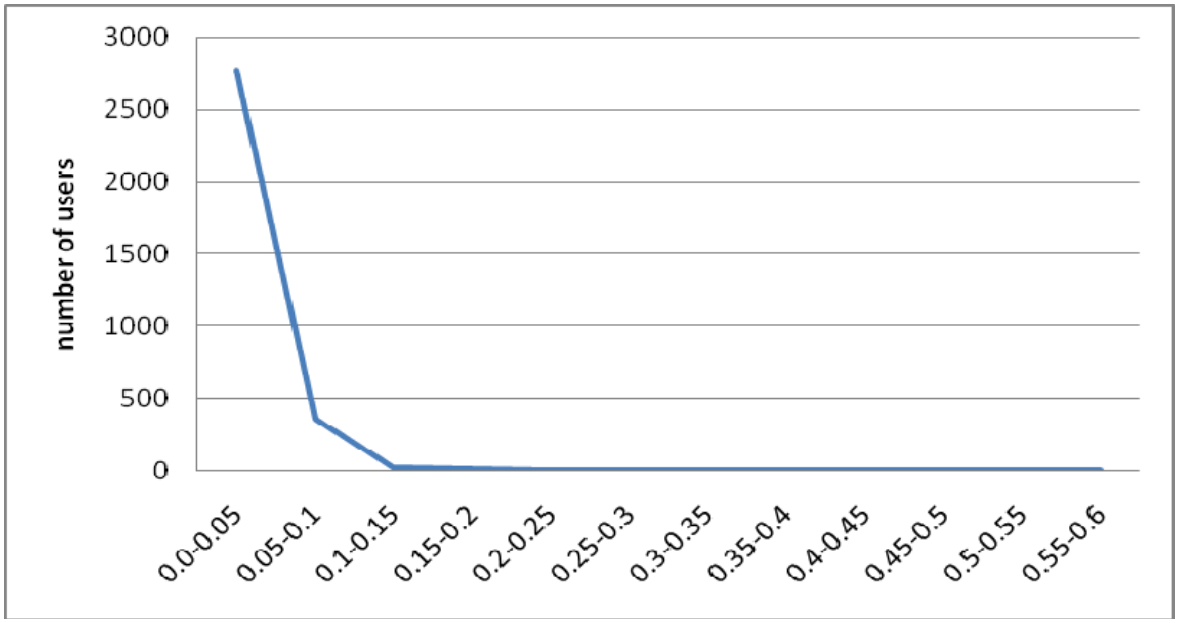

Figure 26: Distribution of degree centrality among the Semantic Web Wiki users

Centrality was calculated for all the users and is presented in Figure 26. It is immediately apparent that there is a small number of very active users in the centre of the network, and a lot of less well centred users. The betweenness centralization achieved in the user network is 0.15. It is interesting to compare the top five users according to these different criteria. Such a comparison is given in Table 1. The users Patrick, Markus Krotzsch and Denny occur at the top in all three lists,

with some difference in their ranking. As can be seen in Table 1, while Patrick has the highest number of revisions, Markus Krotzsch has the highest degree of centrality and betweenness.

Number of revisions	Degree centrality	Betweenness
1.Patrick	1. Markus Krötzsch	1. Markus Krötzsch
2.Markus Krötzsch	2. Patrick	2.Patrick
3.Denny	3. MovGPO	3.Denny
4.Skierpage	4. Denny	4.MovGPO
5.Joris Gillis	5. Knud	5.WikiSym

Table 1Top five users according to three importance criteria from social network analysis

4.6.3 Brief overview of methods for Social Network Analysis

Social network analysis enables analysis of connections between the ontology engineers working on the same ontology or on similar ontologies. The idea is that the system is storing different information that is useful for sharing and information on the sharing process itself. In general, social network analysis deals with mapping and measuring of relationships between nodes in the network. Nodes of the social network might be people, organizations, computers or other information processing agents. Methods provide for visual and for mathematical analysis of relationships between the social network agents.

A method to understand networks and their participants is to evaluate the location of actors in the network. Measuring the network location is finding the centrality of a node. These measures help determine the importance, or prominence, of a node in the network. There are different measures that are commonly used in social network analysis such as betweenness, centrality closeness, centrality degree, centralization, clustering coefficient, cohesion, reach, etc.

- **Degree Centrality**. A node is central in a network, if it is active enough in the sense that it has a lot of links to other nodes.
- **Closeness Centrality**. The most central nodes according to closeness centrality can quickly interact to all others because they are, on average, close to all others.
- **Betweenness Centrality**. A node is central, if it lies on several shortest paths among other pairs of nodes.
- **Network Centralization**. Computed node centralities in a network can have large or small variance. A network where a low number of nodes have much higher centrality than other nodes is highly centralized.
- **Network Reach**. The degree any member of a network can reach other members of the network.
- **Network clustering coefficient**. The clustering coefficient is a measure of the likelihood that two associates of a node are associates themselves. A higher clustering coefficient indicates a greater 'cliquishness' of the network.
- **Network cohesion**. Refers to the degree to which actors are connected directly to each other by cohesive bonds. Groups are identified as 'cliques' if every actor is directly tied to every other actor or 'social circles' if there is less stringency of direct contact.
- **Modularity**. Modularity is a property of a network and a specific proposed division of that network into communities. It measures when the division is a good one, in the sense that there are many edges within communities and only a few between them.

We present several selected measures for social network analysis in more details, namely network cohesion, centrality, betweenness and ranking as presented in [11, 16].

4.6.3.1 Cohesion

Network cohesion is an attractive force between individuals. Solidarity, shared norms, identity, collective behaviour, and social cohesion are considered to emerge from social relations. Therefore, the first concern of social network analysis is to investigate who is related and who is not. Why are some people related, whereas others are not? The general hypothesis here states that people who match on social characteristics will interact more often and people who interact regularly will foster a common attitude or identity.

Social networks usually contain dense pockets of people who ``stick together''. We call them cohesive subgroups and we hypothesize that the people involved are joined by more than interaction. We expect similar people to interact a lot, at least more often than with dissimilar people. This phenomenon is called homophily: birds of a feather flock together.

The ultimate goal is to test whether structurally delineated subgroups differ with respect to other social characteristics, for instance, norms, behaviour, or identity. There are a number of techniques to detect cohesive subgroups in social networks, all of which are based on the ways in which vertices are interconnected. Several techniques can be used to detect cohesive subgroups in a network including density, degree, components and cores.

Density. Intuitively, cohesion means that a social network contains many ties. In network analysis the density represents the number of lines in a simple network, expressed as a proportion of the maximum possible number of lines. It is inversely related to network size: the larger the social network, the lower the density because the number of possible lines increases rapidly with the number of vertices, whereas the number of ties which each person can maintain is limited. This is a problem if you want to interpret or compare network density.

Network density is not very useful because it depends on the size of the network. It is better to look at the number of ties in which each vertex is involved. This is called the degree of a vertex. A higher degree of vertices yields a denser network, because vertices have more ties.

Degree. The degree of a vertex is the number of lines of an individual vertex. Degree is a discrete attribute of a vertex (it is always an integer), so it is stored as a partition.

Components. Components identify cohesive subgroups in a straightforward manner: each vertex belongs to exactly one component. Networks are connected weakly or strongly. A network is weakly connected if all vertices are connected by a semipath. A semipath is a semiwalk in which no vertex in between the first and last vertex of the semiwalk occurs more than once. A semiwalk from vertex *u* to vertex *v* is a sequence of lines such that the end of one line is starting vertex of the next line and the sequence starts at vertex *u* and ends at vertex *v*. A network is strongly connected if each pair of vertices is connected by a path. Strong connectedness is more restricted than weak connectedness: each strongly connected network is also weakly connected but a weakly connected network is not necessarily strongly connected. In an undirected network, components are isolated from one another; there are no lines between vertices of different components.

Cores. When we try to find cores we pay no attention to the degree of one vertex but to the degree of all vertices within a cluster. These clusters are called *k-cores*, where *k* indicates the minimum degree of each vertex within the core. A k-core is though a maximal subnetwork in which each vertex has at least degree *k* within the subnetwork. K-Cores are nested, which means that higher k-cores are always contained in lower k-cores, so a vertex may belong to several k-cores simultaneously. A k-core is not necessarily a cohesive group itself.

4.6.3.2 Centre and periphery

When we talk about centre and periphery in social network (the term *social network* usual refers to a network of people or organizations) we are discovering who's got better access to information or better opportunity to spread the information. The term *centrality* is used for referring to the position of individual vertices within an observed network with undirected edges. A network is highly centralized if there are clear boundaries between the centre and periphery, which means that information can be distributed very quickly between subjects on one hand and on the other side of the centre. It is indispensable for the transition of information.

The larger the number of sources accessible to a subject the easier it is to obtain the information. Hence, the simplest indicator of centrality is the number of connection of vertices to its neighbour vertices (i.e. more sources of information are available). The star network is known to be the most efficient structure given the fixed number of lines (connections). The centralization is much more apparent in star network than in a line network. The vertices vary more with respect to their centrality. Degree of centralization is the variation of the degree of vertices divided by the maximum degree variation which is possible in the network of the same size.

Distance. In an undirected network the distance between two vertices is the number of lines or steps in the shortest path that connects the vertices and is called geodesic distance. In a directed network, the geodesic distance from one vertex to the other may differ from the geodesic distance in the reverse direction so the distance may be different.

Closeness. With concept of distance, we can define closeness centrality. The closer the vertex is to all other vertices the easier information may reach it (i.e., the higher its centrality).

Betweenness. Degree and closeness centrality are based on the reachability of a vertex within a network. Another approach to centrality and centralization is based on an idea that specific vertices are more important as an intermediary in the communication network. This approach is based on the concept of betweenness. Betweenness centralization is the variation in the betweenness centrality of vertices divided by the maximum variation in betweenness centrality scores possible in the network of the same size.

4.6.3.3 Bridges

It is discovered by network analysts that strong ties with people that are themselves related yield less useful information than weak ties with people who do not know each other. Having a lot of ties within a group exposes a person to the same information again and again, whereas ties outside one's group yield more diverse information that is worth passing on. On the basis of this we have to pay more attention to ties between a person's contacts. A person who is connected to people who are not directly connected has opportunity to act as a bridge. It is also hypothesized that people who bridge structural holes have more control and perform better.

In an organization, the social system of the ties is relevant to the diffusion of the information. Can information reach all members of the organization or is it more likely to circulate in one segment of the network. We are interested in who are the bottlenecks (if any) who are vital to the flow of the information. Who may prohibit the spread of the information? That kind of line is called bridge. Removing that kind of line produces segments that are unreachable with information between each other.

A bridge is a line whose removal increases the number of components in the network. When translating to vertices: deleting a vertex from a network means that the vertex and all lines connecting with this vertex are removed from the network. A cut-vertex is a vertex whose deletion increases the number of components in the network. In the same way we can define a part of network that are relatively more invulnerable to the withdrawal or the manipulation of a single

vertex. They are called bi-component. A bi-component is a component of minimum size 3 that does not contain a cut-vertex.

4.6.3.4 Ranking

Prestige. In directed networks, people who receive many positive inlinks are considered *prestigious*. When dealing with social networks, many techniques to calculate the so called *structural prestige* of a person are at hand. Note, however, that structural prestige is not identical to the concept of *social prestige* (or social status), but from the data at hand we are sometimes able to infer a structure so that the structural prestige of a person reflects his/her social prestige. But how do we measure the alignment of a structural prestige to the actual social prestige? For this purpose we can calculate the *correlation* between the structural and the social prestige using the well-known correlation measures such as the *Pearson's correlation coefficient* and the *Spearman's rank correlation coefficient*. The difference between the two is in the fact that the Pearson's coefficient takes the exact numerical scores of the two prestige perspectives into account while the Spearman's coefficient only considers the rankings of the two characteristics. A correlation coefficient has the value between -1 and 1. A value of -1 means that the two characteristics are in a perfect negative correlation (if one is high the other is low and vice versa), a value of 1 represents a perfect positive correlation (if one characteristic is high, so is the other), and a value of 0 represents no correlation between the two characteristics. By the rule of thumb, we partition the correlation coefficient interval into the following classes of *association*:

- **No association**. The absolute value of the correlation coefficient is below 0.05.
- **Weak association**. The absolute value of the correlation coefficient is between 0.05 and 0.25.
- **Moderate association**. The absolute value of the correlation coefficient is between 0.25 and 0.60.
- **Strong association**. The absolute value of the correlation coefficient is above 0.60.

These classes help us interpret correlations in a less technical manner. Another need-to-know heuristic is that we can use the Pearson's coefficient only if its results do not diverge too much from the Spearman's coefficient. If the results are very different, the data contain irregularities. The following sections discuss several measures of structural prestige.

Popularity or indegree. The indegree of a vertex represents the popularity of the person the vertex represents. To measure popularity, we need to have a directed network; in an undirected network the degree of a vertex represents a simple centrality measure.

Domains. Domains represent an effort to extend prestige to indirect inlinks so that the overall structure of the network is taken into account. The input domain of a vertex is defined as the number or percentage of all other vertices that are connected by a path to this vertex. We talk about a restricted input domain when paths are restricted to a certain number of maximum steps. In a well-connected network, the input domain of a vertex often contains (almost) all other vertices so it does not distinguish very well between them. This means that it is better to limit the input domain to direct neighbours (i.e., to use popularity instead) or to those at a predefined maximum distance (e.g., 2).

Proximity prestige. The choice of a maximum distance from neighbours within a restricted input domain is quite arbitrary - the concept of proximity prestige overcomes this problem. When calculating the proximity prestige of a vertex, all vertices within the input domain are considered, but closer neighbours are deemed more important (i.e., the neighbours are weighted by their path-distance to the vertex). More specifically, the proximity prestige of a vertex is defined as the proportion of all vertices (except itself) in its input domain divided by the mean distance from all vertices in its input domain. It ranges from 0 (no proximity prestige) to 1 (highest proximity prestige).

Triad analysis and acyclic decomposition In this section we discuss techniques to extract discrete ranks from social relations. We will first discuss triad analysis as the triad analysis helps us determine whether our network is biased toward unrelated clusters, ranked clusters, or hierarchical clusters. Then we will present a recipe for determining the hierarchy in the social network, namely acyclic decomposition.

Triad analysis. Triads are atomic network structures based on three vertices. There are 16 different types of triads. The triads are named according to the M-A-N naming convention. According to the M-A-N convention, each triad is named with three numbers. The first number denotes the number of the triad vertex pairs that are connected both ways (mutual arcs), the second number denotes the number of the triad vertex pairs that are connected only in one direction (asymmetric arcs), and the third number denotes the number of the triad vertex pairs that are not connected (null arcs). In addition to these three numbers, each triad can have another specifier denoting whether the triad is U, C, or T (we will not go into details about this additional property herein). According to the types of triads that are discovered in a social network, we distinguish between five (or six) balance-theoretic models. The simplest one is the balance model which is limited to only two types of triads, namely triads of type 102 and triads of type 300. The next less restricted model is the clusterability model, followed by the ranked clusters, transitivity, and hierarchical clusters models. The last of these five is still restricted to certain types of triads. Therefore we can add another model to the list - the theoretic model. This model is completely unrestricted. Many algorithms however require the model to be more or less restricted therefore the theoretic model is not used in practice. This discussion has led us to the concept of the triad census. The triad census is a report about all the triads found in the network. The discovered triads are arranged according to the balance-theoretic models to which they belong.

Acyclic decomposition. Once we have determined the nature of our network, we can start discovering the clusters and/or the hierarchy. The first approach we discuss is the so called acyclic decomposition. While cyclic sub-networks (i.e., strong components) represent clusters of equals, acyclic sub-networks perfectly reflect the hierarchy. The recipe for determining the hierarchy is thus as follows:

1. Partition the network into strong components (i.e.clusters of equals).
2. Create a new network in which each vertex represents one cluster.
3. Compute the maximum depth of each vertex to determine the hierarchy.

4.6.3.5 Divisive method based algorithms

Girvan Newman algorithm [13] - Algorithm progressively removes edges with highest edge betweenness from the original graph. If a network contains communities or groups that are only loosely connected by few inter-group edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, we separate groups from one another and so reveal the underlying community structure of the graph.

The algorithm of Radicchi et al. [14]. removes edges that belong to a relatively low number of loops, for they are likely to be edges between communities.

4.6.3.6 Agglomerative hierarchical clustering method based algorithms

The *modularity optimization algorithm* [12] works as follows. Starting with a state in which each vertex is the sole member of one of *n* communities, we repeatedly join communities together in pairs, choosing at each step the join that results in the greatest increase (or smallest decrease) in modularity. This method can be applied to very large networks.

The idea behind *single linkage methods* is to develop a measure of similarity between pairs of vertices, based on the network structure one is given. Many different such similarity measures are

possible. Once one has such a measure then, starting with an empty network of *n* vertices and no edges, one adds edges between pairs of vertices in order of decreasing similarity, starting with the pair with strongest similarity. Structural equivalence is an example of a similarity measure. Two vertices are said to be structurally equivalent if they have the same set of neighbours.

4.6.3.7 Other algorithms

The *spectral bisection algorithm* [12] is a method based on the eigendecomposition of the Laplacian matrix. The eigenvector that corresponds to the second lowest eigenvalue determines a partition of nodes into two communities. Some approaches enable combining structure analysis with content analysis as described in [10].

4.6.4 SocialOnto C-ODO Light model

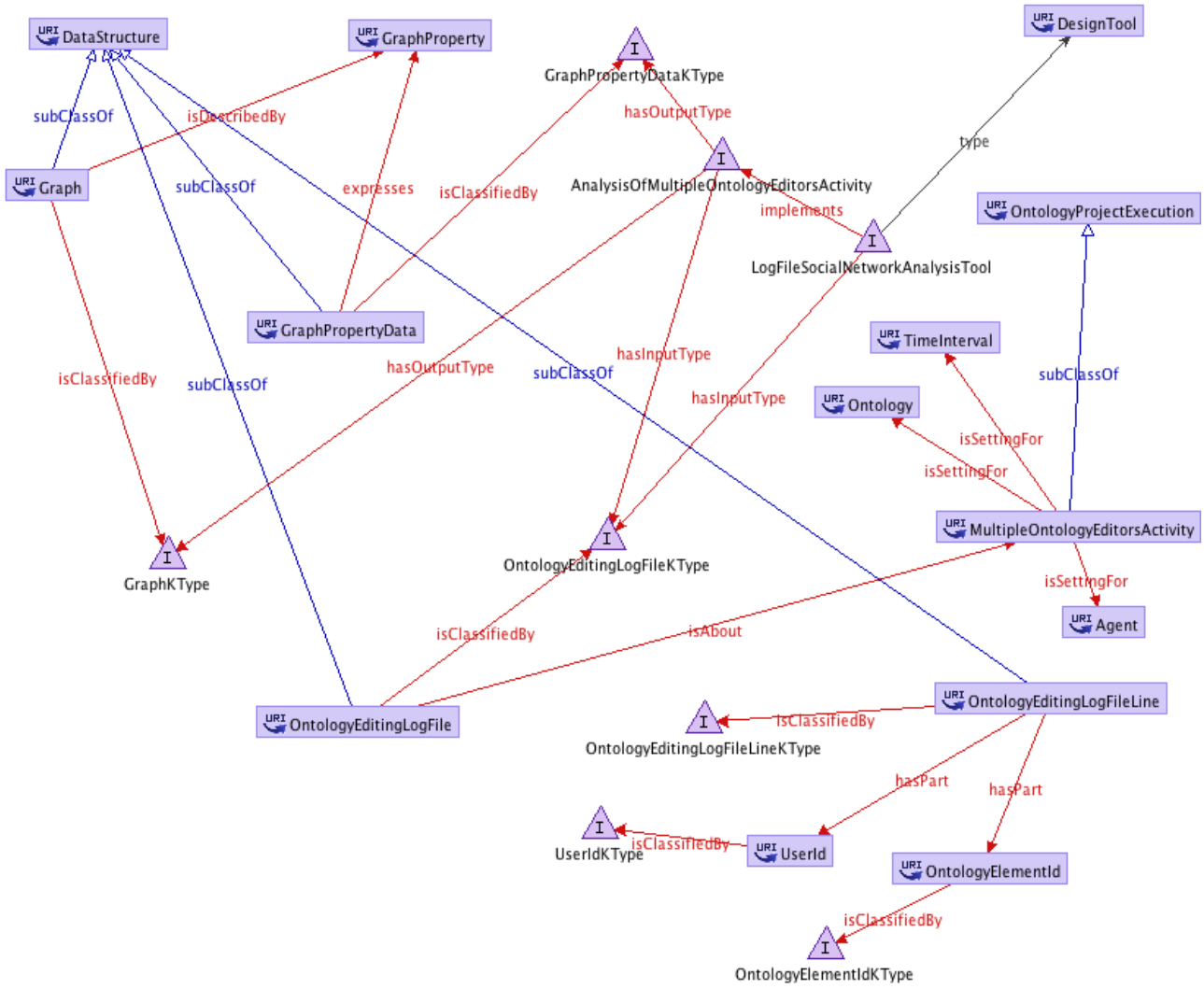

Figure 27: SocialOnto in C-ODO Light

4.7 COAT: Collaborative Ontology-based text Annotation Tool (USFD)

4.7.1 Introduction

COAT has been developed as NeOn plug-in to address distributed collaboration amongst annotators. Annotators need to be able to flexibly work on any number of documents associated with a particular task, for which activity they do not necessarily have to share the same location when performing the annotation tasks.

The text annotation tasks are performed within the GATE[4] system. GATE [18] is a framework and graphical development environment, which enables users to develop and deploy language engineering components and resources in a robust fashion.

GATE supports the following text formats: XML, PDF, RTF, Microsoft Word, HTML, SGML, email and plain text. Not all versions of PDF and Microsoft Word are supported.

Whenever a document is created or opened in GATE, the format is analysed and converted into a single unified model of annotation.

The annotation format is a modified form of the TIPSTER format [19], which has been made largely compatible with the Atlas format [20], and uses the now standard mechanism of 'stand-off mark-up[5]' [21].

The user of COAT will be able to access and process documents in this format on the GATE annotation server.

4.7.2 The Functionality of COAT in CODO Light terms

This section describes the functionality of the COAT web service. In a nutshell, COAT enables annotators to load and annotate documents with annotation types, which either refers to ontology classes, or to new candidate classes for categorization.
Coat enables the manual semantic annotation of texts with references to a browsable ontology. The human text annotators can collaboratively extend and amend the text annotations by adding, deleting or modifying annotations and annotation types.

[4] See http://gate.ac.uk/

[5] For more information, please see http://gate.ac.uk/sale/tao/index.html#x1-1450006

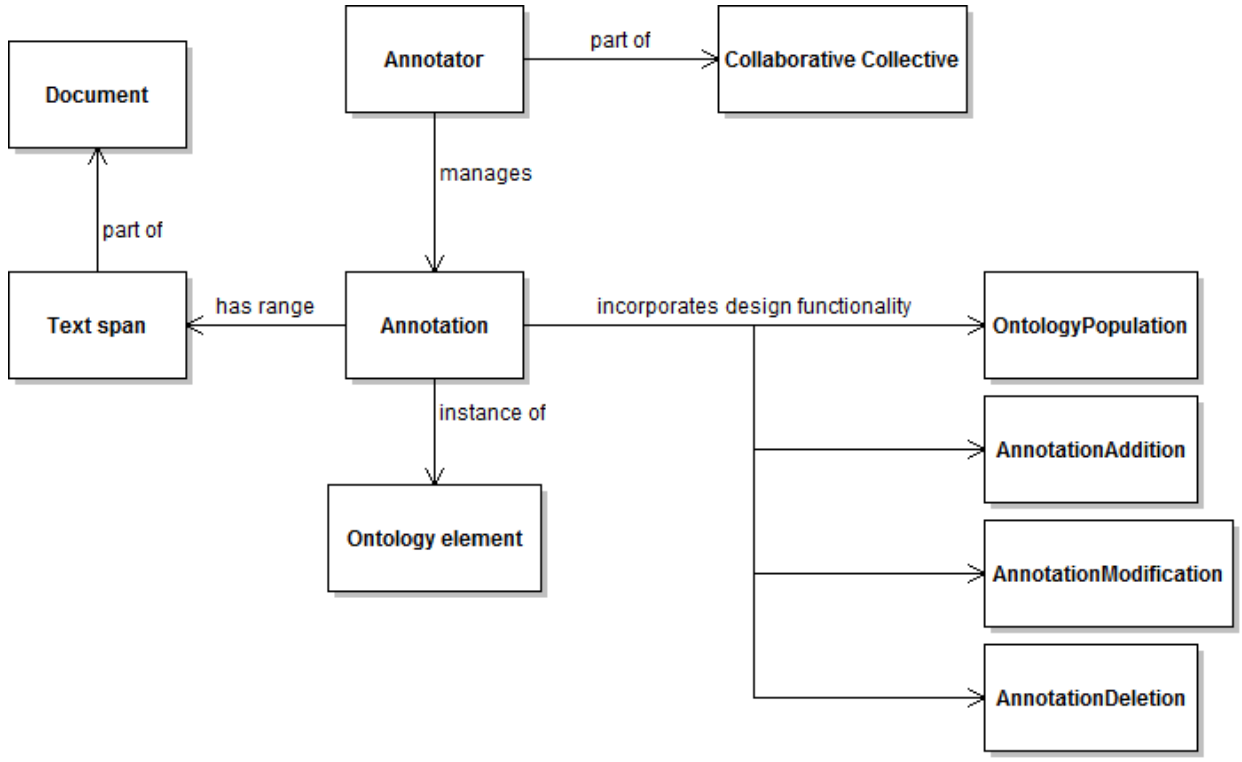

Figure 28: a simplified overview of COAT's functionality

A description in terms of the more fine-grained CODO Light conceptualization of COAT's workflow follows below. The elements from the CODO Light ontology used in this description are in italics.

The main functionality of *COAT* is *ontology population*. This is achieved by defining a *workflow* that involves human *text annotators*, and the distributed collaborative annotation of *text spans* to *ontology* classes as instances. The human *text annotators* can collaboratively extend and amend the text *annotations* by accessing the files, and *adding*, *deleting* or *modifying* annotations. The details of the annotation process are described below in section 4.7.2 below.

The *COAT workflow* involves the following participants:

1. a number of human text annotators who populate an ontology;

2. an optional ontology containing the allowed classes to be used as annotations (either populated with instances from the documents or not);

3. a collection of documents.

The main restriction on COAT's collaborative process is that two annotators cannot simultaneously work on the same file. Once one annotator is working on a file, it remains locked until she finishes her session.

The distributed nature of the annotation process implies that annotators do not necessarily have to share the same location when performing the annotation tasks (*ComputationalDesignTasks*).

The *texts* may have been pre-processed by any annotation tool, the result of which can be used in the annotation task.

Also, given the fact that annotators can work on each others products, the annotation process involves manual creation and verification of ontology instances.

The tool allows the annotators to work on a collection of documents, each of which can be selected and processed, and an ontology containing the allowed classes to be used as annotations (either populated with instances from the documents or not).

Figure 29 below illustrates the CODO-based conceptual design of COAT.

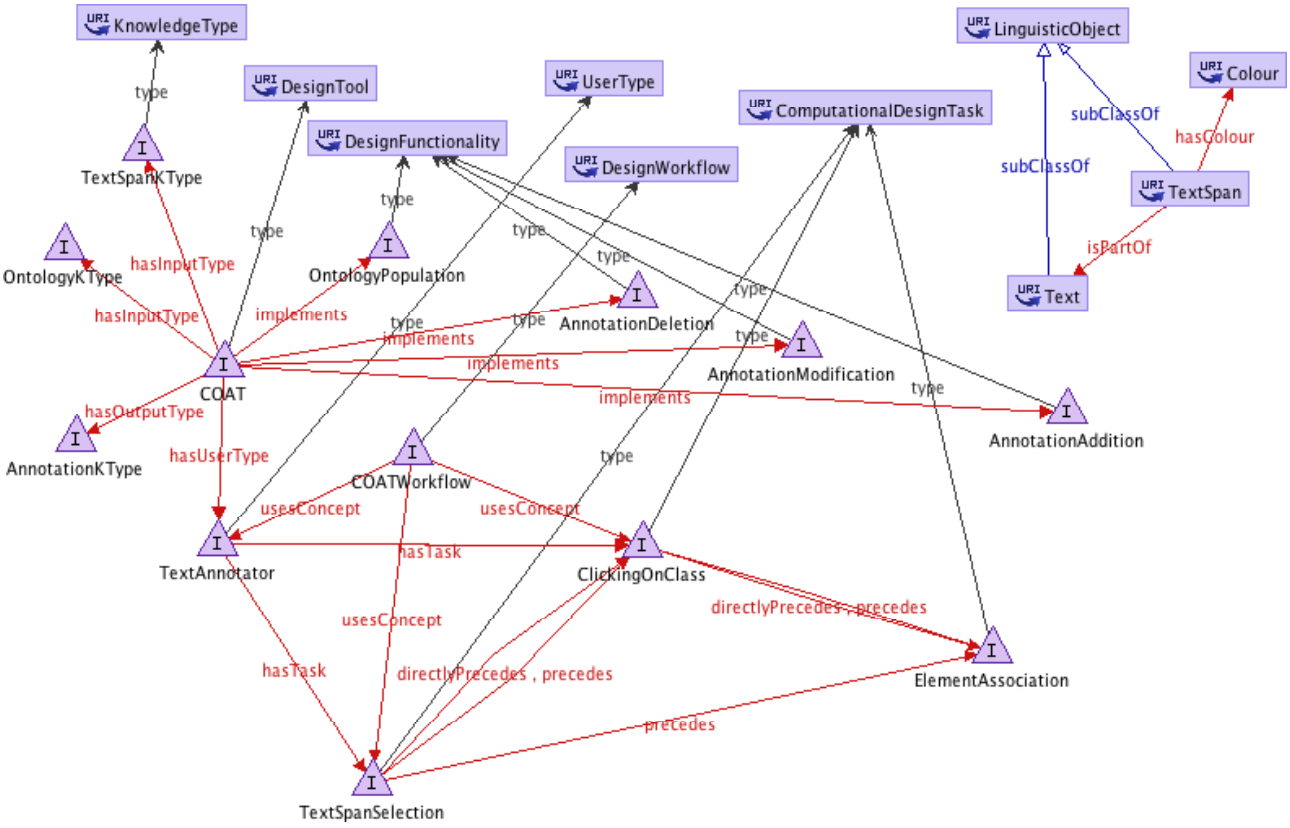

Figure 29: C-ODO Light model of COAT

4.7.3 How to use COAT

Since, we present its user manual in this section, since COAT is an integral part of this deliverable.

Invoke the web service through the NeOn Toolkit by clicking on the “GATE Services” menu item, and the “select COAT” option.

A web browser will open the COAT main page (http://safekeeper1.dcs.shef.ac.uk/neon/coat/).

The main page offers two options:

1. View corpora
2. Upload documents into corpus.

I. **Main page option 1: view (and create) corpora**

The page will show a list of available corpora with the documents they contain.

It is possible to create a new corpus on the bottom half of this page. A newly created corpus needs a name and an optional description.

II. Main page option 2: load documents into corpus

The documents need to be compressed into a zip file. Each zip file should contain at least one document.

As mentioned above, COAT accepts the following document formats:

- Plain Text
- HTML
- SGML
- XML
- RTF
- Email
- PDF (some documents)
- Microsoft Word (some documents)

For each document in the zip file, COAT will list whether it has accepted the file and loaded it into the corpus.

III. Corpus details and choice of document and ontology

When you click on a corpus after choosing main page option 1, a new page shows the list of documents contained in the corpus. Clicking on one of the documents selects it as the document you are going to annotate.

The “Select ontology” box shows a list of ontologies available on the GATE annotation server. Choosing the option “Other, enter an ontology URL in the box below” in this list allows you to select either a web based ontology (e.g. http://proton.semanticweb.org/2005/04/protonu), or an ontology from your local file system (e.g. file:///c:/ontologies/protonu.owl). You can choose this option automatically by clicking in the "enter an ontology URL" box.

IV. Activation of the annotator GUI

Pressing “Open” will activate the annotation service, which starts up as a Java Webstart application.

If your browser asks you to open the application, press “yes”.

A new application window, the Annotator GUI, will now show the text in the left hand pane, and the (expandable) ontology in the right hand pane (see Fig.30). The ontology loaded for this example is the Proton[6] ontology.

In this mode, the top left pane will remain empty and can be minimised by clicking on the little black triangle at the bottom of the pane, in order to maximise your view of the text.

The “Options” tab in the right hand ontology pane has a number of useful options:

[6] http://proton.semanticweb.org/

Case sensitive “Annotate all”: annotate all text spans identical to the present one with the ontology class.

“Disable filtering” will list all ontology classes. Filters can be applied by ticking either “Classes to omit”, or “Classes to show”, which both allow loading a file with ontology class labels (one per line).

The user is advised to ignore the other options on this tab.

V. Text annotation

Select a text span by double-clicking the text span if it consists of one word, or dragging the cursor over the text span if multiple words are within this span.

Associate an ontology class with it by either:

- selecting the appropriate class from the drop-down menu, or
- typing in the name of the class in the pop-up window and pressing enter, or
- typing the name and selecting from a drop down menu filtered by the characters you have typed in. This feature is useful if you are working with large ontologies.
- double-clicking on the class in the ontology pane, ignoring the pop up window.

VI. Attribute instantiation and creation

In the pop-up window, attributes of the ontology class can be instantiated by selecting the appropriate ones from a drop down menu, and typing the value of the attribute in the value box. If you wish to create a new attribute, this can be performed by typing the attribute name in the empty field marked by a yellow “C” in the attribute list, and filling in the value.

When finished with the annotation of the text span, the text span will now be highlighted with the colour of the concept of which it is deemed an instance.

The pop-up window can be discarded by either

- pressing the “x” in the top right corner, or
- highlighting another text span for annotation

Once the annotation has been created, the pop-up window will come up again when the cursor is hovering over a coloured text span. This window will disappear again if the cursor is moved to another annotated text span. It can be permanently displayed by pushing the red peg.

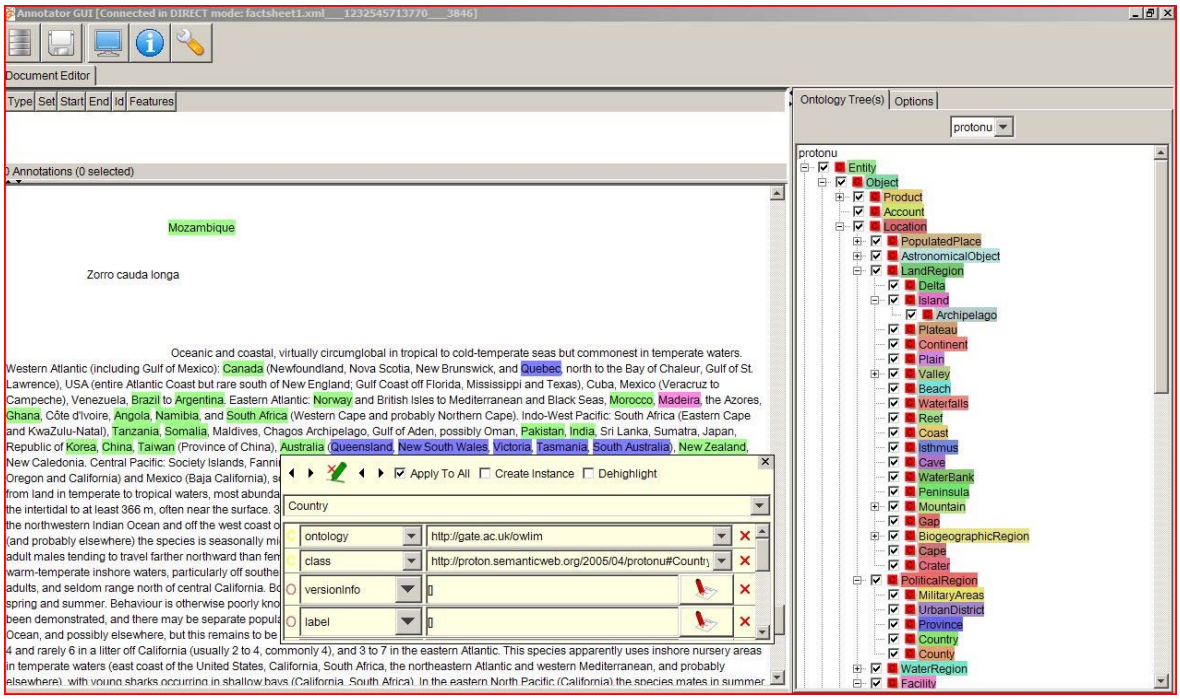

Figure 30: Annotation with ontology classes

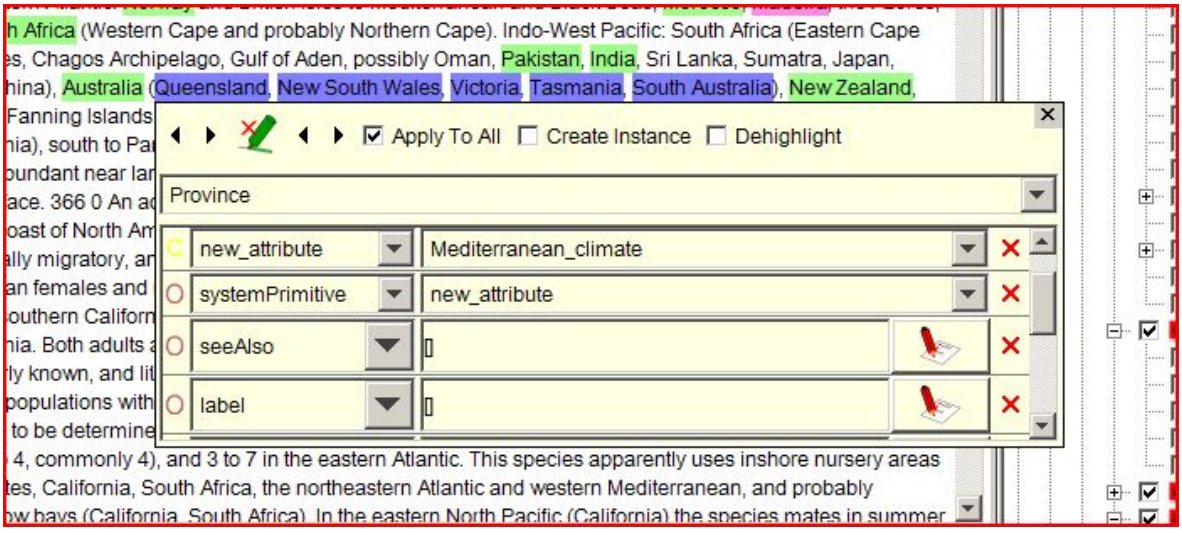

Figure 31: Addition of attributes

VII. Addition of new annotation types

If you do not want to annotate text spans with ontology classes, but want to identify interesting informally defined categorizations in the text, you can opt for adding new annotation types. These are categories that are not part of any ontology, but are either agreed upon by the annotators, or defined ad hoc by the annotator. They can serve as candidates for future inclusion into an ontology.

You can go about this in two ways.

1. “without ontology” mode: You can start off with the annotation process without an ontology by opening an application, for which you select a document without loading an ontology (choose the “No ontology” option in the “select ontology” box. New annotation types can be created by typing their name in the pop-up window. Any text spans previously annotated as class instances are now displayed as members of the class Mention, which has the ontology class as an attribute for each text span.
2. If you are working on text annotation with ontology classes (“with ontology” mode, as described in V above), but want to switch to adding annotation types, you press the “Connect to Document Service” button on the top left of the Annotator GUI (see encircled option in Figure 21).
   Then delete the value in the “Ontology URL” field and press “Connect”. This will change the operation of the application to “without ontology” mode, and displays, in the same vein as in 1. above, any existing ontology classes as instances of “Mention”.
   Switching back to “with ontology” mode can be done by pressing the “Connect to Document Service” button again, putting the ontology URL back in the “Ontology URL” field, and pressing “Connect”. Alternatively, you can exit the application, and start up again in “with ontology” mode as described in III. above.

Figure 32: Buttons in Annotator GUI

For each new annotation type any number of attributes can be added and instantiated with a value.

Figure 33 below illustrates this for the newly created annotation type ScientificTerm (with “oophagous” as instance), which has been enriched with the attribute “root”. The pink text spans in the figure are all ontology instances whose creation is described above. The top pane shows that these instances of the “mention” class have the class URI as feature.

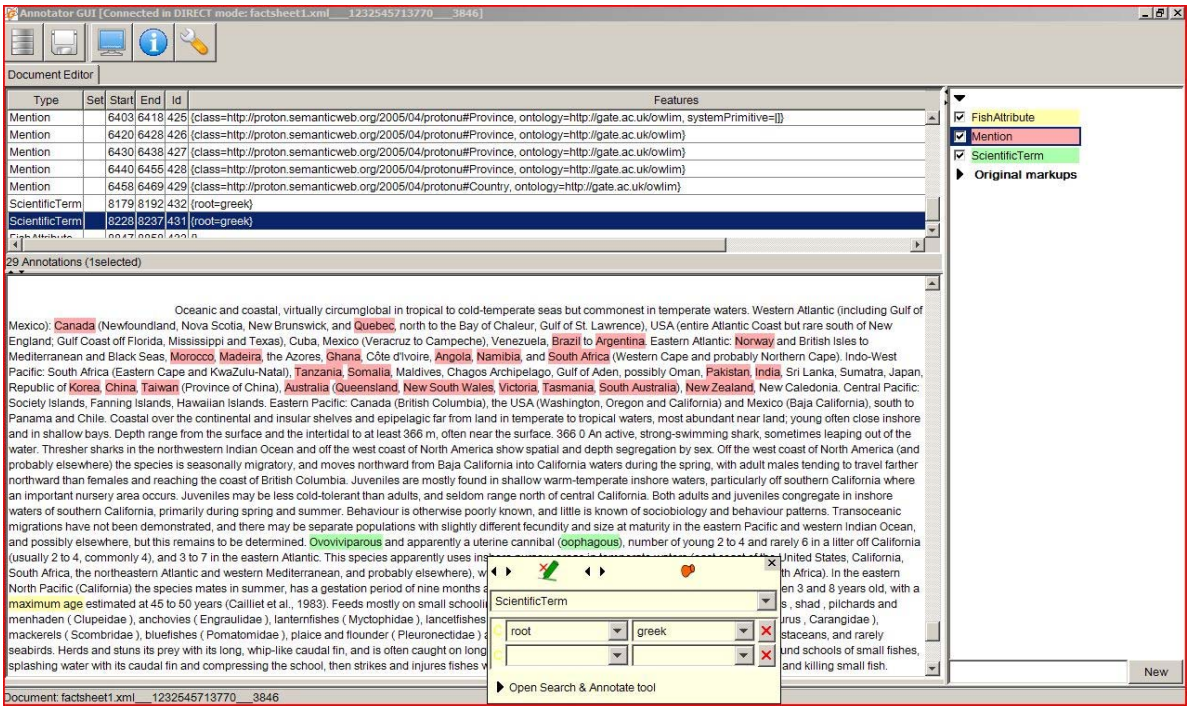

Figure 33: Annotation Types

VIII. Saving the annotations

The annotations can be saved at any time by pressing the save button (second from left on top pane see Fig.21). The documents and their annotations will be saved on the server, and will be accessible. There is no option yet to save the documents locally. This functionality will be provided in the next version of COAT.

IX. Remaining Annotator GUI buttons

The other buttons on this pane that are not yet discussed are the following:

 sets the look and feel of the interface

 shows the application log

 shows this manual in a pop-up window

X. Housekeeping

Metadata about corpora, documents and annotations are not visible within the Annotator GUI.

Behind the scenes, each corpus and document receives a time stamp with the date of creation/upload.

Also, the GATE web server keeps track of access, and records time and agent.

We expect to integrate these data in version 2 of COAT.

XI. Future work

The next version of COAT will incorporate the following additional functionalities:

- The deletion of files and corpora if the user does not want the data to stay on the server, from which they are openly accessible. At present, any data can be removed solely by the GATE team.
- The creation of an export facility that allows the user to save the annotated text locally.

4.8 C-ODO Light-based ODP repository management (CNR)

The eXtreme Design tool for NeOn toolkit (XD) offers the possibility to make operations on ontology design patterns (ODPs). XD has its own conceptual definition according with the C-ODO light ontology, implemented as an OWL file. XD uses C-ODO Light as a reference model for implementing its functionalities. Furthermore, part of its behaviour is determined by C-ODO Light-based ontologies. Concepts such as OntologyLibrary, Ontology, OntologyDesignPattern, Content Ontology Design Pattern (ContentOP) are sample items that the XD plug-in manages. The concept of networked ontology is native in XD. It provides the user with pattern-based operations, such as specialization, composition, instantiation, etc [cf. NeOn D2.5.1] instead of operations based on a certain logical language such as subclassOf, etc.

4.8.1 Use of C-ODO for managing Ontology Repositories: the ODP library browser

The ODP library browser (Fig. 35) is integrated in XD. It allows the user to directly access the ODP repository of patterns. The ODP repository (Fig. 34) is modelled itself as an OWL ontology based on C-ODO-light and it is used by the XD plug-in of the NeOn toolkit for browsing the ODP repository. Accessing the ODP repository by the ODP library browser offers to the user direct access to the patterns/ontologies that can be (re)used within the XD tool.
The use of C-ODO light in the ODP repository instance has been done by generating an OWL representation of the folder structure of the file repository. In the ODP case, each folder has been represented as a coddata:OntologyLibrary (according with the C-ODO definition), which can host a collection of codkernel:Ontology instances. This ontology[7] represents the whole content of the ODP repository. XD uses this representation to build the navigation tree of its ODP library browser, as depicted in figure 35. The ODP library browser shows the folder structure of the repository, but at the core level it is reading the OWL representation. Other kinds of representations of the repository can be provided by means of the C-ODO description.
Using OWL modelling and the C-ODO ontology to represent the repository brings various benefits. First of all, it provides flexibility, since any repository can be described through a codolight-based ontology, and this would be enough for making it available through the XD tool. XD tools allow, through the C-ODO light support, the reuse of ontology libraries from external providers.
XD can be implemented providing different functionalities depending on the ontologies described in the library. Within the C-ODO light description of the repository, providers can describe an ontology as a ContentOP, defining which functionalities XD tool can enable. A ContentOP can be described as a specialization or composition of other patterns, describing also the relations of a certain ontology to others in the library.
All this information can be provided by the repository ontology in order to describe how an ontology library is organized. This information can be also exploited by the tool for customizing operations on the library. The XD tool will also use C-ODO light annotations to navigate the library by means of the relations described.

[7] hosted at http://www.ontologydesignpatterns.org/schemas/repository.owl

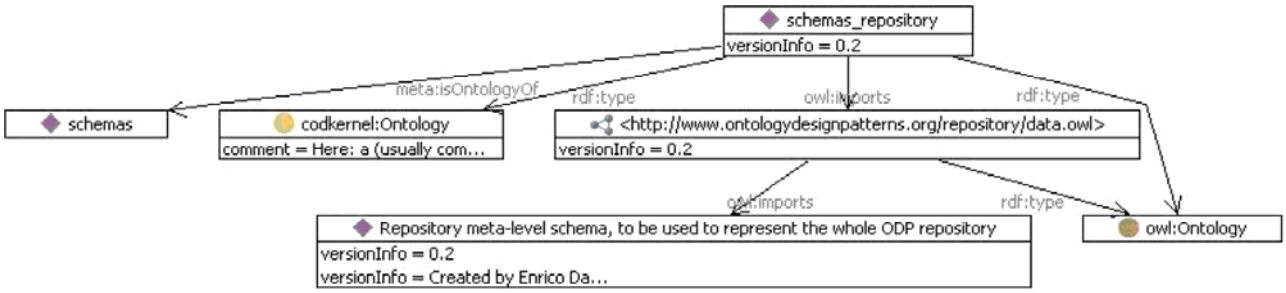

Figure 34: the ODP repository

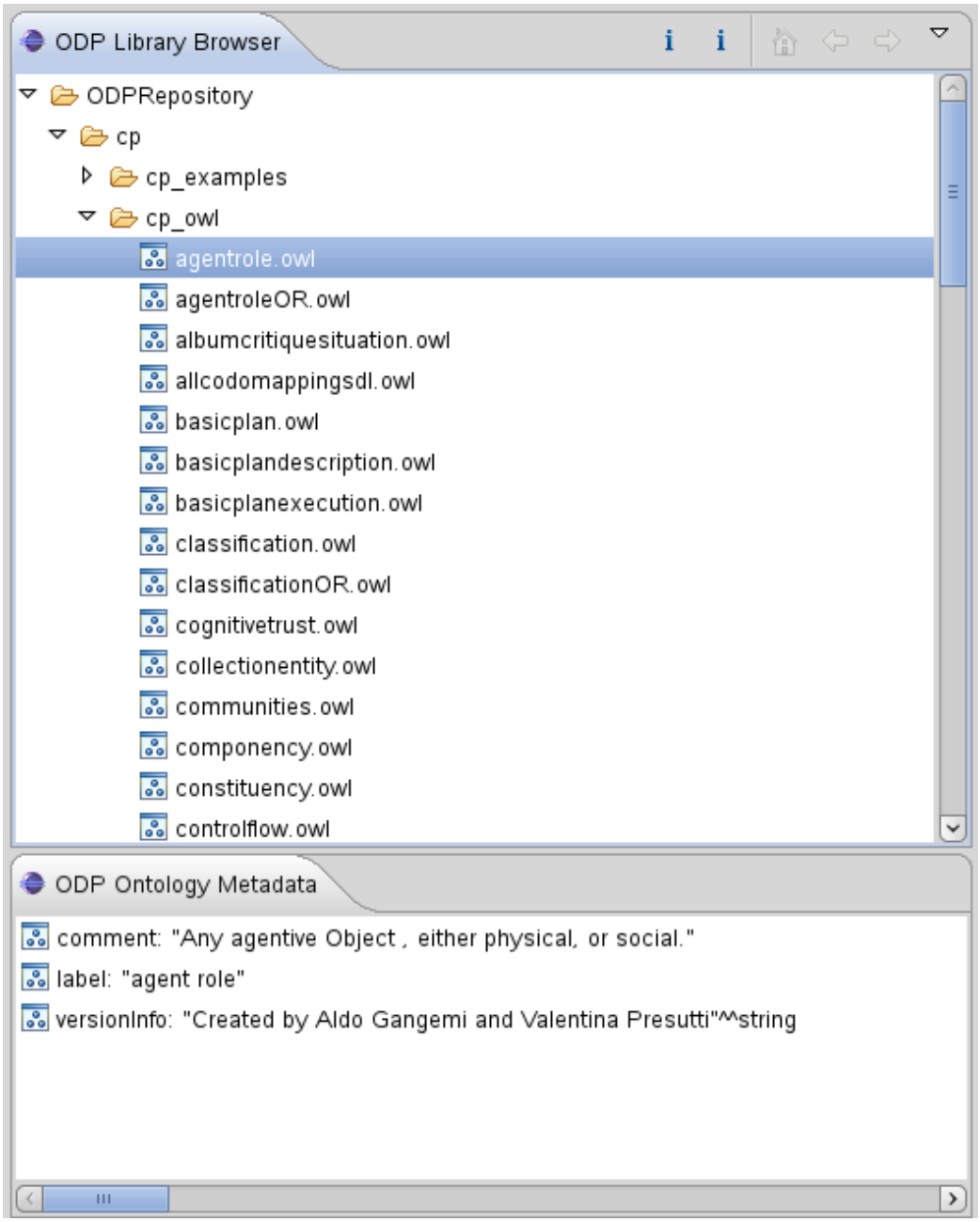

Figure 35: the ODP library browser

The XD description as a C-ODO ontology can be found at the following url:

http://www.ontologydesignpatterns.org/cpont/codo/xd.owl

http://www.ontologydesignpatterns.org/cpont/codo/xd2codo.owl holds alignments to the C-ODO knowledge types.

The ODP repository can be browsed at http://www.ontologydesignpatterns.org/repository.

5. Conceptual navigation of design tools and the Kali-ma plugin

Most of the design-oriented tools presented in this paper come with a conceptual model based on C-ODO Light. Each of these models provides several arguments for supporting the assessment and organization of the respective tools with regard to ontology design principles, yet software support for design-oriented ontology authoring within the NeOn Toolkit is scarce. This chapter presents the Kali-ma plug-in for the NeOn Toolkit, which leverages C-ODO Light-based descriptions in order to support end-users in managing ontology lifecycles based on design principles. While a detailed description of the plug-in: its architecture, implementation, and testing, is deferred to a dedicated deliverable planned for Y4 (D2.3.3), outlining some of its key features here is useful due to the fact that Kali-ma exploits the formal descriptions outlined in this deliverable to present design tools on a uniform level that can be shared in a collaborative environment.

5.1 Querying the tool space

Codolight models of tools can be used in several application tasks, including:

1) browsing semantic data about ontology projects
2) smart searching and selecting of design components
3) creating custom design configuration interfaces
4) helping ontology requirement collection
5) providing a shared network of vocabularies.

(1) The network of ontologies emerging from the models of plugins presented in this deliverable conveniently allows us to know more about the plugins: what knowledge types do they take in input or produce in output? What functionalities are implemented? What user types are allowed for a certain functionality? What sequence, if any, exists between functionalities? And once tools are used in real ontology projects, we can also mix the basic data with data produced by the tool, e.g.: what users have actually used a functionality on what knowledge resources? What tools can take as input a certain knowledge resource?

(2) With existing tools, it is not trivial to find the right functionality from the viewpoint of a user, especially if that user is not an expert in ontology design, in implementing semantic technologies, in using Eclipse, etc. A healthy direction is to semi-automatically check what functionality can be used for what design operation. This can require different approaches:
   a. a way to classify tools and functionalities
   b. to obtain a perspective on what pieces of software within a tool can support what functionality
   c. to eventually be able to access tools at a method-API level

   Goal (a) can be achieved with an extension of codolight called designaspects.owl,[8] which implements some (customizable) axioms that allow a DL reasoner to automatically classify a tool (or a functionality) within an aspect, mostly based on input/output knowledge types. An example of an axiom is the following (in N3 encoding):

```
:ReuseReengineeringTool
      a       owl:Class ;
      rdfs:label "Reuse or reengineering tool {@en}"^^xsd:string ;
      rdfs:subClassOf codkernel:DesignTool ;
      owl:equivalentClass
              [ a       owl:Class ;
                owl:intersectionOf (codkernel:DesignTool [ a       owl:Restriction ;
```

[8] http://www.ontologydesignpatterns.org/cpont/codo/designaspects.owl

```
                                owl:onProperty codtools:hasOutputType ;
                                owl:someValuesFrom
                                        [ a       owl:Class ;
                                          owl:oneOf        (coddata:NetworkedOntologyKType
coddata:NetworkOfOntologiesKType                                  coddata:OntologyAxiomKType
coddata:OntologyElementKType    coddata:OntologyKType    coddata:OntologyMappingKType
coddata:OntologyModuleKType)
                                        ]
                                ])
                ] .
```

In practice, each owl:equivalentClass axiom establishes some knowledge types that, if in input and/or output, make a tool classifiable in a certain way. The variety of design aspects is of course open to customization: the current one follows understandability considerations and personal experience in administering ontology design tutorials.
However, given the axioms and e.g. the plugin models, a DL reasoner like Pellet 2 can infer that e.g. CiceroWiki and COAT are tools for reuse or reengineering knowledge resources.

(3) Given (2), a smart tool can try and produce a custom configuration of an ontology design toolkit, provided that some behavioral access to plugins is allowed and they are arranged in a way meaningful to interaction patterns.

(4) From the social viewpoint, codolight can be used to gather requirements, which can more easily matched against existing functionalities.

(5) From the semantic web viewpoint, codolight has been aligned to several vocabularies, as reported in [NeOnD2.1.2]: OWL, OMV (Ontology Metadata Vocabulary), DOAP (Description Of A Project), Protégé Workflow ontology, NeOn Access Rights ontology, SOM (Software Ontology Model), Sweet Tools, and the NeOn Trust ontology. This means that the data available in those vocabularies (e.g., Sweet Tools has almost 800 semantic web tools described), and the functionalities available with them (SOM is used with the OWL version of the MIT Process Handbook) can be in principle reused more easily, and more easily links can be established. This is also true for NTK plugins, where commonalities can be found, e.g. in terms of the knowledge types in input or output.

5.2 Beyond the logic-driven approach to modelling ontologies

One of the benefits of providing a model for collaborative ontology design, such as the C-ODO Light ontology, is to allow for a data-driven and design-oriented presentation of functionalities available in an ontology engineering environment. As with dataflow computer architectures, where the execution of instructions exclusively depends on the availability of their input arguments, in a design-oriented environment it is no longer the logical language but the actual bulk of data presented by design tools in the system that governs the execution of engineering tasks. The model for ontology design then becomes a backbone for mutual integration of the functionalities exposed by design tools, with particular attention on supporting collaboration-oriented tasks.

From this angle, the standard integration mechanisms provided by the Eclipse platform may not suffice. The most common criterion for classifying NeOn Toolkit (NTK) plug-ins in a way that is accessible to end-users is to group their views[9], where available, into custom categories. Apart from that, actual design operations are mostly performed with respect to the language, in that users directly interact with items that are a metaphor of OWL or F-Logic elements. As a matter of fact, users may not even know certain functionality is available unless an access point to its implementing plug-in is provided in the form of an item within the user interface.

[9] As in Eclipse Workbench views, see Eclipse Rich Client Platform (RCP).

Most NeOn Toolkit plug-ins, as well as ontology design tools of any other kind, can help overcome this hurdle by providing a reference to a model describing them using the C-ODO Light ontology. One of the means to leverage these descriptions will be provided by a NeOn Toolkit plug-in, hereinafter referred to as Kali-ma.

The Kali-ma plug-in will equip the NeOn Toolkit with a design-oriented, rather than language-oriented user interface. This aims at facilitating the integration of existing NeOn Toolkit plug-ins in anticipation of their use in a collaborative context, for which Kali-ma can provide an environment. The plug-in will be able to classify design tools with regard to those aspects of ontology design that they are known to cover. These aspects are generic functionalities of ontology design management, including use of design patterns, reuse and reengineering of existing modules, project management, workflow management, discussion and evaluation of the end product. Note that, given the nature of C-ODO Light, the notion of a NTK plug-in disappears in this context. Everything that implements functionality or supports certain design activities is a design tool, regardless of its concrete implementation.

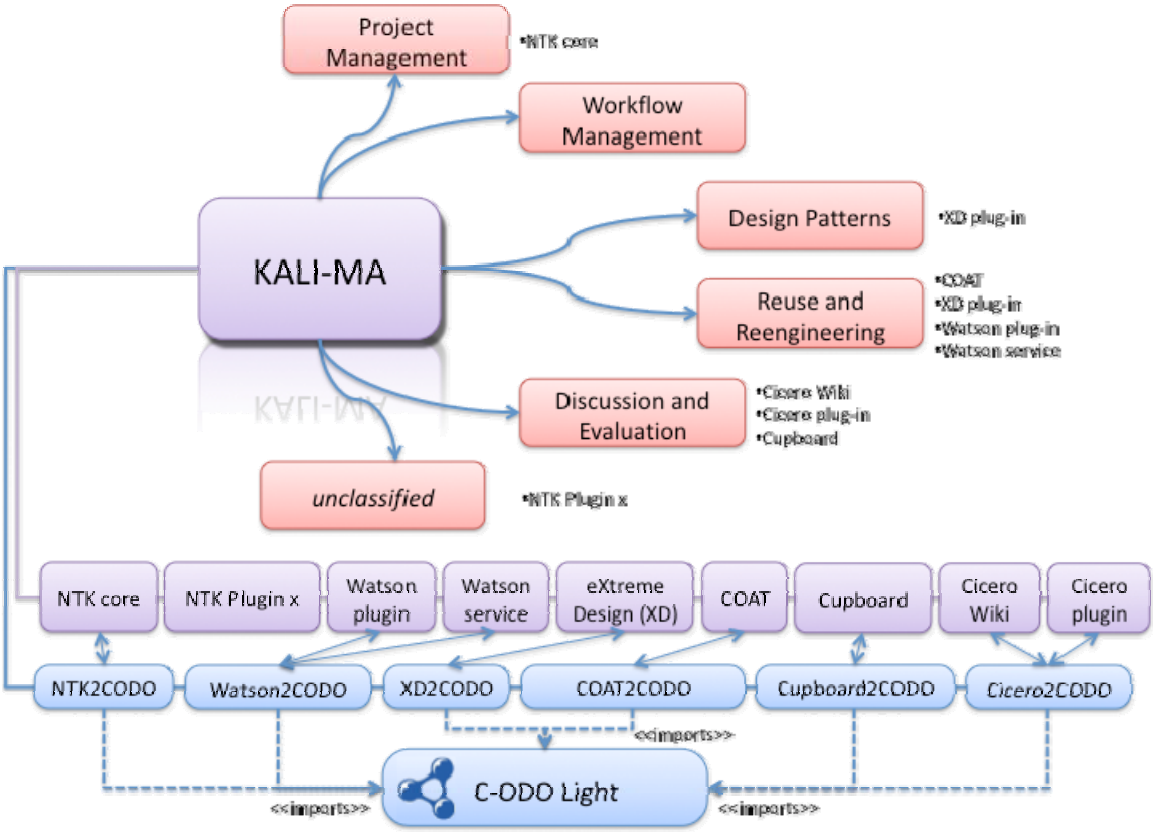

Figure 36: C-ODO Light-based organization of design tools as performed by Kali-ma.

Figure 36 depicts an example scenario where ontology design tools are automatically presented by Kali-ma in an integrated view according to their formal descriptions. Each description is a light ontology stating how a tool relates to design aspects exposed by the C-ODO Light ontology, which is a dependency of every single plug-in description. These design aspects are not hardcoded into Kali-ma, as it is entirely dependent on C-ODO Light, and therefore Kali-ma is unaware of such definitions until runtime. The user is then partially relieved from the tiresome task of figuring out what interface controls are used to trigger the execution of a tool, e.g. an Eclipse perspective, a wizard or a context menu item.

In addition to grouping known design tools with respect to these criteria, Kali-ma organizes all functionalities as widgets populating its interface, or dashboard. A widget is generated for each tool

that conforms to the Kali-ma specifications, which are described below. These widgets provide an access point to the tools and, at a later stage, will act as proxies for them, meaning that users will be able to use some of the functionalities provided by a tool through the widget rather than the tool itself. For instance, the results of a Watson keyword-based query will be displayed in a lightweight form directly on the widget for the Watson plug-in, and if further user input is required, e.g. adding a relation between the target ontology and an entry in the Watson results, appropriate controls can be displayed on the same widget.

5.3 Benefits of C-ODO Light-based plug-in descriptions

Although the Kali-ma plug-in ultimately pursues other goals, most prominently the enhancement of user interaction with the NTK, exploiting the conceptual integration offered by C-ODO Light remains a key task. As it follows an iterative design process, the plug-in is evolving towards an alternative interaction approach that may involve any set of NTK functionalities. In order for these functionalities to be integrated, the key requirement is that their role in an ontology design process must be clearly stated, no matter how intrinsically collaboration-oriented they are. That is to say, any tool that is described by means of C-ODO Light can be plugged into the model provided by Kali-ma, which can be thought of as an object model for a subset of C-ODO elements. If a tool fails to provide a description that links it with some design aspect, it will be grouped with other “unclassified” or “miscellaneous” tools.

One effect of defining the tasks and operations involved with a particular tool is that Kali-ma can aggregate them with respect to available functionalities, and subsequently organize them into sets for user convenience, called profiles. From a collaborative standpoint, profile management is being designed so that the profiles generated by Kali-ma can be shared across ontology development projects and instances of the NeOn Toolkit.

Another advantage can be derived from the C-ODO Light module for the description of user interfaces, i.e. the *codinterfaces* ontology. Plug-in providers can utilize this module to describe interface components for their tools. Instances for this module can describe a standard interface for a plug-in as well as its appearance as a Kali-ma dashboard widget. Also, a third-party plug-in is given a means to share a formal definition of the types of resources that are involved in the task(s) it was designed for. A description based on the C-ODO kernel[10] (see NeOn deliverable 2.1.2: “Model for collaborative design of networked ontologies” for a detailed description) allows a tool to define the *codkernel:KnowledgeResource*(s) that are reused by workflows and projects involving the tool. A particular *KnowledgeResource*, or type thereof, can be declared as serving as input or output for anything that may use it, i.e. tasks, design operations, design tools or workflows. With such knowledge, a Kali-ma widget representing a tool could trigger a particular operation when an instance of its expected input type is presented to the system through another widget.

We are considering that description elements not strictly related to interaction might nonetheless have an impact on the widget interface. To clarify with an example, let us consider a tool such as Cicero. This tool describes a workflow associated to it as a sequence of operations to be executed, or the tasks realized by them, in a given order. If such a workflow can be executed by using this tool, then its widget could display these operations or tasks as elements in a tabbed pane and grey out those tabs representing operations that cannot be performed until their pre-requisites are met.

[10] http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl

5.4 Operational requirements

The examples related to C-ODO Light descriptions presented so far show some advantages offered by Kali-ma with regard to organizing and presenting NeOn Toolkit plug-ins. However, they offer little help when it comes to accessing plug-in functionalities operationally, i.e. by executing the procedures that implement them. Although the organization of functionalities by design principles does not depend on this operational aspect, it is useful to give an insight as to how the issue will be addressed in subsequent versions of Kali-ma.

It would not be wise to enlarge the C-ODO Light model with excessive technicalities beyond its scope, such as Eclipse framework components or Java language constructs, only for allowing Kali-ma to locate the methods to be called and the parameters to be passed on to them. It has therefore been established that most operational requirements will have to be satisfied by enriching plug-in manifests (i.e. *plugin.xml* files provided by all Eclipse user interface plug-ins) and integrating additional Java classes in the plug-in code. This approach follows Eclipse plug-in development standards, hence it should entail a minimum of programming effort from plug-in developers.

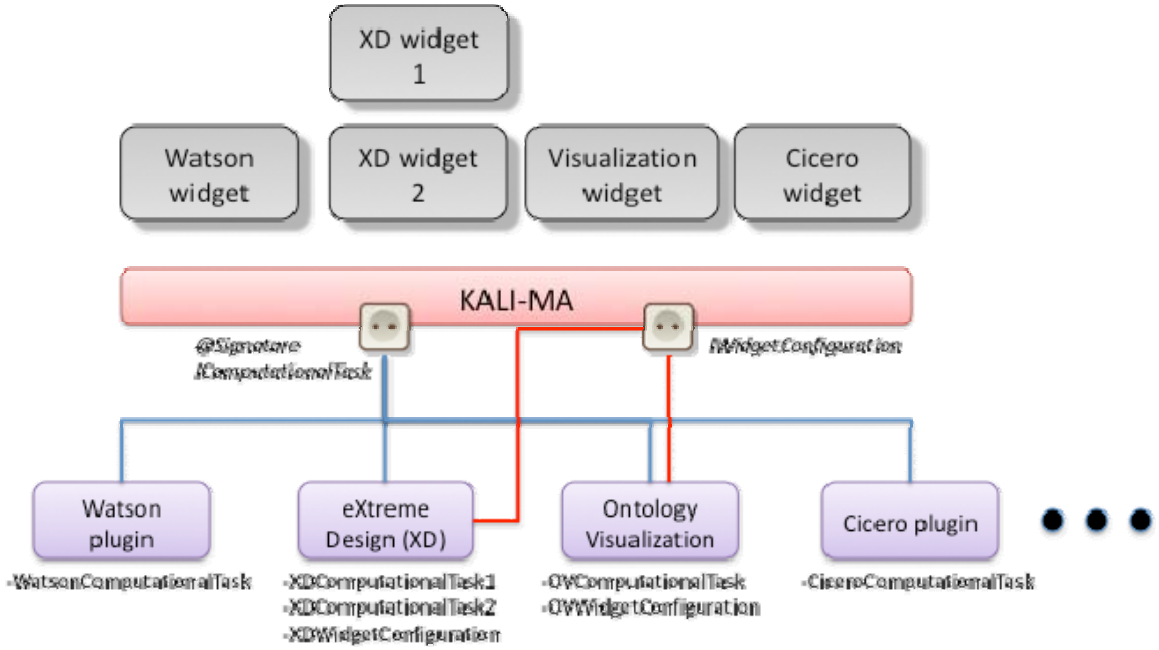

Figure 37: Usage of extension points provided by Kali-ma for the generation of plug-in widgets.

An outline of the extension mechanisms provided by Kali-ma for the generation of dashboard widgets is provided in fig.37. Kali-ma exposes its own Eclipse extension point (leftmost socket), currently called *computationalTask* after the C-ODO Light class it is based on. This extension point is associated with the Java interface *IComputationalTask*, consisting of a single *execute()* method having generic Objects as types for its input parameters and return value. Developers wishing to allow Kali-ma to execute a task provided by their plug-ins will have to provide an extension to the *computationalTask* extension point and an implementation of the *IComputationalTask* interface. The *execute()* method will have to be implemented so as to perform a task the plug-in has been designed for, and for which the related Java code is assumed to already exist. Additionally, plug-in developers may wish to override default widget generation heuristics by providing custom controls or settings through the *IWidgetConfiguration* interface (rightmost socket), although this will obviously imply a greater creative and programming effort. Multiple *computationalTask* extensions may be provided so as to generate multiple widgets or a group.

A NeOn Toolkit plug-in developer will also be required to specify the actual types of the parameters expected by its computational task, as well as the value returned, if any. This can be achieved by annotating its *execute()* method implementation with a Java 1.5 annotation called *Signature*, also provided by Kali-ma. There, developers can specify Java classes for the returned value and as many arguments as required. Kali-ma is then able to use Java reflection in order to provide such typed arguments. Obviously, Kali-ma will be able to instantiate solely those types available in its own classpath, therefore only a restricted set of types will be allowed, including the OWL API datamodel and all the Java 5 SE classes.

5.5 Interoperability Support

The notion of design tool as it is described in C-ODO Light is not restricted to NeOn Toolkit plug-ins. As a matter of fact, it encompasses any piece of software that can implement functionality, cover an aspect, realize a workflow or contribute in any way to an ontology engineering process, no matter whether it is a standalone application, a plug-in or a web application. Kali-ma is intended to exploit this generalization provided by C-ODO Light.

In addition to the descriptions provided by individual NeOn Toolkit plug-ins, Kali-ma may be able to classify any set of C-ODO Light-compliant tools and present their descriptions to the user. In cases where these tools are not available as NTK plug-ins, Kali-ma is still able to present them to the user and provide a method for obtaining them, e.g. the corresponding project page or download page on the Web. This means that Kali-ma can provide an overview of a broad span of ontology development tools that are known to contribute to some design aspect, even when they have been developed to run within a framework that has no connection with Eclipse or the NeOn Toolkit. In turn, such a view can be stored and shared for reuse among other collaboration-oriented tools, including multiple running instances of Kali-ma. As an example, suffice it to say that C-ODO Light is aligned with the Collaborative Protégé ontology, therefore Kali-ma is likely to evolve to be interoperable with the Protégé toolkit and its collaborative features.

6. Conclusions and future work

In this deliverable, we have provided descriptions of the functionality of a number of NeOn Toolkit plug-ins in terms of both informal characterization and the C-ODO Light ontology, which enables the integration of tools in a collaborative workflow.

Further, we have sketched a picture of an upcoming NeOn Toolkit plug-in that is able to present a design-driven uniform view on ontology design tools and, in doing so, depends on a formal semantic model that is the C-ODO Light ontology and the design tool descriptions based thereon. This plug-in will be able to help users overcome the problem of locating available functionalities within the NTK framework. It will also offer users a design-centred interaction approach, as opposed to the traditional approach that relies on the language used for representing knowledge. Users will be able to customize their interaction experience, as well as share their environment settings in collaborative contexts.

The Kali-ma features so far exposed lay but the first stepping-stones towards an array of future development scenarios concerning both collaboration and interaction support. Organizing and presenting design tools that can be related to C-ODO Light is merely a starting point. For one step further, once a tool can be described as a workflow or part of one, users could be given the choice to assemble the toolchain that realizes this workflow on the basis of specific tasks or operations. If a user can reasonably argue that a third-party tool is able to realize a design task better than the one provided by the tool currently used, they may want to replace specific functionalities in the workflow as if they were standalone service bundles. This desirable feature shares several points, yet it is different from workflow realization support. The latter is another feature that has been set as a goal of future Kali-ma releases, and it is the capability of arranging NeOn Toolkit plug-in widgets so as to assemble a step-by-step supply chain.

References

[1] Buitelaar, P., Cimiano, P., Magnini, B. (2005). Ontology Learning from Text: Methods, Applications and Evaluation, Frontiers in Artificial Intelligence and Applications, IOS Press.

[2] Grobelnik, M., Mladenic, D. (2006). Knowledge discovery for ontology construction. In: Davies, J., Studer, R., Warren, P., (eds.) Semantic web technologies: trends and research in ontology-based systems. Chichester: John Wiley & Sons, cop. pp. 9-27, 2006.

[3] Maedche, A. and Staab, S. (2004). Ontology Learning. In Staab, S. and Studer, R., editors Handbook on Ontologies. International Handbooks on Information Systems. Springer-Verlag., pages 173 – 190.

[4] Buitelaar, P., Olejnik, D., and Sintek, M. (2004b). A Protege Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. In Bussler, C., Davies, J., Fensel, D., and Studer, R., editors, Proceedings of the First European Semantic Web Symposium (ESWS2004), volume 3053 of LNCS. Springer-Verlag, Heraklion, Crete, Greece.

[5] Cimiano, P. and Voelker, J. (2005). Text2Onto - A Framework for Ontology Learning and Data driven Change Discovery. In Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'2005).

[6] Gomez-Perez, A. and Manzano-Mancho, D. (2003). A Survey of Ontology Learning Methods and Techniques. OntoWeb Delieverable 1.5.

[7] Fortuna, B., Mladenic, D., Grobelnik, M. (2006). Semi-automatic construction of topic ontologies, In Knowledge Discovery and Ontologies, Berendt et al. (eds), Springer Lecture Notes.

[8] Mladenic, D., Grobelnik, M. Mapping documents onto web page ontology. In Berendt et al. (eds.), Web mining : from web to semantic web, (Lecture notes in artificial inteligence, Lecture notes in computer science, vol. 3209). Berlin; Heidelberg; New York: Springer, 2004, 77-96.

[9] Soumen Chakrabarti, Mining the Web: Discovering Knowledge from Hypertext Data,Morgan-Kaufmann Publishers, 2002

[10] Matt Richardson, Pedro Domingos, Combining Link and Content Information in Web Search, In M. Levene and A. Poulovassilis (eds.), Web Dynamics (pp. 179-193), 2004. New York: Springer.

[11] Stanley Wasserman, Katherine Faust, Social Network Analysis: Methods and Applications, United Kingdom: Cambridge University Press, 1994

[12] Aaron Clauset, M. E. J. Newman, and Cristopher Moore: Finding community structure in very large networks, Phys. Rev. E 70, 066111 (2004)

[13] M. Girvan and M. E. J. Newman, Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002).

[14] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, Defining and identifying communities in networks. Preprint cond-mat/0309488 (2003)

[15] A. Pothen, H. Simon, and K.-P. Liou, Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Matrix Anal. Appl. 11, 430-452 (1990).

[16] S. Sabo, M. Grčar, D. A. Fabjan, P. Ljubi, N. Lavrač, Exploratory analysis of the ILPNet2 social network, In Proceedings of the 10th International multi-conference Information society, SiKDD2007 (2007).

[17] V. Batagelj, A. Mrvar Pajek: Program for Analysis and Evaluation of Large Networks. 2003.

[18] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002

[19] R. Grishman. 1997. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA. http://www.itl.nist.gov/-div894/894.02/related projects/tipster/.

[20] S. Bird, D. Day, J. Garofolo, J. Henderson, C. Laprun, and M. Liberman. 2000. ATLAS: A flexible and extensible architecture for linguistic annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens.

[21] H. Thompson and D. McKelvie. 1997, semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe'97*, Barcelona.

[22] Presutti V., Gangemi A. (2008). Content Ontology Design Patterns as practical building blocks for web ontologies. In *Proceedings of ER2008* Barcelona, Spain.

[23] Gangemi, A., Presutti, V. (2007). C-ODO: an OWL meta-model for collaborative ontology design. In *Proceedings of CKC Workshop at WWW2007, Banff, Canada* Berlin, Springer.

[24] Raúl Palma, Peter Haase: Oyster - Sharing and Re-using Ontologies in a Peer-to-Peer Community. International Semantic Web Conference 2005:1059-1062

Appendix

This appendix contains the alignment models between the tools described in this deliverable and the C-ODO Light framework. The ontologies are in N3 format.

The url of the owl versions is:
http://www.ontologydesignpatterns.org/cpont/codo/allcodoalignments.owl

Editorial Workflow

```
# baseURI:
http://www.ontologydesignpatterns.org/cpont/codo/editorialworkflow2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl
# imports: http://www.ontologydesignpatterns.org/cp/owl/controlflow.owl

@prefix taskrole:  <http://www.ontologydesignpatterns.org/cp/owl/taskrole.owl#>
.
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix precedence:
<http://www.ontologydesignpatterns.org/cp/owl/precedence.owl#> .
@prefix intensionextension:
<http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sequence:  <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl#>
.
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix codinteraction:
<http://www.ontologydesignpatterns.org/cpont/codo/codinteraction.owl#> .
@prefix controlflow:
<http://www.ontologydesignpatterns.org/cp/owl/controlflow.owl#> .
@prefix partof:  <http://www.ontologydesignpatterns.org/cp/owl/partof.owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/editorialworkflow2codo.owl#> .

:TransformOntologyAxiom
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:comment "Every change in the editor is transformed into instances of
the change representation model (Change Ontology)." ;
      rdfs:label "Transform ontology axiom"@en ;
      partof:isPartOf :ChangeLogging ;
      sequence:directlyFollows
              :CaptureOntologyAxiom ;
      sequence:directlyPrecedes
              :RegisterOntologyAxiom ;
      sequence:follows :CaptureOntologyAxiom ;
      sequence:precedes :RegisterOntologyAxiom ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:DecideIfNewUser_No , :ChangeLogging , :DecideIfUserIsAllowed_Yes ,
:VisualizeChanges , :RegisterOntologyAxiom , :DecideIfUserIsAllowed ,
:DecideIfTaskRequiresOntologyChange_No , :CaptureOntologyAxiom ,
:DecideIfNewUser_Yes , :ChangeOntology , :Register , :DecideIfUserIsAllowed_No ,
```

```
:WorkflowVisualization , :LogIn , :SynchronizeChanges ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:DecideIfNewUser_No
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if new user_No"@en ;
      partof:isPartOf :UserIdentification ;
      sequence:directlyFollows
              :DecideIfNewUser ;
      sequence:directlyPrecedes
              :LogIn ;
      sequence:follows :DecideIfNewUser ;
      sequence:precedes :LogIn ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :VisualizeChanges , :ChangeLogging ,
:RegisterOntologyAxiom , :DecideIfUserIsAllowed , :CaptureOntologyAxiom ,
:DecideIfNewUser_Yes , :ChangeOntology , :Register , :SynchronizeChanges ,
:WorkflowVisualization , :LogIn , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser .

:VisualizeChanges
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:comment "Visualize the ontologies that are being logged and for each
of them, the history of changes sorted in chronological order." ;
      rdfs:label "Visualize changes"@en ;
      codtools:hasInputType
              :OntologyIDKType ;
      codtools:hasOutputType
              :OntologyChangeListKType ;
      codtools:isImplementedIn
              :ChangeManagementTool ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom , :DecideIfUserIsAllowed ,
:DecideIfTaskRequiresOntologyChange_No , :CaptureOntologyAxiom ,
:DecideIfNewUser_Yes , :ChangeOntology , :Register , :DecideIfUserIsAllowed_No ,
:WorkflowVisualization , :SynchronizeChanges , :LogIn ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:ChangeLogging
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:comment "Capture ontology changes from the NTK editor and log them
into Oyster distributed registry" ;
      rdfs:label "Change logging"@en ;
      partof:hasPart :CaptureOntologyAxiom , :TransformOntologyAxiom ,
:RegisterOntologyAxiom ;
      codtools:hasInputType
              :OntologyAxiomListKType ;
      codtools:isImplementedIn
              :ChangeManagementTool ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :VisualizeChanges ,
:DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :WorkflowVisualization ,
```

```
:SynchronizeChanges , :LogIn , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser ,
:DecideIfTaskRequiresOntologyChange_Yes .

:RegisterOntologyAxiom
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:comment "The instances of the change ontology are registered into
Oyster distributed registry" ;
      rdfs:label "Register ontology axiom"@en ;
      partof:isPartOf :ChangeLogging ;
      sequence:directlyFollows
              :TransformOntologyAxiom ;
      sequence:follows :CaptureOntologyAxiom , :TransformOntologyAxiom ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :DecideIfUserIsAllowed_Yes ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology , :Register ,
:DecideIfUserIsAllowed_No , :SynchronizeChanges , :LogIn ,
:WorkflowVisualization , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser ,
:DecideIfTaskRequiresOntologyChange_Yes .

:UserInformationKType
      a       codkernel:KnowledgeType ;
      rdfs:label "User information KType"@en ;
      codtools:isInputTypeFor
              :WorkflowVisualization , :WorkflowManagement ,
:WorkflowSupportTool , :UserIdentification .

:CaptureOntologyAxiom
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:comment "Every change from the NTK editor is captured" ;
      rdfs:label "Capture ontology axiom"@en ;
      partof:isPartOf :ChangeLogging ;
      sequence:directlyPrecedes
              :TransformOntologyAxiom ;
      sequence:precedes :TransformOntologyAxiom , :RegisterOntologyAxiom ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :DecideIfUserIsAllowed_Yes ,
:VisualizeChanges , :ChangeLogging , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:DecideIfNewUser_Yes , :ChangeOntology , :Register , :DecideIfUserIsAllowed_No ,
:LogIn , :WorkflowVisualization , :SynchronizeChanges ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:OntologyChangeInformation
      a       owl:Class ;
      rdfs:label "Ontology change information"@en ;
      rdfs:subClassOf :OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyChangeInformationKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:WorkflowManagement
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
```

```
      rdfs:comment """This task is executed when manipulating ontology changes
according to the role of a user and the workflow requirements.
In particular, it is composed of the following tasks:
,Ä¢   Verifies that the user is allowed to perform the requested workflow
action.
,Ä¢   Evaluates if the requested workflow action requires performing a change in
the corresponding ontology.
,Ä¢   Registers the workflow action""" ;
      rdfs:label "Workflow management"@en ;
      partof:hasPart :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:ChangeOntology , :DecideIfUserIsAllowed_No ,
:DecideIfTaskRequiresOntologyChange , :DecideIfUserIsAllowed_Yes ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:DecideIfTaskRequiresOntologyChange_Yes ;
      codtools:hasInputType
              :UserInformationKType ;
      codtools:isImplementedIn
              :WorkflowSupportTool ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :DecideIfUserIsAllowed_Yes ,
:ChangeLogging , :VisualizeChanges , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :SynchronizeChanges , :LogIn ,
:WorkflowVisualization , :DecideIfTaskRequiresOntologyChange ,
:UserIdentification , :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes
.

:OntologyRelatedData
      a       owl:Class ;
      rdfs:label "Ontology related data"@en ;
      rdfs:subClassOf coddata:Annotation ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyRelatedDataKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:UserRole
      a       owl:Class ;
      rdfs:label "User role"@en ;
      rdfs:subClassOf codkernel:UserType .

:DecideIfTaskRequiresOntologyChange_Yes
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if task requires ontology change_Yes"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfTaskRequiresOntologyChange ;
      sequence:directlyPrecedes
              :ChangeOntology ;
      sequence:follows :DecideIfTaskRequiresOntologyChange ,
:DecideIfUserIsAllowed_Yes , :DecideIfUserIsAllowed ;
      sequence:precedes :ChangeOntology ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :VisualizeChanges , :ChangeLogging ,
:RegisterOntologyAxiom , :DecideIfTaskRequiresOntologyChange_No ,
:DecideIfUserIsAllowed , :CaptureOntologyAxiom , :ChangeOntology , :Register ,
:WorkflowVisualization , :LogIn , :SynchronizeChanges ,
```

```
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser .

:DeliverErrorMessageToUser
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:label "Deliver error message to user"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfUserIsAllowed_No ;
      sequence:follows :DecideIfUserIsAllowed_No , :DecideIfUserIsAllowed ;
      owl:differentFrom :RegisterWorkflowAction , :TransformOntologyAxiom ,
:DecideIfNewUser_No , :DecideIfUserIsAllowed_Yes , :ChangeLogging ,
:VisualizeChanges , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology , :Register ,
:DecideIfUserIsAllowed_No , :LogIn , :WorkflowVisualization ,
:SynchronizeChanges , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement
, :UserIdentification , :DecideIfNewUser ,
:DecideIfTaskRequiresOntologyChange_Yes .

:OntologyAxiomList
      a       owl:Class ;
      rdfs:label "Ontology axiom list"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl#Collection> ,
:OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:allValuesFrom coddata:OntologyAxiom ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl#hasMember>
              ] .

:WorkflowSupportTool
      a       codkernel:DesignTool ;
      rdfs:label "Workflow support tool"@en ;
      codtools:hasInputType
              :UserInformationKType , :WorkflowStateChangeInformationKType ;
      codtools:implements :ChangeOntology , :WorkflowVisualization ,
:WorkflowManagement , :UserIdentification .

:DecideIfUserIsAllowed
      a       controlflow:BooleanCaseTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if user is allowed"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyPrecedes
              :DecideIfUserIsAllowed_No , :DecideIfUserIsAllowed_Yes ;
      sequence:precedes :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:ChangeOntology , :DecideIfUserIsAllowed_No ,
:DecideIfTaskRequiresOntologyChange , :DecideIfUserIsAllowed_Yes ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfTaskRequiresOntologyChange_Yes
;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :DecideIfUserIsAllowed_Yes ,
:VisualizeChanges , :ChangeLogging , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :CaptureOntologyAxiom ,
:DecideIfNewUser_Yes , :ChangeOntology , :DecideIfUserIsAllowed_No , :Register ,
:SynchronizeChanges , :WorkflowVisualization , :LogIn ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .
```

```
:DecideIfTaskRequiresOntologyChange_No
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if task requires ontology change_No"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfTaskRequiresOntologyChange ;
      sequence:directlyPrecedes
              :RegisterWorkflowAction ;
      sequence:follows :DecideIfTaskRequiresOntologyChange ,
:DecideIfUserIsAllowed_Yes , :DecideIfUserIsAllowed ;
      sequence:precedes :RegisterWorkflowAction ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :VisualizeChanges , :ChangeLogging ,
:RegisterOntologyAxiom , :DecideIfUserIsAllowed , :CaptureOntologyAxiom ,
:ChangeOntology , :Register , :WorkflowVisualization , :SynchronizeChanges ,
:LogIn , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement ,
:UserIdentification , :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes
.

:OntologyAxiomListKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology axiom list KType"@en ;
      codtools:isInputTypeFor
              :ChangeLogging , :ChangeManagementTool .

:LogIn
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:label "Log in"@en ;
      partof:isPartOf :UserIdentification ;
      sequence:directlyFollows
              :DecideIfNewUser_No ;
      sequence:follows :DecideIfNewUser_No , :DecideIfNewUser ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :SynchronizeChanges ,
:WorkflowVisualization , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser ,
:DecideIfTaskRequiresOntologyChange_Yes .

:OntologyIDKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology IDKType"@en ;
      codtools:isInputTypeFor
              :VisualizeChanges , :ChangeManagementTool .

:OntologyID
      a       owl:Class ;
      rdfs:label "Ontology ID"@en ;
      rdfs:subClassOf :OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyIDKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .
```

```
:UserIdentification
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      a       [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/description.owl#isConceptUsedIn> ;
                owl:someValuesFrom
<http://www.ontologydesignpatterns.org/cpont/codo/codworkflows.owl#Collaborative
Workflow>
              ] ;
      rdfs:comment "This task is executed when identifying a user to the system
and logging-in. The user information includes user firstname, lastname, and user
role." ;
      rdfs:label "User identification"@en ;
      partof:hasPart :DecideIfNewUser_Yes , :Register , :DecideIfNewUser_No ,
:LogIn , :DecideIfNewUser ;
      codtools:hasInputType
              :UserInformationKType ;
      codtools:isImplementedIn
              :WorkflowSupportTool ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfUserIsAllowed , :DecideIfTaskRequiresOntologyChange_No ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :LogIn , :WorkflowVisualization ,
:SynchronizeChanges , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:DecideIfNewUser
      a       controlflow:BooleanCaseTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if new user"@en ;
      partof:isPartOf :UserIdentification ;
      sequence:directlyPrecedes
              :DecideIfNewUser_Yes , :DecideIfNewUser_No ;
      sequence:precedes :DecideIfNewUser_Yes , :Register , :DecideIfNewUser_No ,
:LogIn ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfUserIsAllowed , :DecideIfTaskRequiresOntologyChange_No ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :LogIn , :WorkflowVisualization ,
:SynchronizeChanges , :WorkflowManagement , :UserIdentification ,
:DecideIfTaskRequiresOntologyChange_Yes .

:WorkflowStateChangeInformationKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Workflow state change information KType"@en ;
      codtools:isInputTypeFor
              :WorkflowVisualization , :WorkflowSupportTool .

:DecideIfNewUser_Yes
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if new user_Yes"@en ;
      partof:isPartOf :UserIdentification ;
      sequence:directlyFollows
              :DecideIfNewUser ;
      sequence:directlyPrecedes
              :Register ;
```

```
      sequence:follows :DecideIfNewUser ;
      sequence:precedes :Register ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :RegisterOntologyAxiom , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :ChangeOntology , :Register , :SynchronizeChanges ,
:LogIn , :WorkflowVisualization , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser .

:ChangeOntology
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:label "Change ontology"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfTaskRequiresOntologyChange_Yes ;
      sequence:follows :DecideIfTaskRequiresOntologyChange ,
:DecideIfUserIsAllowed_Yes , :DecideIfUserIsAllowed ,
:DecideIfTaskRequiresOntologyChange_Yes ;
      codtools:isImplementedIn
              :WorkflowSupportTool ;
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :DecideIfUserIsAllowed_Yes ,
:VisualizeChanges , :ChangeLogging , :RegisterOntologyAxiom ,
:DecideIfUserIsAllowed , :DecideIfTaskRequiresOntologyChange_No ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :Register ,
:DecideIfUserIsAllowed_No , :SynchronizeChanges , :WorkflowVisualization ,
:LogIn , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement ,
:UserIdentification , :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes
.

:DecideIfUserIsAllowed_No
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if user is allowed_No"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfUserIsAllowed ;
      sequence:directlyPrecedes
              :DeliverErrorMessageToUser ;
      sequence:follows :DecideIfUserIsAllowed ;
      sequence:precedes :DeliverErrorMessageToUser ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :VisualizeChanges , :DecideIfUserIsAllowed_Yes ,
:ChangeLogging , :RegisterOntologyAxiom , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :ChangeOntology , :Register , :SynchronizeChanges ,
:WorkflowVisualization , :LogIn , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser .

:WorkflowVisualization
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:comment "This task is executed when visualizing changes within the
workflow that are in a particular state." ;
      rdfs:label "Workflow visualization"@en ;
      codtools:hasInputType
              :UserInformationKType , :WorkflowStateChangeInformationKType ;
      codtools:hasOutputType
              :OntologyChangeInformationKType ;
      codtools:isImplementedIn
              :WorkflowSupportTool ;
```

```
      owl:differentFrom :RegisterWorkflowAction , :DeliverErrorMessageToUser ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :VisualizeChanges ,
:DecideIfUserIsAllowed_Yes , :ChangeLogging , :RegisterOntologyAxiom ,
:DecideIfUserIsAllowed , :DecideIfTaskRequiresOntologyChange_No ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology , :Register ,
:DecideIfUserIsAllowed_No , :LogIn , :SynchronizeChanges ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:SynchronizeChanges
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:comment "Start Oyster synchronization process in the distributed
environment and apply changes received from other clients to the same
ontologies." ;
      rdfs:label "Synchronize changes"@en ;
      codtools:isImplementedIn
              :ChangeManagementTool ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :ChangeLogging ,
:VisualizeChanges , :DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology , :Register ,
:DecideIfUserIsAllowed_No , :LogIn , :WorkflowVisualization ,
:DecideIfTaskRequiresOntologyChange , :WorkflowManagement , :UserIdentification
, :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes .

:WorkflowStateChangeInformation
      a       owl:Class ;
      rdfs:label "Workflow state change information"@en ;
      rdfs:subClassOf :OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :WorkflowStateChangeInformationKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:RegisterWorkflowAction
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:label "Register workflow action"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfTaskRequiresOntologyChange_No ;
      sequence:follows :DecideIfTaskRequiresOntologyChange ,
:DecideIfUserIsAllowed_Yes , :DecideIfTaskRequiresOntologyChange_No ,
:DecideIfUserIsAllowed ;
      owl:differentFrom :DeliverErrorMessageToUser , :TransformOntologyAxiom ,
:DecideIfNewUser_No , :ChangeLogging , :DecideIfUserIsAllowed_Yes ,
:VisualizeChanges , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :LogIn , :WorkflowVisualization ,
:SynchronizeChanges , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement
, :UserIdentification , :DecideIfNewUser ,
:DecideIfTaskRequiresOntologyChange_Yes .

:OntologyChangeInformationKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology change information KType"@en ;
      codtools:isOutputTypeFor
```

```
              :WorkflowVisualization .

:DecideIfUserIsAllowed_Yes
      a       controlflow:DeliberationTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if user is allowed_Yes"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfUserIsAllowed ;
      sequence:directlyPrecedes
              :DecideIfTaskRequiresOntologyChange ;
      sequence:follows :DecideIfUserIsAllowed ;
      sequence:precedes :RegisterWorkflowAction , :ChangeOntology ,
:DecideIfTaskRequiresOntologyChange , :DecideIfTaskRequiresOntologyChange_No ,
:DecideIfTaskRequiresOntologyChange_Yes ;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :VisualizeChanges , :ChangeLogging ,
:RegisterOntologyAxiom , :DecideIfUserIsAllowed , :CaptureOntologyAxiom ,
:ChangeOntology , :DecideIfUserIsAllowed_No , :Register , :WorkflowVisualization
, :LogIn , :SynchronizeChanges , :DecideIfTaskRequiresOntologyChange ,
:WorkflowManagement , :UserIdentification , :DecideIfNewUser .

:OntologyChangeListKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology change list KType"@en ;
      codtools:isOutputTypeFor
              :VisualizeChanges .

:UserInformation
      a       owl:Class ;
      rdfs:comment "The user information includes user firstname, lastname, and
user role." ;
      rdfs:label "User information"@en ;
      rdfs:subClassOf :OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :UserInformationKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty partof:hasPart ;
                owl:someValuesFrom :UserRole
              ] .

<http://www.ontologydesignpatterns.org/cpont/codo/editorialworkflow2codo.owl>
      a       owl:Ontology ;
      owl:imports <http://www.ontologydesignpatterns.org/cp/owl/controlflow.owl>
, <http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ;
      owl:versionInfo """0.1: Derived by Aldo Gangemi from Raul Palma's
description of Workflow Management and Change Management tools.
0.2: added labels and knowledge types"""^^xsd:string .

:Register
      a       controlflow:ActionTask , codinteraction:ComputationalDesignTask ;
      rdfs:label "Register"@en ;
      partof:isPartOf :UserIdentification ;
      sequence:directlyFollows
              :DecideIfNewUser_Yes ;
      sequence:follows :DecideIfNewUser_Yes , :DecideIfNewUser ;
```

```
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :VisualizeChanges ,
:ChangeLogging , :DecideIfUserIsAllowed_Yes , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :SynchronizeChanges , :WorkflowVisualization ,
:LogIn , :DecideIfTaskRequiresOntologyChange , :WorkflowManagement ,
:UserIdentification , :DecideIfNewUser , :DecideIfTaskRequiresOntologyChange_Yes
.

:OntologyRelatedDataKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology related data KType"@en .

:DecideIfTaskRequiresOntologyChange
      a       controlflow:BooleanCaseTask ,
codinteraction:ComputationalDesignTask ;
      rdfs:label "Decide if task requires ontology change"@en ;
      partof:isPartOf :WorkflowManagement ;
      sequence:directlyFollows
              :DecideIfUserIsAllowed_Yes ;
      sequence:directlyPrecedes
              :DecideIfTaskRequiresOntologyChange_No ,
:DecideIfTaskRequiresOntologyChange_Yes ;
      sequence:follows :DecideIfUserIsAllowed_Yes , :DecideIfUserIsAllowed ;
      sequence:precedes :RegisterWorkflowAction , :ChangeOntology ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfTaskRequiresOntologyChange_Yes
;
      owl:differentFrom :DeliverErrorMessageToUser , :RegisterWorkflowAction ,
:TransformOntologyAxiom , :DecideIfNewUser_No , :VisualizeChanges ,
:DecideIfUserIsAllowed_Yes , :ChangeLogging , :RegisterOntologyAxiom ,
:DecideIfTaskRequiresOntologyChange_No , :DecideIfUserIsAllowed ,
:CaptureOntologyAxiom , :DecideIfNewUser_Yes , :ChangeOntology ,
:DecideIfUserIsAllowed_No , :Register , :WorkflowVisualization , :LogIn ,
:SynchronizeChanges , :WorkflowManagement , :UserIdentification ,
:DecideIfTaskRequiresOntologyChange_Yes .

:OntologyChangeList
      a       owl:Class ;
      rdfs:label "Ontology change list"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl#Collection> ,
:OntologyRelatedData ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:allValuesFrom :OntologyChangeInformation ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl#hasMember>
              ] .

:ChangeManagementTool
      a       codkernel:DesignTool ;
      rdfs:label "Change management tool"@en ;
      codtools:hasInputType
              :OntologyAxiomListKType , :OntologyIDKType ;
      codtools:implements :SynchronizeChanges , :VisualizeChanges ,
:ChangeLogging .
```

Open Rating System

```
# baseURI: http://www.ontologydesignpatterns.org/cpont/codo/tsors2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/odm2codo.owl

@prefix owl2xml:  <http://www.w3.org/2006/12/owl2-xml#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix intensionextension:
<http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix codinteraction:
<http://www.ontologydesignpatterns.org/cpont/codo/codinteraction.owl#> .
@prefix description:
<http://www.ontologydesignpatterns.org/cp/owl/description.owl#> .
@prefix agentrole:
<http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/tsors2codo.owl#> .

:ReviewRank
      a       owl:Class ;
      rdfs:label "Review rank"@en ;
      rdfs:subClassOf coddata:Annotation ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :ReviewRankKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :ReviewRanking ;
                owl:onProperty codtools:isOutputDataFor
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty intensionextension:isAbout ;
                owl:someValuesFrom :Review
              ] .

:TrustAnnotationKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Trust annotation KType"@en ;
      codtools:isInputTypeFor
              :TS-ORS , :ReviewRanking , :OntologyRanking .

:TrustAnnotation
      a       owl:Class ;
      rdfs:comment "Users express trust in other users." ;
      rdfs:label "Trust annotation"@en ;
```

```
      rdfs:subClassOf coddata:Annotation ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty intensionextension:isAbout ;
                owl:someValuesFrom agentrole:Agent
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/informationobjectsandrepresentatio
nlanguages.owl#isConceptualizedBy> ;
                owl:someValuesFrom agentrole:Agent
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :TrustAnnotationKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty intensionextension:isAbout ;
                owl:someValuesFrom :Review
              ] .

coddata:OntologyKType
      codtools:isInputTypeFor
              :TS-ORS , :OntologyRanking .

:OntologyRankKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology rank KType"@en ;
      codtools:isOutputTypeFor
              :TS-ORS , :OntologyRanking .

:ReviewKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Review KType"@en ;
      codtools:isInputTypeFor
              :TS-ORS , :ReviewRanking , :OntologyRanking ;
      codtools:isOutputTypeFor
              :TS-ORS .

agentrole:Agent
      rdfs:comment "Agents in the TS-ORS are reviewers as well as users
commenting on the reviews." .

:TS-ORS
      a       codkernel:DesignTool ;
      rdfs:label "TS-ORS"@en ;
      codtools:hasInputType
              :TrustAnnotationKType , coddata:OntologyKType , :ReviewKType ;
      codtools:hasOutputType
              :OntologyRankKType , :ReviewRankKType , :ReviewKType ;
      codtools:implements :ReviewRanking , :OntologyRanking .

:Review
      a       owl:Class ;
      rdfs:comment "A review consists of a star rating (1-5) and a textual
description justifying this rating. It covers an OntologyProperty." ;
      rdfs:label "Review"@en ;
```

```
      rdfs:subClassOf coddata:Annotation ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/informationobjectsandrepresentatio
nlanguages.owl#isConceptualizedBy> ;
                owl:someValuesFrom agentrole:Agent
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :ReviewKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty intensionextension:isAbout ;
                owl:someValuesFrom codkernel:Ontology
              ] .

:ReviewRankKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Review rank KType"@en ;
      codtools:isOutputTypeFor
              :TS-ORS , :ReviewRanking .

<http://www.ontologydesignpatterns.org/cpont/codo/tsors2codo.owl>
      a       owl:Ontology ;
      rdfs:comment "The Topic-Specific Open Rating System (TS-ORS) allows users
to rate properties of ontologies. Furthermore, other users can then express
trust on these reviews, producing an underlying web of trust. This can be
exploited to personlize the ranking of reviews and ontologies." ;
      owl:imports
<http://www.ontologydesignpatterns.org/cpont/codo/odm2codo.owl> ,
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ;
      owl:versionInfo """0.1: Created by Holger Lewen
0.2: Revised by Aldo Gangemi
0.3: Changed name of Trust class to TrustAnnotation, added labels
0.4: added labels and knowledge types""" .

:ReviewRanking
      a       codinteraction:ComputationalDesignTask ;
      rdfs:comment "Based on the reviews available and the web of trust computed
by the TS-ORS, the reviews are ranked." ;
      rdfs:label "Review ranking"@en ;
      codtools:hasInputType
              :TrustAnnotationKType , :ReviewKType ;
      codtools:hasOutputType
              :ReviewRankKType ;
      codtools:isImplementedIn
              :TS-ORS .

:OntologyRank
      a       owl:Class ;
      rdfs:label "Ontology rank"@en ;
      rdfs:subClassOf coddata:Annotation ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyRanking ;
                owl:onProperty codtools:isOutputDataFor
              ] ;
```

```
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyRankKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty intensionextension:isAbout ;
                owl:someValuesFrom codkernel:Ontology
              ] .

:OntologyRanking
      a       codinteraction:ComputationalDesignTask ;
      rdfs:comment "The ranking of ontologies is based on the trust expressed by
agents in other agents and the reviews available." ;
      rdfs:label "Ontology ranking"@en ;
      codtools:hasInputType
              :TrustAnnotationKType , coddata:OntologyKType , :ReviewKType ;
      codtools:hasOutputType
              :OntologyRankKType ;
      codtools:isImplementedIn
              :TS-ORS .
```

Cicero

```
# baseURI: http://www.ontologydesignpatterns.org/cpont/codo/cicero2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl
# imports: http://www.ontologydesignpatterns.org/cp/owl/specialization.owl

@prefix codworkflows:
<http://www.ontologydesignpatterns.org/cpont/codo/codworkflows.owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix specialization:
<http://www.ontologydesignpatterns.org/cp/owl/specialization.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sequence:  <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl#>
.
@prefix descriptionandsituation:
<http://www.ontologydesignpatterns.org/cp/owl/descriptionandsituation.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl2xml:  <http://www.w3.org/2006/12/owl2-xml#> .
@prefix objectrole:
<http://www.ontologydesignpatterns.org/cp/owl/objectrole.owl#> .
@prefix taskrole:  <http://www.ontologydesignpatterns.org/cp/owl/taskrole.owl#>
.
@prefix taskexecution:
<http://www.ontologydesignpatterns.org/cp/owl/taskexecution.owl#> .
@prefix intensionextension:
<http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix codarg:  <http://www.ontologydesignpatterns.org/cpont/codo/codarg.owl#>
.
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix codinteraction:
<http://www.ontologydesignpatterns.org/cpont/codo/codinteraction.owl#> .
@prefix situation:
<http://www.ontologydesignpatterns.org/cp/owl/situation.owl#> .
@prefix codprojects:
<http://www.ontologydesignpatterns.org/cpont/codo/codprojects.owl#> .
@prefix description:
<http://www.ontologydesignpatterns.org/cp/owl/description.owl#> .
@prefix informationobjectsandrepresentationlanguages:
<http://www.ontologydesignpatterns.org/cp/owl/informationobjectsandrepresentatio
nlanguages.owl#> .
@prefix partof:  <http://www.ontologydesignpatterns.org/cp/owl/partof.owl#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/cicero2codo.owl#> .

codarg:ArgumentKType
      specialization:isSpecializedBy
              :CiceroArgumentKType .

:CastVote
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
Casting a single vote for one of the proposed solutions."""^^xsd:string ;
      rdfs:label "Cast vote"@en ;
```

```
      partof:isPartOf :PreferentialVoting ;
      codtools:hasInputType
              :CiceroArgumentKType ;
      codtools:hasOutputType
              codarg:PositionKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ;
      owl:differentFrom :CiceroAdministrator , :ProjectModerator ,
:IssueModerator , :ProjectMember .

codarg:Idea
      a       owl:Class .

:CiceroWiki
      a       codkernel:DesignTool ;
      rdfs:comment """
The Cicero Argumentation Wiki can be used for discussing issues that are raised
during an OntologyProject with regard to the development of the
ontology."""^^xsd:string ;
      rdfs:label "Cicero wiki"@en ;
      codtools:hasUserType
              :ProjectMember ;
      codtools:implements :PreferentialVoting , :CreateProject ,
:ProposeSolution , :DiscussDesignRationale , :ProvideArgument ,
:CiceroWikiWorkflow , :TakeDecision , :DecideOnSolution , :CreateAnnotation ,
:CastVote , :CreateIssue , :StartPreferentialVoting .

:IssueModerator
      a       codkernel:UserType ;
      rdfs:label "Issue moderator"@en ;
      specialization:isRequiredBy
              :CiceroAdministrator ;
      specialization:requires
              :ProjectModerator ;
      taskrole:hasTask :DecideOnSolution , :StartPreferentialVoting ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
:ProposeSolution , :ProvideArgument , codinteraction:ActionButton ,
:DecideOnSolution , codinteraction:Slideshow , codinteraction:Accordion ,
codinteraction:PulldownButton , :CreateProject , :DiscussDesignRationale ,
:CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow , codinteraction:Stepping ,
codinteraction:Paging , codinteraction:Wizard , :StartPreferentialVoting .

:CiceroArgumentKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Cicero argument KType"@en ;
      specialization:specializes
              codarg:ArgumentKType ;
      codtools:isInputTypeFor
              :CastVote ;
      codtools:isOutputTypeFor
              :ProvideArgument .

:ProjectMember
      a       codkernel:UserType ;
      rdfs:label "Project member"@en ;
      specialization:isRequiredBy
              :ProjectModerator ;
      taskrole:hasTask :PreferentialVoting , :DiscussDesignRationale ;
      codtools:isUserTypeFor
              :CiceroToolkitPlugin , :CiceroWiki ;
```

```
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
:ProposeSolution , :ProvideArgument , codinteraction:ActionButton ,
:DecideOnSolution , codinteraction:Accordion , codinteraction:Slideshow ,
codinteraction:PulldownButton , :CreateProject , :DiscussDesignRationale ,
:CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow , codinteraction:Stepping ,
codinteraction:Paging , codinteraction:Wizard , :StartPreferentialVoting .

coddata:AnnotationKType
      codtools:isOutputTypeFor
              :CreateAnnotation .

:CiceroSolutionProposal
      a       owl:Class ;
      rdfs:label "Cicero solution proposal"@en ;
      rdfs:subClassOf codarg:Idea ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :CiceroSolutionProposalKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:allValuesFrom
informationobjectsandrepresentationlanguages:LinguisticObject ;
                owl:onProperty intensionextension:isExpressedBy
              ] .

<http://www.ontologydesignpatterns.org/cpont/codo/cicero2codo.owl>
      a       owl:Ontology ;
      rdfs:comment """0.1: Description of Cicero created by Klaas Dellschaft
0.2: Adapted by Aldo Gangemi for linking to codolight
0.3: Revised a redundant axiom for CiceroWikiWorkflow
0.4: Made broad refactoring after more careful analysis. Refactored UserType
subclasses as individuals; also refactored the anonymous typing dependent on
those classes as simple object values; added import of specialization.owl
pattern; refactored rdf:type axioms to DesignOperation as inputType axioms to
DesignFunctionality; also refactored the executesTask axioms dependent on those
individuals as isPartOf axioms.
0.5: added labels and knowledge types
0.6: corrected bug in type of CiceroWikiWorkfow""" ;
      owl:imports
<http://www.ontologydesignpatterns.org/cp/owl/specialization.owl> ,
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> .

:CiceroAdministrator
      a       codkernel:UserType ;
      rdfs:label "Cicero administrator"@en ;
      specialization:requires
              :IssueModerator ;
      taskrole:hasTask :CreateProject ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
:ProposeSolution , codinteraction:ActionButton , :ProvideArgument ,
:DecideOnSolution , codinteraction:Slideshow , codinteraction:Accordion ,
codinteraction:PulldownButton , :CreateProject , :DiscussDesignRationale ,
:CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow , codinteraction:Stepping ,
codinteraction:Paging , codinteraction:Wizard , :StartPreferentialVoting .

:CiceroToolkitPluginWorkflow
      a       codworkflows:CollaborativeWorkflow ;
      rdfs:label "Cicero toolkit plugin workflow"@en ;
```

```
      codtools:isImplementedIn
              :CiceroToolkitPlugin ;
      codworkflows:includesFunctionality
              :PreferentialVoting , :CreateProject , :ProposeSolution ,
:ProvideArgument , :DiscussDesignRationale , :TakeDecision , :CreateAnnotation ,
:DecideOnSolution , :CastVote , :CreateIssue , :StartPreferentialVoting ;
      owl:differentFrom :CreateProject , :DecideOnSolution , :ProjectModerator ,
:StartPreferentialVoting .

:CreateAnnotation
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
Annotating ontology elements with a discussion in the Cicero
Wiki."""^^xsd:string ;
      rdfs:label "Create annotation"@en ;
      codtools:hasInputType
              :CiceroIssueKType , coddata:OntologyElementKType ;
      codtools:hasOutputType
              coddata:AnnotationKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , :IssueModerator , codinteraction:Slideshow ,
codinteraction:Accordion , codinteraction:PulldownButton , :ProjectMember ,
:CiceroAdministrator , :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ,
codinteraction:Stepping , :ProjectModerator , codinteraction:Paging ,
codinteraction:Wizard .

:CiceroArgument
      a       owl:Class ;
      rdfs:label "Cicero argument"@en ;
      rdfs:subClassOf codarg:Argument ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :CiceroArgumentKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

intensionextension:isExpressedBy
      a       owl:ObjectProperty .

coddata:OntologyElementKType
      codtools:isInputTypeFor
              :CreateAnnotation .

:PreferentialVoting
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
The preferential voting is an optional phase in the workflow of the Cicero
Wiki."""^^xsd:string ;
      rdfs:label "Preferential voting"@en ;
      partof:hasPart :CastVote ;
      sequence:directlyFollows
              :StartPreferentialVoting ;
      sequence:follows :CreateProject , :DiscussDesignRationale ,
:StartPreferentialVoting ;
      sequence:precedes :DecideOnSolution ;
      taskrole:isTaskOf :ProjectMember ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
```

```
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , :IssueModerator , codinteraction:Slideshow ,
codinteraction:Accordion , codinteraction:PulldownButton , :ProjectMember ,
:CiceroAdministrator , :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ,
:ProjectModerator , codinteraction:Stepping , codinteraction:Paging ,
codinteraction:Wizard .

:CiceroToolkitPlugin
      a       codkernel:DesignTool ;
      rdfs:comment "The Cicero Plugin for the NeOn Toolkit can be used for
annotating ontology elements with related discussions in a Cicero Argumentation
Wiki."^^xsd:string ;
      rdfs:label "Cicero toolkit plugin"@en ;
      codtools:hasUserType
              :ProjectMember ;
      codtools:implements :PreferentialVoting , :CreateProject ,
:ProposeSolution , :DiscussDesignRationale , :ProvideArgument ,
:CiceroToolkitPluginWorkflow , :TakeDecision , :DecideOnSolution ,
:CreateAnnotation , :CastVote , :CreateIssue , :StartPreferentialVoting .

:CiceroIssueKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Cicero issue KType"@en ;
      specialization:specializes
              codarg:ArgumentationThreadKType ;
      codtools:isInputTypeFor
              :ProposeSolution , :CreateAnnotation ;
      codtools:isOutputTypeFor
              :CreateIssue .

:ProposeSolution
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
A single solution is proposed how to solve the previously raised
issue."""^^xsd:string ;
      rdfs:label "Propose solution"@en ;
      partof:isPartOf :DiscussDesignRationale ;
      sequence:follows :CreateIssue ;
      sequence:precedes :ProvideArgument ;
      codtools:hasInputType
              :CiceroIssueKType ;
      codtools:hasOutputType
              :CiceroSolutionProposalKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ;
      owl:differentFrom :ProjectModerator .

:ProvideArgument
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
A supporting or objecting argument with regard to a specific solution proposal
is given."""^^xsd:string ;
      rdfs:label "Provide argument"@en ;
      partof:isPartOf :DiscussDesignRationale ;
      sequence:follows :ProposeSolution , :CreateIssue ;
      codtools:hasInputType
              :CiceroSolutionProposalKType ;
      codtools:hasOutputType
              :CiceroArgumentKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ;
```

```
      owl:differentFrom :ProjectModerator .

codarg:Argument
      a       owl:Class .

:TakeDecision
      a       codkernel:DesignFunctionality ;
      rdfs:label "Take decision"@en ;
      partof:isPartOf :DecideOnSolution ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
      owl:differentFrom :CiceroAdministrator , :ProjectModerator ,
:IssueModerator , :ProjectMember .

:DecideOnSolution
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
An Issue Moderator decides, which solution should be implemented in the ontology
project. The decision is based on the previous discussion and may be based on a
previously held preferential voting."""^^xsd:string ;
      rdfs:label "Decide on solution"@en ;
      partof:hasPart :TakeDecision ;
      sequence:follows :PreferentialVoting , :CreateProject ,
:DiscussDesignRationale , :StartPreferentialVoting ;
      taskrole:isTaskOf :IssueModerator ;
      codtools:hasInputType
              codarg:PositionKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , :CiceroToolkitPluginWorkflow ,
codinteraction:Stepping , :ProjectModerator , codinteraction:Paging ,
codinteraction:Accordion , codinteraction:Slideshow , codinteraction:Wizard ,
codinteraction:PulldownButton .

informationobjectsandrepresentationlanguages:LinguisticObject
      a       owl:Class .

:CiceroSolutionProposalKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Cicero solution proposal KType"@en ;
      specialization:specializes
              codarg:IdeaKType ;
      codtools:isInputTypeFor
              :ProvideArgument ;
      codtools:isOutputTypeFor
              :ProposeSolution .

:CreateProject
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
Creating a new discussion project that groups all discussions related to a
specific ontology project."""^^xsd:string ;
      rdfs:label "Create project"@en ;
      sequence:precedes :PreferentialVoting , :DiscussDesignRationale ,
:DecideOnSolution , :StartPreferentialVoting ;
      taskrole:isTaskOf :CiceroAdministrator ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , :CiceroToolkitPluginWorkflow ,
```

```
codinteraction:Stepping , :ProjectModerator , codinteraction:Paging ,
codinteraction:Accordion , codinteraction:Slideshow ,
codinteraction:PulldownButton , codinteraction:Wizard .

:DiscussDesignRationale
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
During the discussion of an issue, the members of a project exchange solution
proposals and arguments."""^^xsd:string ;
      rdfs:label "Discuss design rationale"@en ;
      partof:hasPart :ProposeSolution , :ProvideArgument , :CreateIssue ;
      sequence:directlyPrecedes
              :StartPreferentialVoting ;
      sequence:follows :CreateProject ;
      sequence:precedes :PreferentialVoting , :DecideOnSolution ,
:StartPreferentialVoting ;
      taskrole:isTaskOf :ProjectMember ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , codinteraction:Slideshow ,
codinteraction:Accordion , codinteraction:PulldownButton , :ProjectMember ,
:CiceroWikiWorkflow , :CiceroToolkitPluginWorkflow , :ProjectModerator ,
codinteraction:Stepping , codinteraction:Paging , codinteraction:Wizard .

:CiceroWikiWorkflow
      a       codworkflows:CollaborativeWorkflow ;
      rdfs:label "Cicero wiki workflow"@en ;
      codtools:isImplementedIn
              :CiceroWiki ;
      codworkflows:includesFunctionality
              :PreferentialVoting , :CreateProject , :ProposeSolution ,
:ProvideArgument , :DiscussDesignRationale , :TakeDecision , :CreateAnnotation ,
:DecideOnSolution , :CastVote , :CreateIssue , :StartPreferentialVoting ;
      owl:differentFrom :CreateProject , :DecideOnSolution , :ProjectModerator ,
:StartPreferentialVoting .

:CiceroIssue
      a       owl:Class ;
      rdfs:label "Cicero issue"@en ;
      rdfs:subClassOf codarg:ArgumentationThread ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :CiceroIssueKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:ProjectModerator
      a       codkernel:UserType ;
      rdfs:label "Project moderator"@en ;
      specialization:isRequiredBy
              :IssueModerator ;
      specialization:requires
              :ProjectMember ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
:ProvideArgument , codinteraction:ActionButton , :DecideOnSolution ,
codinteraction:Accordion , codinteraction:Slideshow ,
codinteraction:PulldownButton , :CreateProject , :CiceroToolkitPluginWorkflow ,
codinteraction:Stepping , codinteraction:Paging , codinteraction:Wizard ,
:StartPreferentialVoting .
```

```
codarg:ArgumentationThreadKType
      specialization:isSpecializedBy
              :CiceroIssueKType .

codarg:IdeaKType
      specialization:isSpecializedBy
              :CiceroSolutionProposalKType .

:CreateIssue
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
Creating a new issue that should be discussed in the following."""^^xsd:string ;
      rdfs:label "Create issue"@en ;
      partof:isPartOf :DiscussDesignRationale ;
      sequence:precedes :ProposeSolution , :ProvideArgument ;
      codtools:hasOutputType
              :CiceroIssueKType ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ;
      owl:differentFrom :CiceroAdministrator , :ProjectModerator ,
:IssueModerator , :ProjectMember .

codkernel:KnowledgeResource
      a       owl:Class .

:StartPreferentialVoting
      a       codkernel:DesignFunctionality ;
      rdfs:comment """
An Issue Moderator explicitly starts a preferential voting. This phase is
optional, i.e. either an Issue Moderator can directly take a decision or an
automatic transistion into the PreferentialVoting phase is triggered after a
certain time."""^^xsd:string ;
      rdfs:label "Start preferential voting"@en ;
      sequence:directlyFollows
              :DiscussDesignRationale ;
      sequence:directlyPrecedes
              :PreferentialVoting ;
      sequence:follows :CreateProject , :DiscussDesignRationale ;
      sequence:precedes :PreferentialVoting , :DecideOnSolution ;
      taskrole:isTaskOf :IssueModerator ;
      codworkflows:isFunctionalityIncludedIn
              :CiceroToolkitPluginWorkflow , :CiceroWikiWorkflow ;
      owl:differentFrom codinteraction:GuidedTour , codinteraction:Breadcrumbs ,
codinteraction:ActionButton , :CiceroToolkitPluginWorkflow , :ProjectModerator ,
codinteraction:Stepping , codinteraction:Paging , codinteraction:Accordion ,
codinteraction:Slideshow , codinteraction:PulldownButton , codinteraction:Wizard
.
```

Collaboration Server

```
# baseURI:
http://www.ontologydesignpatterns.org/cpont/codo/collaborationserver2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/odm2codo.owl

@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix owl1odm:
<http://www.ontologydesignpatterns.org/cpont/codo/odm2codo.owl#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix sequence:  <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl#>
.
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/collaborationserver2codo.owl#>
.
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

:OWLOntology
      a       owl:Class ;
      rdfs:label "OWLOntology"@en ;
      rdfs:subClassOf codkernel:Ontology ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OWLOntologyKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

owl1odm:OWL1DatatypePropertyKType
      codtools:isInputTypeFor
              :CreateAndEditDatatypePropertiesRemotely ;
      codtools:isOutputTypeFor
              :CreateAndEditDatatypePropertiesRemotely .

:F-LogicMappingRuleKType
      a       codkernel:KnowledgeType ;
      rdfs:comment "A F-logic mapping rule is a special case of a F-logic rule."
;
      rdfs:label "F-Logic mapping rule KType"@en ;
      codtools:isInputTypeFor
              :CreateAndEditMappingRulesRemotely ;
      codtools:isOutputTypeFor
              :CreateAndEditMappingRulesRemotely .

:F-LogicRelationKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic relation KType"@en .

:F-LogicConcept
      a       owl:Class ;
      rdfs:label "F-Logic concept"@en ;
```

```
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#OntologyElement> ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicConceptKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#DatatypeProperty>
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue owl1odm:OWL1DatatypePropertyKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:F-LogicRule
      a       owl:Class ;
      rdfs:label "F-Logic rule"@en ;
      rdfs:subClassOf coddata:Rule ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicRuleKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:CreateOntologiesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create ontologies ontologies remotely." ;
      rdfs:label "Create ontologies remotely"@en ;
      sequence:follows :CreateProjectsRemotely ;
      sequence:precedes :CreateAndEditMappingRulesRemotely ,
:AccessSharedOntologiesRemotely , :CreateAndEditRulesRemotely ,
:ExecuteQueriesRemotely , :CreateAndEditObjectPropertiesRemotely ,
:CreateAndEditDatatypePropertiesRemotely , :CreateAndEditClassesRemotely ,
:CreateAndEditQueriesRemotely ;
      codtools:hasOutputType
              :OWLOntologyKType , :F-LogicOntologyKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:F-LogicRuleKType
      a       codkernel:KnowledgeType ;
      rdfs:comment "F-logic rules consist of a rule body and a rule head." ;
      rdfs:label "F-Logic rule KType"@en .

:AccessSharedOntologiesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may access shared ontologies remotely." ;
      rdfs:label "Access shared ontologies remotely"@en ;
      sequence:follows :CreateProjectsRemotely , :CreateOntologiesRemotely ;
      codtools:hasOutputType
              :OWLOntologyKType , :F-LogicOntologyKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:F-LogicConceptKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic concept KType"@en .
```

```
:F-LogicOntologyKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic ontology KType"@en ;
      codtools:isOutputTypeFor
              :AccessSharedOntologiesRemotely , :CreateOntologiesRemotely .

:F-LogicAttribute
      a       owl:Class ;
      rdfs:label "F-Logic attribute"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#OntologyElement> ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicAttributeKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:CreateAndEditDatatypePropertiesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit datatype properties in
shared ontologies remotely." ;
      rdfs:label "Create and edit datatype properties remotely"@en ;
      sequence:follows :CreateProjectsRemotely , :CreateOntologiesRemotely ;
      sequence:precedes :CreateAndEditMappingRulesRemotely ,
:CreateAndEditRulesRemotely , :ExecuteQueriesRemotely ,
:CreateAndEditQueriesRemotely ;
      codtools:hasInputType
              owl1odm:OWL1DatatypePropertyKType ;
      codtools:hasOutputType
              owl1odm:OWL1DatatypePropertyKType ;
      codtools:isImplementedIn
              :CollaborationServer .

owl1odm:OWL1ObjectPropertyKType
      codtools:isOutputTypeFor
              :CreateAndEditObjectPropertiesRemotely .

:CreateAndEditMappingRulesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit mapping rules in shared
ontologies remotely." ;
      rdfs:label "Create and edit mapping rules remotely"@en ;
      sequence:follows :CreateAndEditObjectPropertiesRemotely ,
:CreateProjectsRemotely , :CreateAndEditDatatypePropertiesRemotely ,
:CreateAndEditClassesRemotely , :CreateOntologiesRemotely ;
      codtools:hasInputType
              :F-LogicMappingRuleKType ;
      codtools:hasOutputType
              :F-LogicMappingRuleKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:SPARQLQuery
      a       owl:Class ;
      rdfs:label "SPARQLQuery"@en ;
      rdfs:subClassOf coddata:Query ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :SPARQLQueryKType ;
```

```
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:CollaborationServer
      a       codkernel:DesignTool ;
      rdfs:label "Collaboration server"@en ;
      codtools:implements :CreateAndEditMappingRulesRemotely ,
:AccessSharedOntologiesRemotely , :CreateAndEditRulesRemotely ,
:ExecuteQueriesRemotely , :CreateAndEditObjectPropertiesRemotely ,
:CreateProjectsRemotely , :CreateAndEditDatatypePropertiesRemotely ,
:CreateAndEditClassesRemotely , :CreateOntologiesRemotely ,
:CreateAndEditQueriesRemotely .

:ExecuteQueriesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may execute queries remotely." ;
      rdfs:label "Execute queries remotely"@en ;
      sequence:follows :CreateAndEditObjectPropertiesRemotely ,
:CreateProjectsRemotely , :CreateAndEditDatatypePropertiesRemotely ,
:CreateAndEditClassesRemotely , :CreateOntologiesRemotely ,
:CreateAndEditQueriesRemotely ;
      codtools:hasInputType
              :SPARQLQueryKType , :F-LogicQueryKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:CreateAndEditClassesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit classes in shared
ontologies remotely." ;
      rdfs:label "Create and edit classes remotely"@en ;
      sequence:follows :CreateProjectsRemotely , :CreateOntologiesRemotely ;
      sequence:precedes :CreateAndEditMappingRulesRemotely ,
:CreateAndEditRulesRemotely , :ExecuteQueriesRemotely ,
:CreateAndEditQueriesRemotely ;
      codtools:hasInputType
              owl1odm:OWL1ClassKType ;
      codtools:hasOutputType
              owl1odm:OWL1ClassKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:OWLOntologyKType
      a       codkernel:KnowledgeType ;
      rdfs:label "OWLOntology KType"@en ;
      codtools:isOutputTypeFor
              :AccessSharedOntologiesRemotely , :CreateOntologiesRemotely .

:F-LogicQuery
      a       owl:Class ;
      rdfs:label "F-Logic query"@en ;
      rdfs:subClassOf coddata:Query ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicQueryKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:F-LogicOntology
```

```
      a       owl:Class ;
      rdfs:label "F-Logic ontology"@en ;
      rdfs:subClassOf codkernel:Ontology ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicOntologyKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:F-LogicProjectKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic project KType"@en ;
      codtools:isOutputTypeFor
              :CreateProjectsRemotely .

:F-LogicMappingRule
      a       owl:Class ;
      rdfs:label "F-Logic mapping rule"@en ;
      rdfs:subClassOf coddata:Rule ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicMappingRuleKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:OWLProjectKType
      a       codkernel:KnowledgeType ;
      rdfs:label "OWLProject KType"@en ;
      codtools:isOutputTypeFor
              :CreateProjectsRemotely .

:F-LogicQueryKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic query KType"@en ;
      codtools:isInputTypeFor
              :ExecuteQueriesRemotely , :CreateAndEditQueriesRemotely ;
      codtools:isOutputTypeFor
              :CreateAndEditQueriesRemotely .

<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#Class>
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue owl1odm:OWL1ClassKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

coddata:RuleKType
      codtools:isOutputTypeFor
              :CreateAndEditRulesRemotely .

:CreateProjectsRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create projects remotely." ;
      rdfs:label "Create projects remotely"@en ;
      sequence:precedes :CreateAndEditMappingRulesRemotely ,
:AccessSharedOntologiesRemotely , :CreateAndEditRulesRemotely ,
:ExecuteQueriesRemotely , :CreateAndEditObjectPropertiesRemotely ,
```

```
:CreateAndEditDatatypePropertiesRemotely , :CreateAndEditClassesRemotely ,
:CreateOntologiesRemotely , :CreateAndEditQueriesRemotely ;
      codtools:hasOutputType
              :F-LogicProjectKType , :OWLProjectKType ;
      codtools:isImplementedIn
              :CollaborationServer .

<http://www.ontologydesignpatterns.org/cpont/codo/collaborationserver2codo.owl>
      a       owl:Ontology ;
      owl:imports
<http://www.ontologydesignpatterns.org/cpont/codo/odm2codo.owl> ,
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ;
      owl:versionInfo """0.1: Created by Anne Becker and Michael Erdmann
0.2: Revised by Aldo Gangemi (finetuned types, added relations between
functionalities and kowledge types, added design tool)"""^^xsd:string .

:F-LogicRelation
      a       owl:Class ;
      rdfs:label "F-Logic relation"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#OntologyElement> ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :F-LogicRelationKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:CreateAndEditObjectPropertiesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit object properties in
shared ontologies remotely." ;
      rdfs:label "Create and edit object properties remotely"@en ;
      sequence:follows :CreateProjectsRemotely , :CreateOntologiesRemotely ;
      sequence:precedes :CreateAndEditMappingRulesRemotely ,
:ExecuteQueriesRemotely , :CreateAndEditRulesRemotely ,
:CreateAndEditQueriesRemotely ;
      codtools:hasOutputType
              owl1odm:OWL1ObjectPropertyKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:CreateAndEditQueriesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit queries in shared
ontologies remotely." ;
      rdfs:label "Create and edit queries remotely"@en ;
      sequence:follows :CreateAndEditObjectPropertiesRemotely ,
:CreateAndEditDatatypePropertiesRemotely , :CreateProjectsRemotely ,
:CreateOntologiesRemotely , :CreateAndEditClassesRemotely ;
      sequence:precedes :ExecuteQueriesRemotely ;
      codtools:hasInputType
              :F-LogicQueryKType ;
      codtools:hasOutputType
              :F-LogicQueryKType ;
      codtools:isImplementedIn
              :CollaborationServer .

:SPARQLQueryKType
      a       codkernel:KnowledgeType ;
      rdfs:label "SPARQLQuery KType"@en ;
```

```
      codtools:isInputTypeFor
              :ExecuteQueriesRemotely .

:F-LogicAttributeKType
      a       codkernel:KnowledgeType ;
      rdfs:label "F-Logic attribute KType"@en .

owl1odm:OWL1ClassKType
      codtools:isInputTypeFor
              :CreateAndEditClassesRemotely ;
      codtools:isOutputTypeFor
              :CreateAndEditClassesRemotely .

<http://www.ontologydesignpatterns.org/ont/odm/owl10b.owl#ObjectProperty>
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue owl1odm:OWL1ObjectPropertyKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:CreateAndEditRulesRemotely
      a       codkernel:DesignFunctionality ;
      rdfs:comment "Multiple users may create and edit  rules in shared
ontologies remotely." ;
      rdfs:label "Create and edit rules remotely"@en ;
      sequence:follows :CreateAndEditObjectPropertiesRemotely ,
:CreateProjectsRemotely , :CreateAndEditDatatypePropertiesRemotely ,
:CreateAndEditClassesRemotely , :CreateOntologiesRemotely ;
      codtools:hasOutputType
              coddata:RuleKType ;
      codtools:isImplementedIn
              :CollaborationServer .
```

OntoConto

```
# baseURI:
http://www.ontologydesignpatterns.org/cpont/codo/facebookmapper2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl

@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/facebookmapper2codo.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

<http://www.ontologydesignpatterns.org/cpont/codo/facebookmapper2codo.owl>
      a       owl:Ontology ;
      owl:imports
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ;
      owl:versionInfo """0.1: Derived by Aldo Gangemi from Dunja Mladenic'
description of Facebook application plugin to NTK.
0.2: added labels and knowledge types"""^^xsd:string .

<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#NetworkOfOntologie
sKType>
      codtools:isInputTypeFor
              :FacebookBasedMapper .

<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#OntologyMappingKTy
pe>
      codtools:isOutputTypeFor
              :FacebookBasedMapper .

:FacebookBasedMapper
      a       codkernel:DesignTool ;
      rdfs:comment """Facebook-based application, integrated with alignment
server [JSI, D3.2.2]
The main goal of this application is to support collaborative mapping on
networked onotologies. The application consists of server side and client side.
-                      Server side takes ontologies as input, creates mappings
between them and stores the ontologies and mappings into a database.
-                      Client side get data from the server side and enables
ontology editing, mappings editing and visualization of ontologies and mappings
between them. Client side is written in FLASH ActionScript and integrated into
Facebook as standard application using Facebook plug-in architecture.""" ;
      rdfs:label "Facebook based mapper"@en ;
      codtools:hasInputType

<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#NetworkOfOntologie
sKType> ;
      codtools:hasOutputType

<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#OntologyMappingKTy
pe> .
```

SocialOnto

```
# baseURI: http://www.ontologydesignpatterns.org/cpont/codo/logsna2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl
# imports: http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl

@prefix codinteraction:
<http://www.ontologydesignpatterns.org/cpont/codo/codinteraction.owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix timeinterval:
<http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix specialization:
<http://www.ontologydesignpatterns.org/cp/owl/specialization.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/logsna2codo.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

:UserId
      a       owl:Class ;
      rdfs:label "User id"@en ;
      rdfs:subClassOf codkernel:KnowledgeResource ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :UserIdKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:OntologyElementId
      a       owl:Class ;
      rdfs:label "Ontology element id"@en ;
      rdfs:subClassOf codkernel:KnowledgeResource ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyElementIdKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] .

:GraphKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Graph KType"@en ;
      codtools:isOutputTypeFor
              :AnalysisOfMultipleOntologyEditorsActivity .

:Graph
      a       owl:Class ;
      rdfs:label "Graph"@en ;
      rdfs:subClassOf coddata:DataStructure ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
```

```
                owl:hasValue :GraphKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/descriptionandsituation.owl#isDesc
ribedBy> ;
                owl:someValuesFrom :GraphProperty
              ] .

:UserIdKType
      a       codkernel:KnowledgeType ;
      rdfs:label "User id KType"@en .

:OntologyEditingLogFileKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology editing log file KType"@en ;
      codtools:isInputTypeFor
              :LogFileSocialNetworkAnalysisTool ,
:AnalysisOfMultipleOntologyEditorsActivity .

:MultipleOntologyEditorsActivity
      a       owl:Class ;
      rdfs:label "Multiple ontology editors activity"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cpont/codo/codprojects.owl#OntologyProjec
tExecution> ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/situation.owl#isSettingFor> ;
                owl:someValuesFrom timeinterval:TimeInterval
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/situation.owl#isSettingFor> ;
                owl:someValuesFrom codkernel:Ontology
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/situation.owl#isSettingFor> ;
                owl:someValuesFrom
<http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl#Agent>
              ] .

:AnalysisOfMultipleOntologyEditorsActivity
      a       codinteraction:ComputationalDesignTask ,
codkernel:DesignFunctionality ;
      rdfs:label "Analysis of multiple ontology editors activity"@en ;
      codtools:hasInputType
              :OntologyEditingLogFileKType ;
      codtools:hasOutputType
              :GraphKType , :GraphPropertyDataKType ;
      codtools:isImplementedIn
              :LogFileSocialNetworkAnalysisTool .

:OntologyEditingLogFileLine
```

```
      a       owl:Class ;
      rdfs:label "Ontology editing log file line"@en ;
      rdfs:subClassOf coddata:DataStructure ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyEditingLogFileLineKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/partof.owl#hasPart> ;
                owl:someValuesFrom :UserId
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/partof.owl#hasPart> ;
                owl:someValuesFrom :OntologyElementId
              ] .

:GraphPropertyData
      a       owl:Class ;
      rdfs:label "Graph property data"@en ;
      rdfs:subClassOf coddata:DataStructure ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :GraphPropertyDataKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl#expresses>
;
                owl:someValuesFrom :GraphProperty
              ] .

:OntologyEditingLogFile
      a       owl:Class ;
      rdfs:comment "Each line of the log-file contains at least timestamp,
userId, conceptId or relationId (that the user has edited)" ;
      rdfs:label "Ontology editing log file"@en ;
      rdfs:subClassOf coddata:DataStructure ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl#isAbout> ;
                owl:someValuesFrom :MultipleOntologyEditorsActivity
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue :OntologyEditingLogFileKType ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/classification.owl#isClassifiedBy>
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:allValuesFrom :OntologyEditingLogFileLine ;
```

```
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl#hasMember>
              ] .

:LogFileSocialNetworkAnalysisTool
      a       codkernel:DesignTool ;
      rdfs:comment """The main goal of this tool is analysis of activity of
different users when editing the same ontology.
Social network analysis tool takes a log-file of ontology editing as input,
where each line of the log-file contains at least timestamp, userId, conceptId
or relationId (that the user has edited). It outputs:
-     graph (vertex=userId, link=linking similar users, weight on the link =
similarity of two users)
-     graph properties obtained by social network analysis
(centrality,components,â€¦)""" ;
      rdfs:label "Log file social network analysis tool"@en ;
      codtools:hasInputType
              :OntologyEditingLogFileKType ;
      codtools:implements :AnalysisOfMultipleOntologyEditorsActivity .

:GraphPropertyDataKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Graph property data KType"@en ;
      codtools:isOutputTypeFor
              :AnalysisOfMultipleOntologyEditorsActivity .

<http://www.ontologydesignpatterns.org/cpont/codo/logsna2codo.owl>
      a       owl:Ontology ;
      owl:imports
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ,
<http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl> ;
      owl:versionInfo """0.1: Derived by Aldo Gangemi from Dunja Mladenic'
description of the SNA NTK plugin.
0.2: added labels and knowledge types"""^^xsd:string .

:OntologyEditingLogFileLineKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology editing log file line KType"@en .

:GraphProperty
      a       owl:Class ;
      rdfs:label "Graph property"@en ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cp/owl/description.owl#Description> .

:OntologyElementIdKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Ontology element id KType"@en .
```

COAT

```
# baseURI: http://www.ontologydesignpatterns.org/cpont/codo/coat2codo.owl
# imports: http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl

@prefix codinteraction:
<http://www.ontologydesignpatterns.org/cpont/codo/codinteraction.owl#> .
@prefix taskrole:  <http://www.ontologydesignpatterns.org/cp/owl/taskrole.owl#>
.
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix description:
<http://www.ontologydesignpatterns.org/cp/owl/description.owl#> .
@prefix codkernel:
<http://www.ontologydesignpatterns.org/cpont/codo/codkernel.owl#> .
@prefix coddata:
<http://www.ontologydesignpatterns.org/cpont/codo/coddata.owl#> .
@prefix codtools:
<http://www.ontologydesignpatterns.org/cpont/codo/codtools.owl#> .
@prefix sequence:  <http://www.ontologydesignpatterns.org/cp/owl/sequence.owl#>
.
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :
<http://www.ontologydesignpatterns.org/cpont/codo/coat2codo.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

:AnnotationDeletion
      a       codkernel:DesignFunctionality ;
      rdfs:label "Annotation deletion"@en , "Annotation deletion"^^xsd:string ;
      codtools:isImplementedIn
              :COAT .

:ClickingOnClass
      a       codinteraction:ComputationalDesignTask ;
      rdfs:label "Clicking on class"@en , "Clicking"^^xsd:string ;
      description:isConceptUsedIn
              :COATWorkflow ;
      sequence:directlyFollows
              :TextSpanSelection ;
      sequence:directlyPrecedes
              :ElementAssociation ;
      sequence:follows :TextSpanSelection ;
      sequence:precedes :ElementAssociation ;
      taskrole:isTaskOf :TextAnnotator .

:TextSpan
      a       owl:Class ;
      rdfs:label "Text span"@en , "Text span"^^xsd:string ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cp/owl/informationobjectsandrepresentatio
nlanguages.owl#LinguisticObject> ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cpont/codo/codinterfaces.owl#hasColour> ;
                owl:someValuesFrom
<http://www.ontologydesignpatterns.org/cpont/codo/codinterfaces.owl#Colour>
              ] ;
      rdfs:subClassOf
```

```
              [ a       owl:Restriction ;
                owl:onProperty
<http://www.ontologydesignpatterns.org/cp/owl/partof.owl#isPartOf> ;
                owl:someValuesFrom :Text
              ] .

coddata:OntologyKType
      codtools:isInputTypeFor
              :COAT .

:ElementAssociation
      a       codinteraction:ComputationalDesignTask ;
      rdfs:label "Element association"@en , "Annotation generation"^^xsd:string
;
      sequence:directlyFollows
              :ClickingOnClass ;
      sequence:follows :ClickingOnClass , :TextSpanSelection .

:AnnotationModification
      a       codkernel:DesignFunctionality ;
      rdfs:label "Annotation modification"@en , "Annotation
modification"^^xsd:string ;
      codtools:isImplementedIn
              :COAT .

:Text
      a       owl:Class ;
      rdfs:label "Text"@en , "Text"^^xsd:string ;
      rdfs:subClassOf
<http://www.ontologydesignpatterns.org/cp/owl/informationobjectsandrepresentatio
nlanguages.owl#LinguisticObject> .

coddata:AnnotationKType
      codtools:isOutputTypeFor
              :COAT .

:TextAnnotator
      a       codkernel:UserType ;
      rdfs:label "Text annotator"@en , "Text annotator"^^xsd:string ;
      description:isConceptUsedIn
              :COATWorkflow ;
      taskrole:hasTask :ClickingOnClass , :TextSpanSelection ;
      codtools:isUserTypeFor
              :COAT .

:TextSpanKType
      a       codkernel:KnowledgeType ;
      rdfs:label "Text span KType"@en ;
      codtools:isInputTypeFor
              :COAT .

:AnnotationAddition
      a       codkernel:DesignFunctionality ;
      rdfs:label "Annotation addition"@en , "Annotation addition"^^xsd:string ;
      codtools:isImplementedIn
              :COAT .

<http://www.ontologydesignpatterns.org/cpont/codo/coat2codo.owl>
      a       owl:Ontology ;
      rdfs:comment """The USFD collaborative text annotation tool COAT is a web
service for the annotation of TextSpan(s) as ontology instances. The main
```

```
codkernel:DesignFunctionality of this tool is OntologyPopulation according to a
pre-defined ontology.

User types: a number of text annotators who populate an ontology
Input data: Ontology containing the allowed classes to be used as annotations.
Input data: LinguisticObjects within a collection of documents
Output data: Annotations

The text annotators participate by adding text annotation within a possibly
distributive environment. They do not necessarily have to share the same
location when performing the annotation tasks.
The ontology is set within a multilingual situation, i.e. texts in multiple
languages can be annotated with concepts from the same ontology.

On the screen, the window is divided into two panes. The text is represented in
the left pane, whereas the ontology is displayed in the right pane, with
different colours for each concept. The annotator selects a span of text and
then clicks on the ontology class of which this span is deemed an instance. The
text span will then assume the same colour as the concept within the ontology
pane, which indicates that the Annotation has been associated with the text
span.

The tool allows manual creation and verification of ontology instances. The
annotators can work from scratch by manually creating annotations, or on the
basis of existing annotations provided by eco-annotators with whom they, in C-
ODO terms, co-participate. These are either human or automatic annotation
procedures. Both are covered by C-ODO:Agent.
The creation and verification activities include:
- the addition of annotations using the current ontology;
- the deletion of annotations;
- the modification of annotations in terms of span size and ontology class.

The annotation process can raise issues regarding e.g. the granularity of the
ontology concepts with respect to the coverage of relevant concepts in the
documents.
If after an argumentation round a new ontology version is created, the
annotation process needs to be iteratively applied to those instances whose
classification has been refined in this newer ontology version. For new concepts
that represent an addition to the conceptual coverage of the ontology, and
therefore do not represent a refinement, the annotation process needs to be
performed from scratch.""" ;
      owl:imports
<http://www.ontologydesignpatterns.org/cpont/codo/codolight.owl> ;
      owl:versionInfo """Created by Aldo Gangemi based on specs by Wim Peters
(see comment)
0.2: added assertion of inferences
0.3: broadened domain for hasColour in order to include linguistic objects
0.4: added labels and knowledge types
0.5: corrected bug in type of COAT Workflow"""^^xsd:string .

:OntologyPopulation
      a       codkernel:DesignFunctionality ;
      rdfs:label "Ontology population"@en ;
      codtools:isImplementedIn
              :COAT .

:COAT
      a       codkernel:DesignTool ;
      rdfs:comment """COAT is a web service for the annotation of TextSpan(s) as
ontology instances. The main codkernel:DesignFunctionality of this tool is
OntologyPopulation according to a pre-defined ontology.
```

```
User types: a number of text annotators who populate an ontology
Input data: Ontology containing the allowed classes to be used as annotations.
Input data: LinguisticObjects within a collection of documents
Output data: Annotations

The text annotators participate by adding text annotation within a possibly
distributive environment. They do not necessarily have to share the same
location when performing the annotation tasks.
The ontology is set within a multilingual situation, i.e. texts in multiple
languages can be annotated with concepts from the same ontology.""" ;
      rdfs:label "COAT"@en , "Collaborative Ontology Annotation Tool
{@en}"^^xsd:string ;
      codtools:hasInputType
              :TextSpanKType , coddata:OntologyKType ;
      codtools:hasOutputType
              coddata:AnnotationKType ;
      codtools:hasUserType
              :TextAnnotator ;
      codtools:implements :AnnotationDeletion , :AnnotationAddition ,
:OntologyPopulation , :AnnotationModification .

:TextSpanSelection
      a       codinteraction:ComputationalDesignTask ;
      rdfs:label "Text span selection"@en , "Selection"^^xsd:string ;
      description:isConceptUsedIn
              :COATWorkflow ;
      sequence:directlyPrecedes
              :ClickingOnClass ;
      sequence:precedes :ClickingOnClass , :ElementAssociation ;
      taskrole:isTaskOf :TextAnnotator .

:COATWorkflow
      a       codkernel:DesignWorkflow ;
      rdfs:comment "The workflow of COAT includes two computational tasks:
CickingOnClass and TextSpanSelection, and one role: TextAnnotator" ;
      rdfs:label "COAT Workflow"@en ;
      description:usesConcept
              :TextAnnotator , :ClickingOnClass , :TextSpanSelection .
```

Document Identifier:	NEON/2009/D2.3.2/v0.2
Class Deliverable:	NEON EU-IST-2005-027595
Version:	V0.2
Date:	February 09, 2009
State:	Draft
Distribution:	Public, Restricted, Confidential