NeOn-project.org

NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D2.1.1 Design rationales for collaborative development of networked ontologies – State of the art and the *Collaborative Ontology Design Ontology*

Deliverable Co-ordinator: **Carola Catenacci (CNR)**

Deliverable Co-ordinating Institution: **CNR**

Other Authors: Aldo Gangemi (CNR), Jos Lehmann (CNR), Malvina Nissim (CNR), Valentina Presutti (CNR), Gerardo Steve (CNR)

With Contributions from: Nicola Guarino, Claudio Masolo (CNR), Holger Lewen (UKARL), Klaas Dellschaft (UKO-LD), Marta Sabou (OU)

Abstract:

An ontology for modelling requirements and solutions related to the collaborative design of networked ontologies, supplemented with a survey of the state of the art in methods and tools. The ontology and the survey are companion to the forthcoming deliverables from WP2, which will describe specific solutions for collaborative ontology design.

Document Identifier:	NEON/2007/D2.1.1/v2.7	Date due:	February 15th, 2007
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 15th, 2007
Project start date:	March 1, 2006	Version:	V2.7
Project duration:	4 years	State:	Final
		Distribution:	Public

Keyword list: collaborative ontology design, ontology project, design rationale, argumentation, workflow, design pattern, design solution, social-level requirements, functionalities, ontology metadata, meta-modelling.

NeOn Consortium

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

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta} @open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 28 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Richard Benjamins E-mail address: rbenjamins@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Asociación Española de Comercio Electrónico (AECE)** C/Alcalde Barnils, Avenida Diagonal 437 08036 Barcelona Spain Contact person: Jose Luis Zimmerman E-mail address: jlzimmerman@fecemd.org
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

Work package participants

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

Consiglio Nazionale delle Ricerche (CNR)

Universidad Politécnica di Madrid (UPM)

Open University (OU)

Universität Koblenz-Landau (UKO-LD)

University of Sheffield (USFD)

Universität Karlsruhe – TH (UKARL)

Institut National de Recherche en Informatique et en Automatique (INRIA)

Executive Summary

The deliverable is organized as follows. The Introduction positions this deliverable in the NeOn context, and presents the basic organization of a formal framework for collaborative ontology design, and what notions are treated. In Section 2, Requirements and tools for collaboration in cooperative knowledge communities, we provide a state of art about definitions, methods and tools for collaboration in knowledge-intensive work, and why we need a conceptual framework to discuss them and to move on. In Section 3, Overview and foundations of the Collaborative Ontology Design Ontology, we give an overview of the conceptual framework, encoded in C-ODO, the Collaborative Ontology Design Ontology, and briefly describe the ontologies and modules that have been reused in C-ODO. In Section 4, C-ODO as a language for collaborative ontology design, C-ODO is detailed in terms of its rationale, according to which design-related entities are classified either as descriptions or as situations and organized in a number of layers: the project layer, the workflow layer, the argumentation layer, the rationale layer, and the pattern layer. All concepts, entities, and other relevant aspects are presented in three formats: OWL(DL), UML-style diagrams, natural language. In Section 5, a minimal schedule of the tasks in the next deliverable of T2.1 is outlined.

Table of contents

List of figures

List of tables

1. Introduction

1.1 The place of design and collaboration in the NeOn global vision

A prototypical use case for ontology design in NeOn can be the following, inspired by the WP7 FAO fishery case study:

- *Jeff* is an executive at the *Blue Turtle* marketing strategy consulting firm, and needs to build an ontology to organize/filter information about salmon commodities. He needs to monitor knowledge from experts in fisheries, fish marketing, food safety, and potential consumers
- *Jeff* knows of existing FAO thesauri and fact sheets, food safety models, markets data, folksonomies and textual data from fish consumers forums, wikis, blogs, etc. He wants to reuse them
- *Blue Turtle* has an ISO9001 workflow for knowledge management, including evaluation and argumentation protocols for collaborative work
- *Blue Turtle* needs to deliver annotions of data in all official FAO languages (English, Arabic, French, Chinese, Spanish)
- *Jeff* wants to semi-automatize the creation of the backbone of the ontology by leveraging on best practices and design patterns

The use case prototype has been devised as a thought experiment that sums up the requirements, desiderata, and expectations that have been collected for NeOn WP2 research. The expected support can be articulated as follows:

- **Design** in NeOn
    - Support to reuse existing ontologies, design patterns, etc., or to reengineer thesauri, lexicons, folksonomies, database schemas, knowledge from corpora (cf. tasks T2.2, T2.5)
    - Support to make an ontology functional to a given task through appropriate evaluation and selection (cf. tasks T2.2, T2.5)
- **Collaboration** in NeOn
    - Support to discuss and get consensus on an ontology element and its rationale (cf. task T2.3)
    - Support to make an ontology usable in a multilingual environment or network (cf. task T2.4)

As usual in these cases, we first addressed these four support requests by assessing the state-of-art and acquiring (preliminary) requirements. This made us realize how the multifaceted aspects of ontology design need a common conceptual framework, in which the functionalities required (or already performed) by designers during their interaction in real ontology projects can be modelled. This framework would support the understanding of two important aspects of WP2's subject matter: the specification of functionalities in order to implement WP2 tools, and the integration of such functionalities in collaborative ontology design.

We believe that these two issues can only be clarified by describing actual social interactions, beyond the information flow pertaining to interactions, which has already entered a computational process. For example, we are interested in the process that makes designers decide on a given design solution, rather than in the computational process underlying the exchange of information between designers through, for instance, email or chat messages. The second process needs to

be supported by a robust computational platform for ontology design, but a good design tool should primarily support the first, social process. It is quite obvious that establishing if email or chat metaphors are adequate to decision making should be based on the description and evaluation of typical designers' interaction.

Task T2.1 is therefore dedicated to the development of a common conceptual framework for collaborative ontology design, and this deliverable is the first step towards it.

This deliverable establishes collaborative ontology design as a research agenda, and abstracts out from the current state of art on collaborative ontology design-related functionalities, methods, and tools. The main upshot of the deliverable is C-ODO, the Collaborative Ontology Design Ontology: a formal framework that can be used as a requirement language for the *social level* of ontology design. For example, we expect to model a use case with a complexity at least comparable to the *Blue Turtle* use case, and to use that model to create a *semantically native* web-based work environment that reflects the requirement model. The specification and implementation of tools that support that work environment will be the subject of the forthcoming deliverable D2.3.1, while here we concentrate on the formal framework alone.

We assume that any specification at the computational level should reflect the social requirements of ontology design and do it in one of two ways: either by *substituting* social-level tasks with computational tasks or by *assisting* social-level tasks, specified as *proxies* within a workflow of computational tasks. For example, suppose *Blue Turtle* has a sophisticated (social-level) method for ontology evaluation: from our perspective, it can be formally described and, if the evaluation is limited to the structural aspects of an ontology, most tasks can be accomplished by an implemented algorithm, hence substituting the social-level tasks. On the other hand, if evaluation at some point requires an active decision role played by a human agent, that role can only be *proxied* by the tool, which results to be just an assistant to the social tasks.

Being clear about what (and how) social-level methods or tasks are substituted, and what methods, roles or tasks are proxied can significantly improve the semantic interoperability between tools and social practices, as made clear since a long time in requirements engineering on experimental basis [Gog93, JRS97]. In particular, our distinction between social and computational tasks maps the first three worlds from [JRS97] (domain and usage worlds are mapped to social-level, and system world to computational world). The advantages of meta-modelling and formal representation of these tasks are recognized and motivated at least since early nineties [MBJK90, Poh93]. Recently, a renovated attention is spreading out to formalizing requirements engineering and application dependability in the context of the Semantic Web (e.g. [DS06]).

Expected relations of D2.1.1 to other work in NeOn include: providing WP5 with a formal vocabulary for describing the functionalities and methods involved in ontology engineering; modelling and matching requirements coming from use cases in WP7-WP8; interlacing C-ODO and OMV (jointly with WP1) towards a forthcoming ontology *meta-level* (not only metadata) open initiative. Moreover, several design aspects raised in D2.1.1 by C-ODO and the state of art review are intertwined with WP4 tasks. Finally, the analysis/specification/implementation cycle in NeOn can be informed by the formal analysis available with C-ODO, and this possibility clearly calls for collaboration with WP6.

The deliverable is organized as follows. In the remainder of this Section 1 we introduce the basic organization of the framework, what notions are treated, and how they are related to the use cases questions raised above. In Section 2 we provide a state of art about definitions, methods and tools for collaboration in knowledge-intensive work and in ontology design, together with general motivations for our proposal. In Section 3 we give an overview of the conceptual framework encoded in C-ODO, the Collaborative Ontology Design Ontology, and briefly describe the ontologies and modules that have been reused in C-ODO. In Section 4 C-ODO is detailed in terms of its rationale, according to which design-related entities are classified either as *descriptions* or as *situations* and organized in a number of layers: the *project* layer, the *workflow* layer, the

argumentation layer, the *rationale* layer, and the *pattern* layer. All concepts, entities, and other relevant aspects are presented in three formats: OWL(DL), UML-style diagrams, natural language.

1.2 Informal description of our approach

A unifying framework for describing ontology design should be general enough to encompass all the approaches that address collaboration and ontology design, and should also be practical enough to be implemented without creating unnecessary complexity in local solutions, models, and tools. Moreover, it should not be a particular methodology (the suggestion of specific methodologies is a task carried out in NeOn WP5), but should provide enough expressivity to describe different methods or aggregations of methods.

We consider ontology design in terms of its *objective, scope, components,* and *support*. In the following we provide summary definitions of these terms. This should give the reader a general idea of how this deliverable sets out the treatment of the problems at hand.

1.2.1 Basic assumptions and notions

The **objective** of ontology design is to help solving the problem of making choices from the (potentially infinite) *choice space* offered by the used *logical language* and available *vocabulary*. Formulating an objective helps getting started with designing an ontology. In analogy with the 'blank page effect' of writers, there exists a 'blank model effect', which needs to be dealt with in terms of the objective of the model.

The **scope** of (networked) ontology design is related to establishing *what we want to describe the design of*. In principle, we could describe the design of any kind of data, process, or resource followed, used or generated during the lifecycle of ontologies over the semantic web: classes, individuals, annotations, email discussions, handbooks, etc. Although our proposal is in principle general and robust enough to support the design of all of these kinds of data, the focus of this deliverable is on ontology design only.[1] Please note that because of the networked perspective we take here, design is not to be intended as limited to creation time, i.e. to an *initial* phase of an ontology lifecycle, but as an aspect of the entire ontology lifecycle.

The **components** of ontology design are supposed to characterize the objective and the scope of ontology design. Such components need to be considered from two perspectives. On the one hand, we should be able to determine *what entities should such components be*. On the other hand, we should be able to determine *how to represent such entities*. Listed below are the main types of entities selected so far:

[1] The design of e.g. a workflow or a project can be treated similarly, but goes beyond the scope of this deliverable, which is to characterize the choices made within ontologies and ontology elements.

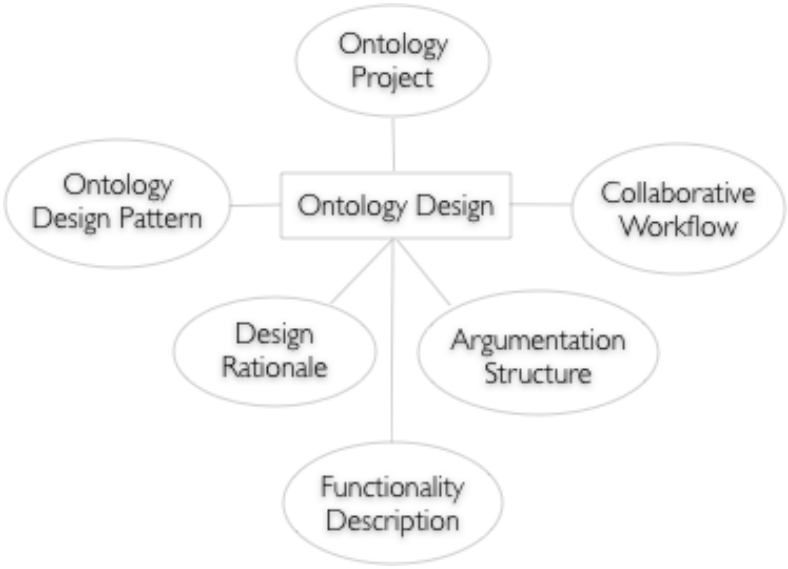

Figure 1: C-ODO's six layers

- **Network of ontologies**: the semantics of the relations between ontologies. Please note that because of the networked perspective we take here, design is not to be intended as limited to creation time, i.e. to an *initial* phase of an ontology lifecycle, but as an aspect of the entire ontology lifecycle (sections 3.2.5, 4.5).
- **Ontology element**: an (identified) element of an ontology, like a concept, a relation, an instance, an axiom, etc. (sections 3.1, 4.4.3).
- **Knowledge resource**: any piece of knowledge that is used while working on the design of an ontology, including modules, workspaces, sources, libraries, networks, etc. (sections 3.1, 4.1, 4.2, 4.4.3).
- **Ontology design rationale**: the reasons why an ontology is designed the way it is. Reasons can be grounded on *content, task, or sustainability* data (section 4.4). The application of ontology design rationales typically produces a choice space for a set of ontology elements.
- **Ontology project**: a project having the goal of influencing the lifecycle of a networked ontology (section 4.1).
- **Epistemic workflow**: a generalization over the possible relations holding between two ontology elements, as they are created, discussed, used or modified by ontology designers (section 4.2).
- **Collaborative workflow**: a special case of epistemic workflow, which is characterized by the ultimate goal of designing networked ontologies, and by specific relations between designers, ontology elements, and collaborative tasks (section 4.2).
- **Argumentation**: a structure for discussing possible design solutions, based on rationales and dialectic rules (section 4.3).
- **Design solution**: a state of an ontology or a part of it at time *t* (section 4.5).
- **Design making**: a situation in which ontology design rationales are implemented in order to obtain certain design solutions. Design making is unfolded by executing design operations that accomplish a functionality by following a method (section 4.4).

- **Choice space**: a space of design solutions allowed by an ontology design rationale on a set of ontology elements (section 4.4).
- **Design pattern**: a configuration of ontology elements that is relevant from the logical, architectural, or conceptual viewpoint (section 4.5).
- **Functionality**: a task to be accomplished by a design operation according to a method (sections 4.1, 4.2, 5).

As mentioned above, a choice is also required on *how to represent* these components. To do so we distinguish between two levels of representation:

1. **Social level**: a 'social view' on an ontology development project. At this level components are characterized in terms of what happens in the real world when a person, a group of people, or a community decide to build an ontology or an ontology network in a collaborative fashion. The social level allows to describe the domain of research through the components and to provide developers of a NeOn-compliant platform with a requirement analysis. This level refers to the first two worlds (domain and usage) of requirements engineering in [JRS97].
2. **System level**: a 'system view' on an ontology development project. At this level components are represented in terms of the methods and the techniques that provide (possible) solutions for supporting what is described at the social level. Such methods and techniques are the base-models for the design and implementation of a NeOn-platform. This level refers to the third world (system) of requirements engineering in [JRS97].

Finally, **support** consists of the functionalities that (are needed to) support collaborative ontology design. The present list of functionalities includes evaluation, selection, re-engineering, learning, upgrading database content, mapping, collaborative workflow, argumentation, provenance, data annotation, social network analysis, lexical domains, ontology localization, multilingual ontology integration.

1.2.2 Preliminary intuition of the support required in the scenario from 1.1

We provide here informal comments to the support requirements from section 1.1:

1. *How to reuse existing ontologies, design patterns, etc., or to reengineer thesauri, lexicons, folksonomies, database schemas, knowledge from corpora*

   When using logical languages like OWL or when reusing existing ontologies, modules, and folksonomies, the choice space is huge and cognitively intractable. Useful solutions include e.g. pre-compiled patterns or best practices, which help making explicit the design rationales of logical and content encoding. For example, it could be more effective in a team to carry out discussions based on a shared understanding of the distinctions assumed by the members. An example in logic: sharing a practice on how to represent classes as values or n-ary relations greatly speeds up the process of getting consensus on an ontology. As a content example, agreeing on a distinction between events and objects or agreeing on the same meaning of a part-of relation (say, on its transitivity) could be key to make progress on a conceptual conflict between two ontology designers. In these cases, by representing the choice space, we can then represent the design rationales of logical modelling and content creation.

2. *How to make an ontology functional to a given task, through appropriate evaluation and selection?*

   The choice space can be reduced by making explicit tasks or competency questions that an ontology should accomplish or help answering. Useful solutions include e.g. pre-compiled use cases, unit tests, etc. For example, agreeing on a typical use case, or matching an ontology against a test knowledge base makes explicit the required or existing design rationales that

depend on task, thus further reducing the choice space. The lower layers of C-ODO structure help expressing requirements for support on design rationales and making, design pattern schemas, and design solutions from choice spaces. Furthermore, the integration of C-ODO with OMV (a metadata vocabulary for ontologies) [HSHP05], and oQual (a metamodel for ontology evaluation and selection) [GCCL06a] makes it easier describing collaborative ontology design against evaluation and selection of ontologies and reusable components.

3. *How to discuss and possibly get consensus on an ontology element and its rationale?*

   Controlling design on the logical, content, and task-oriented aspects is not enough, because the designers in a team should also be enabled to discuss those choices explicitly, and to possibly get consensus. Consensus reaching and team collaboration requires explicit argumentation structures and role-task distribution within an appropriate workflow. The upper layers of C-ODO structure help expressing requirements for support on argumentation, collaborative workflow enactment, and ontology project management.

4. *How to make an ontology usable in a multilingual environment or network?*

   Finally, the lexical encoding of ontologies is crucial for collaborative design, and appropriate support must be provided. Expressing requirements for support on multilingual ontology design is allowed by the classes on the lexical expression of ontology elements in C-ODO.

Having a formal framework in place (C-ODO), the *resources* (functionalities, models, methods, and tools) can be represented and discussed with reference to their place within collaborative ontology design aspects. For example, some resources can be considered as *solutions for the (software) implementation* of a functionality, others are *solutions for the (social) achievement* of a functionality, others are just *representations of a possible solution*, and still others are neither solutions nor representations of solutions, but *knowledge resources* to be employed during collaborative ontology design activities.

2. Literature on collaboration and ontology design

2.1 Introduction

This section presents overviews of the literature on collaboration in cooperative knowledge communities and on ontology design.

2.2 Collaboration

We here present an overview of the literature on collaboration in cooperative knowledge communities, with a focus on requirements (Sec. 2.2.1), tools (Sec.2.2.2) and their matching (Sec. 2.2.3). Going through the following examples of work done on issues of computer-assisted collaboration allows to get acquainted with the terminology used in the field. In addition, this overview makes explicit *two issues* that have determined the direction of our work on collaboration throughout this deliverable:

1. *The notion of collaboration is not univocal. Moreover, despite having been widely treated in the literature, none of the existing treatments provides a sufficiently general definition for it*. The tables below, 1 (Example of comparison between collaboration requirements) and 2 (Example of comparison between collaboration tools), provide a synthetic view on this claim. As a matter of fact, each proposal (i.e. each column) defines specific requirements or functionalities for given groups of users. But the proposals do not describe their intended groups of users, of requirements or of functionalities in terms of a common conceptualisation. In other words, the headings of each row in the two tables are by no means an ontology, but simple abstractions used here for the purpose of the present discussion. Furthermore even by sticking to such simple abstractions, the comparison between the different proposals does not become necessarily easier. This is a consequence of the fact that the each proposal evaluates the same row in a different way or at a different level of detail. For instance, group definition in Table 1 is achieved either by enabling mutual recognition, or by defining group boundaries, or by promoting continuity; but it remains unclear how these three different ways of coping with group definition relate to one another. Furthermore, again in Table
2. 1, requirements on information management and group management in DILIGENT are much more detailed than, for instance, in the proposal of [Axe84]; and the consequence of this are the gaps in the first and the second column. The situation just described makes it impossible to adopt any of the existing proposals as a basis for the definition of a language to talk about collaborative ontology design. In order to fill this gap in the literature, and thus provide NeOn with a general enough understanding of the notion of collaborative ontology design, we introduce in Sections 3 and 4 the Collaborative Ontology Design Ontology (C-ODO). C-ODO is very much based on existing work, especially for what concerns the terminology and the concepts we adopt. But in terms of generality C-ODO's goes beyond all existing proposals.
3. *The lack of generality of existing proposals mainly is a consequence of the social-technical gap* between social requirements and technical feasibility: "the divide between what we know we *must* support socially and what we *can* support technically" [ACK01]. Most existing proposals approach this divide from the technical side. None of them attempts at providing a conceptual framework of the collaborative use of existing or forthcoming technology. C-ODO is a first step towards such generalization, in terms of the specification of functionalities and of their integration.

Collaboration requirements	Evolution of Cooperation [Axe84]	Evolution of Institutions [Ost90]	How virtual communities work [God94]	DILIGENT [Vra06], [Tem06]
Group definition	Enable mutual recognition	Define group boundaries	Promote continuity	
Information management	Circulate information	-	Use good discussion software	Communicate changes in the ontology
	-	-	Provide institutional memory	Integrate control and argumentation support
	-	-	-	Graphic representations of ontology
Group management	Arrange meetings	Rule use of collective goods	Users resolve their own disputes	Use argumentation
	-	Monitor members	Confront users with a crisis	Use a clear methodology
		Sanction members		Use clear decisions processes

Table 1: Example of comparison between collaboration requirements

Collaboration tools	OBOS [DWM01]	HACM (Compendium-based) [SMD02]	Claimspotter [SSm04]
Types of users	Ontologists	Ontologist	Scholars
	Domain experts	Scientists	-
	Business analysts	-	-
Functionalities	Ontologies development	Structure collaborative sense making	Support annotation of scholarly documents
	Ontologies maintenance	Aid group memory	Create triples (source destination, relation between them (e.g. source 'proves', destination)
	Multiple access	Dialog mapping	-
	Discussion rooms	Direct formalization of conceptual proposals	-
	-	Direct display of proposals on screen	-

Table 2: Example of comparison between collaboration tools

2.2.1 Requirements for Collaboration

A substantial volume of research has gone into exploring user requirements in collaborative activities. [Kol96] gives a brief survey of the main studies (at the time when the World Wide Web was emerging) on principles that seems to underlie successful cooperative communities. The aim of the paper was to understand if such principles and best practices (or some of them) were applicable to building successful cooperative online communities. The paper is a bit dated, nevertheless it is worth to report (part of) the list of design principles the author identified, which are still basic good practices for online communities creation and management.

Requirements for the possibility of cooperation These were taken from [Axe84]:

- Arrange that individuals will meet each other again.
- They must be able to recognize each other.
- They must have information about how the other has behaved until now.

Design principles of successful communities These were taken from [Ost90]:

- Group boundaries are clearly defined.
- Rules governing the use of collective goods are well matched to local needs and conditions.
- Most individuals affected by these rules can participate in modifying the rules.
- The right of community members to devise their own rules is respected by external authorities.
- A system for monitoring members' behavior exists; this monitoring is undertaken by the community members themselves.
- A graduated system of sanctions is used.
- Community members have access to low-cost conflict resolution mechanisms.

Principles for making virtual communities work These were inspired by [God94]:

- Use software that promotes good discussion.
- Don't impose a length limitation on postings.
- Front-load your system with talkative, diverse people.
- Let the users resolve their own disputes.
- Provide institutional memory.
- Promote continuity.
- Be host to a particular interest group.
- Provide places for children.
- Confront the users with a crisis.

More recently, in the course of the case studies for the DILIGENT methodology, several requirements were identified with regard to an efficient support of a collaborative workflow. For example, if one wants to create an ontology related to professional knowledge, one should form a team of domain experts and ontology engineers.

Requirements in DILIGENT [Vra06] [Tem06]

- Engineering tools should have support for communicating changes in the collaboratively developed ontology.

- All other required tools like version control and argumentation support should be strongly integrated into the engineering tool.
- It is important to have a graphical visualization of the ontology.
- Use a concrete methodology for the collaborative development process to clarify the objectives and to have a list of things to do next. The methodology should cover the whole ontology lifecycle including the maintenance phase and not only the initial creation of the ontology. With this respect, it proved to be useful to have a well-defined process for feeding back change requests and to document the argumentation process which led to certain design decisions.
- Find good evaluation measures which show whether one reached the goals which were originally set for the ontology.
- Use an argumentation framework. Discussions are often inefficient and time consuming if a clear structure is missing. For example, it is useful to restrict the users to certain argument types so that they didn't get lost in the discussion.
- Use a clear decision process has to be defined which of the proposed solutions should be included into the ontology. Many of the requirements for ontology engineering tools also apply for the argumentation tool.
- The user needs a possibility to monitor a discussion so that she/he is automatically informed of changes to discussions of interest.
- The argumentation tool should be integrated with the ontology engineering tool so that one can access the argumentation data from the engineering tool and vice versa.

Another useful approach to requirements for collaboration is provided by [NCLM06] where several scenarios for collaborative ontology development are identified. Subsequently, requirements for an efficient support of these collaborative activities are derived. Many of those requirements are with regard to an efficient version management and control system.

Requirements for efficient version management and control system [NCLM06]

- Use software that promotes good discussion.
- It is very important for the users that they can attach annotations to their changes which explain the rationale and/or which refer to citations and documents on which the change is based.
- Do not compute textual differences between e.g. two OWL ontologies but that instead a list of changed ontology elements, including information about e.g. which concepts were split or merged, an information typically not available if the changes were computed based on textual differences.
- The description of changes is needed in such a granularity that it is possible to go back to earlier versions of an ontology at any time.
- Having fine grained access rights. It is especially not sufficient to define the access on the level of an ontology. Instead it should be possible to define it on the level of ontology elements. This helps to avoid conflicts between the different versions of editors. Nevertheless, before checking in a new version it is necessary to identify direct and indirect conflicts between two versions. Indirect conflicts may e.g. occur in subclasses that depend on one of the changed classes. In case that a conflict occurs, a kind of negotiation process between editors is needed which helps to resolve the conflict. In some collaborative scenarios, there exists a central authority or curator, which decides which local changes of editors will be included in the shared version of an ontology. In this case, the curator needs the possibility to accept a whole set of changes. This set of changes may be identified structurally or based on who performed the change and when. Otherwise, accepting each single change separately

would be very tedious. In this scenario, the automatic detection of conflicts as well as the annotations made by the authors are very useful for the curator as they explain why a change was necessary.

2.2.2 Tools for Collaboration Support

Ontology Builder and Ontology Server (OBOS) An application suite proposed in [DWM01] and developed for supporting the creation and maintenance of ontologies used in e-commerce and B2B applications. There is not a precise definition of collaboration, the issue is approached focusing mainly on tool functional requirements identification. OBOS has been built with the aim of supporting a distributed and collaborative team of users (ontologists, domain experts, and business analysts) developing and maintaining shared ontologies. Basing on an informal evaluation of four existing tools (i.e., Ontolingua/Chimaera, Protégé/PROMPT, OntoWeb/Tadzebao, Ontosaurus/Loom), a set of requirements is identified. Among them the following are very important for collaborative ontology creation: scalability, availability, reliability, performance, ease of use, distributed multi-user collaboration support, security management, difference and merging support, internationalization, and versioning. OBOS uses a frame-based representation based on OKBC knowledge model and its implementation is based on J2EE. The tool provides a collaborative environment for the development and maintenance of shared ontologies. Multiple access is managed using a role-based policy, and users are provided with discussion rooms where they can communicate about their work on the ontology/ies. OBOS implements a pessimistic locking strategy for editing and changes to the ontologies are immediatly notified so as the user can refresh the information. Multilinguality is supported by means of so called locales, versioning is not supported. The tool resulted to be sufficiently easy to use (as claimed by the authors), nevertheless the different types of users (ontologist, domain expert, and business analyst) are not provided with specific interfaces and/or methods.

Hypertext-Augmented Collaborative Modelling (HACM) An application proposed in [SMD02] is based on a Compendium approach. Within the Advanced Knowledge Technology (AKT) consortium the AKTive Portal was designed to be a next generation portal infrastructure that supports the capture, indexing, dissemination and querying of information. The first application of the portal was to the AKT project itself. Mifflin, a hypertext tool for Compendium, was used to facilitate ontology-based scientific knowledge creation and management in collaborative settings and, interestingly, the case-studies were AKT meetings. Mifflin's main functions in the project were to provide:

1. Structure to collaborative sense making;
2. The rationale for an ontology engineer when implementing the agreed specification;
3. A memory aid in and between meetings for both the group and for the group coordinator;
4. Multiple on screen visualizations of both the existing ontology structure and of the ongoing discussion about it.

Mifflin succeeded in supporting the collaborative creation of an ontology of scientific knowledge, mainly in terms of:

1. Dialog mapping,
2. Direct formalization of conceptual proposals,
3. Direct display of proposals on screen,
4. Compatibility with existing software tools.

The achievement of these four results was not cost-free, though. Mifflin imposed on (even expert) users the development of some literacy and, at the beginning, some cognitive

overhead.

Claimspotter The application presented in [SSM04] is an open architecture based on the ScholOnto ontology. Claimspotter supports the semiformal (collaborative) annotation of scholarly documents. It is based on a simple paradigm: triples. The text of a document is represented by couples of concepts (source and destination concept) plus a relation between them (for instance: 'is an example of', 'is enabled by', 'proves', 'supports', 'is similar to', etc.). Such triples allow building a network of claims about the internal structure of the document, or about its relations with other documents. The resulting network, or parts of it, can be shared and incremented by different users over time. The final result is a commentary to the original text, which is usually dialectic, as the network can host logically contradictory claims about the contents of the document.

Claimspotter supports the creation of triples by means of suggestions given to the user. On the one hand, suggestions have to do with the structure of the document and the scientific rhetoric the keeps it together. Two main families of rhetorical roles are considered: what in the document refers to the work being described (background, aim, textual structure) and what refers to the the work of other researchers (contrast, basis). On the other hand, suggestions have to do with so-called information bricks, i.e. parts of the document, which may be used as concepts in the network (keywords, the instances of ScholOnto relations found in the text - i.e. verbal expressions -, cited documents).

Claimspotter's architecture as well as the presentation of the suggestions is highly modular. A toolbar gathers all different suggestions on the following aspects: the concepts made by the current annotator, the instances of ScholOnto relations found in the text, the documents important sentences (where importance is defined in terms of keyword-matching with title, headers, abstract), the document's rhetorically-consistent zones, the sentences matching a particular user-defined query expressions.

A very limited user study has been done for evaluation of Claimspotter. This has revealed two main points. On the hand, an annotation system should be very flexible with respect to the quantity and the quality of suggestions provided to the user. Users want to be able to switch back and forth from a very structured configuration (where to get support and inspiration from what other annotators have done) to a lightweight configuration (where to "think outside the box"). On the other, it has become clear that gaining expertise with the system corresponds for annotators to move from a "concepts to relations approach" (which tends to produce idiosyncratic networks) to a "relations to concepts approach" (which facilitates standardization).

Ontology of the Academic Field This was presented in [BSD05] material that is very much in line with [SER04]. It generalizes that approach (in terms of an, rather than of scholarly comment only) and it makes one step towards automation (in terms of a number of functionalities that allow to derive knowledge from a model based on the ontology).

The Ontology of the Academic Field comprises three main components:

1. the Community of Practice (with concepts, attributes and relations like Publication, Title, author-of, researcher-at, etc.);
2. the Lexicon (with concepts, attributes and relations like Lexical-Term, Gloss, broader-term, etc.);
3. the Argumentative Discourse (with concepts, attributes and relations like Statement, Question, Issue, Premise, Conclusion, Postulates, supports, coheres).

Services are provided for:

1. The usual bibliographic database functions provided by tools like CiteSeer or Google Scholar;
2. Finding key statements made by an author on a particular issue;

3. Assisting navigation around a complex argumentation network, which renders an ontology as an interactive map;
4. Flexible visualization of the network;
5. Inferences on the network, by creating paths, like for instance, (in)coherence paths that connect the opinions of a given author with a scholarly position that is typical of the reference field.

ClaiMaker An application introduced in [MS06] designed to represent discourse in a semiotic way within the scholarly domain. More generally, the paper discusses the representational requirements for collaborative systems that support sensemaking and argumentation over contested topics. Sensemaking is intended as expressing and contesting explicit, possibly competing views of the world. Supporting sensemaking therefore means supporting a way of annotating different interpretations of the same object or issue. This is what ClaiMaker does, with a theoretical backbone consisting of semiotic (in a Saussurian fashion) and coherence relations, as in Mann and Thompson's Rhetorical Structure Theory (RST) [MT88].

ClaiMaker is a hypertext system that makes use of constrained base relational classes, but imposing no constraints on how such classes are rendered, or on how nodes are expressed/classified. ClaiMaker's ontology allows users to establish as many referential relations between concepts (e.g. the summary message of a document) and sources (e.g. a document) and also connective relations between sources. Claims of the first kind are called "primary claims", whereas those of the second type are called "secondary claims".

1. *Primary claim*: users can associate documents with concepts: this consists in establishing a referential relation between a concept and a referent. In other words: a primary claim is the creation of a sign (=the concept) that refers to a particular referent (=the document) in the virtual reality (=the ClaiMaker repository), in some respect (=a context). From an ontological point of view, it is interesting to note that concepts linked to sources can (optionally) be classified. The classification is not rigid, however, so that the same concepts can be assigned different classes by two different persons, or even by the same one in different contexts. Like in natural language, in ClaiMaker meaning is continually negotiated by means of establishing referential relations between referents and concepts, and by means of defining concepts according to different classes (different from ontology-based systems).
2. *Secondary claim*: A secondary claim establishes a discourse connection between two concepts. The authors borrow plenty of terminology and insights from linguistic theories, such as RST and Sanders et al.'s theory of connectives [SSN93] to model what they term an "upper level discourse relations ontology". Grounding on Sanders et al's approach, they treat coherence relations as psychological constructs and take a small number of cognitively basic concepts. The relational scheme is based on four parameters (Cognitive Coherence Relations [CCR]): Basic operation [additive, causal], Source of Coherence [semantic, pragmatic], Order [basic, non-basic], Polarity [positive, negative]

A relational hierarchy is then derived from these four parameters. By also incorporating insights from Louwerse's (2001) description of coherence relations [LOU01], the authors obtain the final ClaiMaker's relational ontology, which can be used for annotation of secondary claims. Since it is based on cognitive primitives, it has the main advantage of being applicable in different disciplines and domains.

Co-OPR project [BCCV06] presents the integration of two existing tools (i.e., Compendium, and I-X) for the, the simulation of a personnel recovery mission. The experiment presented deals with decision-making support for a team collaborating on the same mission. In particular, Compendium has been used in order to support the collaboration between members of the

team who were geographically distributed; I-X has been involved as a tool supporting a team whose members were physically in the same place. The paper underlines the effectiveness and usability of the two tools when used together by giving a very pragmatic evaluation. The focus is mainly on the usability and utility of provided functionalities.

2.2.3 Matching requirements and tools

Tools that support collaboration are obviously developed on the basis of user requirements. Some valuable insights come from comparing such requirements and available tools, as to identify not only the technical gaps, but rather determine which gaps can be bridged by advancing technology and which are instead unavoidable (i.e. which requirements are unsupportable).

An important contribution is Mark Ackerman's work on the gap existing between social requirements and technical feasibility [ACK01]. What Ackerman terms "the *social-technical gap*" is "the divide between what we know we *must* support socially and what we *can* support technically", and is likely to be the highest challenge of Computer-Supported Cooperative Work (CSCW). Within NeOn, it is not only relevant the description of social requirements (in collective work), but also the attention to what kind of support is difficult to achieve technically, and therefore a definition of an upper bound with respect to tool development for supporting collaborative activities.

The following are some social aspects of communication that need to be considered when building any tool that supports collaborative activities

- *Nuances of social activity*: social activities are fine-grained and flexible, thus making systems technically difficult to build.
- *Multiplicity and Diversity of Goals*: members of a given organisation might have different goals and different organisations may not have shared goals, knowledge, and meanings. Conflict is as important as cooperation in issue resolution. Meanings must be negotiated, for example (see also [MS06] about the importance of building tools that support negotiation of "sense making", such as ClaiMaker [MS06]).
- *Exceptions* in work processes are normal. And roles can often be informal and fluid CSCW approaches to workflow should deal with exceptions and fluidity.
- *Visibility of communication exchanges and information* facilitates learning but might inhibit for fear of criticism. Ways must be found to manage the trade-offs in sharing.
- *Norms for using CSCW*: they are set (negotiated) by the users and can change while using the system. The system must therefore allow for renegotiation, changes and flexibility (see again [MS06]).
- *Critical Mass*: with an insufficient number of users, people will not use a CSCW system.
- *Adaptation*: people adapt their systems to their needs, so not everything can be foreseen when developing a system; however, systems are often too rigid to allow for such changes.
- *Incentives*: using a tool might be time consuming, also from a learning-to-use-it point of view. So it must be rewarding, i.e. benefits must be evident.

Additionally, [PM04] claim how one of the main failures of human-computer interaction (HCI) is the treatment of *turn taking*. Experiments are presented that show that HCI systems are not equipped with means for dealing with natural turn-taking issues, such as pauses, overlaps, and similar behaviour. Although this applies to human-machine interaction, in the spoken dialogue domain there might be similar problems in collaborative activities between humans conducted over the Web, especially if done in a synchronous manner (see also [Cha01]).

In the specifics of collaboration towards ontology development, [Lu03] is a source of interesting points with respect to requirements and tool support. According to [Lu03], since modern ontologies are characterized by their huge size and high complexity, ontology engineering is to be considered an inherently collaborative activity, involving the effort of many domain experts and software developers which are often not co-located. This is especially prominent when not only the initial

design stage, but the whole ontology life cycle is considered. Throughout this work, 'collaboration' seems to be defined as a reiterated process, the output of which, at each of the involved stages, is the obtaining of a 'convergence of views'.

Based on a survey of five authoring tools (Ontolingua Server, OntoEdit, APECKS, CO_4, and Protégé-2000), which were widely used at the time of the inquiry (updated in 2000), the conclusion is reached that collaborative ontology development was – again, at that time – far from being well supported by said tools.

The identified inefficiencies with respect to collaboration support were the following:

- Coordinated group work (e.g. collaborative editing, discussion or annotation) was not well supported, mainly because the systems lack functions for keeping developers informed of each other's activities (compare, by contrast, with current wikis)
- Contextual communication support was not considered in these tools, i.e. no way was provided for easily keeping track of a whole coherent discussion, which is e.g. scattered in many people's mailboxes (compare, by contrast, with Compendium, Claimspotter, and ClaimMaker below)

Since collaborative work across distance in software engineering and ontology engineering share many similar characteristics, three dimensions of collaborative ontology engineering are identified based on documents and experiences in the first field:

- *Distance and communication*: Co-located team members communicate informally anytime during the work day, while this cannot happen to geographically distributed team members (A study at Carnegie Mellon University showed that the rate at which scientists collaborated spontaneously with one another was a function of distance between offices). Informal and unplanned communication has been proved to have a direct impact on development processes, in particular on
- *Coordination* ("the act of integrating each task with each organizational unit, so each unit contributes to the overall objective"), and
- *Control* ("the process of adhering to project goals, specifications, and standards").

Coordination and control are necessary not only to manage interdependencies within the tasks, but also for the development and maintenance of *shared mental models* (compare with FLE36 on *thought-styles*), which are considered to be the most effective support for explicit coordination in team work. Communication affects shared mental models in two ways: a) during task execution, it refines team members' mental models with contextual cues; b) it keeps the models up-to-date, especially in dynamic or novel situations. It has been proved that weak shared mental models in asynchronous tasks can lead to productivity losses. Designing tools to support and enhance informal communication is then a key step toward bridging the missing link in distributed development work.

- *Documentation and knowledge management*:

Information and knowledge obtained during meetings, email correspondences, and instant messaging need to be captured easily, stored and shared effectively. The distribution of resources and developers in space and time combined with the dynamic evolution of knowledge make the use of tools for knowledge management a necessity. Moreover, the documentation must be kept up-to-date.

- *Version control and change tracking*:

Tools for version control and change history tracking are crucial when development resources are not co-located, in order to make sure that two developers do not work on the same part of the ontology and to avoid the complication of resolving conflicts.

Finally, a range of tools and groupware technologies in the Computer Supported Collaborative Work (CSCW) domain are investigated, in order to determine how they can be used in the ontology

development domain.

- *Experiences in the Global Software Development (GSD) field are examined in order to understand the correlation between distance and collaboration:*
  1. instant messaging techniques (support spontaneous and informal communication)
  2. web portals (support tasks in the area of group knowledge management)
  3. Peer-to-Peer (P2P) network technologies (but poor reliability and security)
- *Report of an experiment where the possibility of adding collaborative support to a knowledge engineering tool based on a P2P network was evaluated*
- *Long term vision: to combine these two fields and create a collaborative ontology engineering environment that provides collaboration support in multiple dimensions:*

As far as coordination in collaborative environments is concerned, interesting suggestions may be provided by the coordination models and workflow patterns presented respectively in [PK91] and [vdAtHK03].

Finally, an approach that attempts an abstraction from local, optimized theories for argumentation is AIF (Argumentation Interchange Format, [WVS05]). It exemplifies the need for a metamodel that enables the comparison between different approaches on a formal ground.

2.3 Ontology Design

We present here an overview of the literature on ontology design. There are on this subject problems of generality that are similar to – or even harder than – the problems about collaboration presented at the beginning of section 2.2. As a matter of fact ontology design cannot be specified unequivocally – for instance, as an OWL class – because the entities that are typically referred by the term design can be multifaceted. In addition, the literature on ontology design as a *subject* is not organic yet. Ontology design *as such* is a very recent research topic – so far the research community has mainly worked on designing ontologies.

What has been produced, though, are analyses of general issues that, on a longer run, will be integral part of ontology design as a subject. Such analyses usually verge on three main issues:

1. ontology evaluation, typically through meta-properties;
2. ontology re-engineering, usually in terms of methodologies for re-engineering;
3. ontology design patterns, this being the most fluid sub-field of research in ontology.

Table 3 provides a synthetic view on an example of comparison between ontology design-related proposals. Similarly to table 1 and 2, the impression here is that ontology design-related proposals have touched so far on various aspects of the design of an ontology (e.g. conceptualisation, core aspects of methodologies for re-engineering ontologies, patterns, etc.) without trying to integrate the field. The Collaborative Ontology Design Ontology (C-ODO) presented in Sections 3 and 4 accounts for this variety and tries to clarify it.

Ontology Design	**Ontoclean [WG01]**	**Methontology [FLG04]**	**Ontology Design Patterns [BBC+ 99], [MHG02], [RR04], [Sva04] [Vra05] [Gan05]**
Evaluation thru metaproperties	Rigidity	-	-
	Identity	-	-
	Unity	-	-

	Dependence	-	-
Core aspects of methodologies for re-engineering ontologies	-	Scope of ontology	-
	-	Coverage of ontology	-
	-	Adaptability of ontology	-
	-	Integration of ontology	-
Interpretation of ontology design patterns	-	-	Combinations of foundational concepts
	-	-	Syntactic serialization of semantic patterns
	-	-	Expert-reasoning basic types and relations

Table 3: Example of comparison between ontology design-related proposals

OntoClean [WG01] has the goal to detect both formal and semantic inconsistencies in the properties defined by an ontology during pre-modelling and modelling stages, i.e. during ontology development. The main function of OntoClean is the formal evaluation of the properties defined in the ontology by means of a predefined ideal taxonomical structure of metaproperties (rigidity, identity, unity, dependency). ODEClean [FLG02] is a plug-in created as a support for application of OntoClean based on a number of functionalities (e.g. establish the evaluation mode, assign meta-properties to concepts, focus on rigid properties, evaluate according to the taxonomic constraints).

Methontology This framework is meant to be used by domain experts and ontology makers who are not familiar with implementation environments. It has the goal to let them build ontologies from scratch [FLG04]. To this end, a number of functions are provided that enable easier intermediate representations of ontologies. Such representations are meant to bridge the gap between how people think about a domain and the languages usually used to define ontologies at the formal level. In other words, Methontology makes it possible to work on ontologies at the knowledge level only, and it does so by supporting functions like the specification of the ontology development process as well as of its life-cycle (based on evolving prototypes); the specification of ontologies at the knowledge level; the multilingual translation that automatically transforms the specification into several target codes. Methontology is well exemplified by the following specification of a process of ontology re-use:

- specifying the requirements the ontology must satisfy in the new application (purpose, language in which it is needed, key aspects that should be modelled, scope – the latter defined through competency questions);
- searching an ontology that covers most of the identified necessities;
- adapting the chosen ontology so that it satisfies the necessities completely;
- integrating the ontology in the system (this may involve language translation).

Ontology Re-engineering is defined as the process of retrieving and transforming a conceptual model of an existing and implemented ontology or informal knowledge resource, into a new, semantically explicit or more complete conceptual model, which is re-implemented. The ontological re-engineering process should be carried out bearing in mind the use of the existing ontology by the system (ontology or software) that reuses it. This process consists basically of three activities:

- reverse engineering where the aim of this activity is to output a possible conceptual model on the basis of the code in which the ontology is implemented.
- restructuring where the aim is to correct and reorganize the knowledge contained in the initial conceptual model, and detect missing knowledge.
- forward engineering where the aim is to output a new implementation of the ontology on the basis of the new conceptual model.

Ontology design patterns Traditional design patterns (for instance in the field of architecture) appear more like a collection of shortcuts and suggestions related to a class of context-bound problems and success stories. In recent work, there seems to be a tendency towards a more formal encoding of design patterns (notably [BBC+ 99], [MHG02], [RR04]. [Sva04] also addresses the issue of ontology design patterns for the Semantic Web, taking a foundational approach. [Vra05] proposes the introduction of macros for OWL ontologies in order to ease the expression of tedious tasks that are often repeated. Macros can be seen as a syntactic serialization of semantic patterns. In [Gan05] the notion of 'Content Ontology Design Patterns' (or 'CODePs') is introduced, and its difference with other sibling notions is discussed. Some examples of 'CODePs' are illustrated, and their usefulness in order to acquire, develop, and refine ontologies from either experts or documents is shown. 'CODePs', e.g., can be exploited as a tool to annotate 'focused' fragments of a reference ontology, i.e. the parts of an ontology containing the types and relations that underly 'expert reasoning' in given fields or communities.

3. Overview and foundations of the Collaborative Ontology Design Ontology

3.1 Introduction

As pointed out in Section 2, the notions of collaboration and of ontology design are not univocal. Moreover, none of the existing treatments of these notions provides a sufficiently general definition for them. This makes it impossible to adopt any of the existing proposals on either collaboration or ontology design as a basis for the definition of a language to talk about collaborative ontology design.

In order to fill this gap in the literature, and thus provide NeOn with a general enough understanding of the notion of collaborative ontology design, we introduce here the Collaborative Ontology Design Ontology (C-ODO). C-ODO formally specifies the components of collaborative ontology design. C-ODO is meant as the formal basis of a language to express *social-level* requirements for tools that support ontology design. C-ODO's main components - as already informally explained in Section 1 - capture the epistemological nature of designing knowledge in general and, in particular, of designing ontologies, i.e. reusable knowledge.

C-ODO is based on a relatively large number of assumptions and ontological commitments, which will be detailed both in the following subsections and in Section 4. Here, though, we want to introduce C-ODO at a glance, and do it in terms of the main modelling choices – we should say: the design rationale – that led our efforts towards one single framework for the notions of collaboration and design. Our work pivoted on three choices: basing the whole framework on a reification mechanism, which is founded on the distinction between *descriptions* and *situations*; breaking down the conceptualisation modelled in the framework into five main layers (*ontology project, collaborative workflow, argumentation, design rationale, ontology design pattern*); reusing as many as possible existing ontologies as foundations for C-ODO.

Reification through Descriptions and Situations We have based C-ODO on the reification vocabulary from the ontology of Descriptions and Situations[2] (*DnS*, [GM03]). Just to give an intuition, the form of reification adopted in our framework makes it possible to talk in the same language both of a generic method[3] to, for instance, design and of the actual (composite) operations that allow to perform that method, like, for instance, (the composition of) 'create a class', 'elicit knowledge from a colleague or an expert', 'model knowledge in the class', 'validate the class', etc. In other words, the domain of interpretation of C-ODO contains the generic method as well as the simple and the composite entities that allow to perform the method. This makes the language more expressive without making it computationally more complex – the usual advantage of reification. On the other hand, *DnS* allows a precise partitioning of the reification domain (partitioning the domain of reified entities is the typical source of confusion in logical reification). The form of reification adopted in *DnS* is based on the distinction between *descriptions* (e.g. a method) and *the situations that satisfy a description* (the operations that allow to perform the method). *Descriptions* (and the *concepts* devised in *descriptions*) are used to *classify entities* within a *situation*. For the purpose of intuition, the distinction between *descriptions* and *situations* can be understood as analogous to the UPML (Unified Problem-solving Method Development Language) paradigm [ML00], in which «classification can be seen as the problem of finding the solution (class) which best explains a certain set of known facts (observables) … according to some criterion». *Descriptions* are (as ontological entities) the counterpart of a set of criteria in

[2] Available at http://www.loa-cnr.it/ontologies/ExtendedDnS.owl

[3] Notice that methods can be logical, cognitive, social, or computational.

UPML-based classification library, while *situations* are the counterpart of a solution in a UPML-based classification library. Furthermore, *situations* are *settings for* a set of entities and their relations, which are classified by the *concepts* devised in the *description*. Therefore, related entities in the *setting of* a *situation* may be considered as observables in a UPML-based classification library. Since UPML assumes execution as a computational task, solutions are the outcome of executions. On the other hand, DnS makes no commitment on how classification (i.e. satisfaction) of a solution is executed, leaving it to the particular reasoning resources that are available in a context. Therefore, the UPML classification of observables into a solution (class) is generalized in DnS as finding the class of situations that best satisfies a description according to the available (or required) reasoning procedure.

Six Main Layers of Ontology Design As shown in Fig. 1, we have distinguished six *layers* that fraction the scope of design into ordered components: *ontology project, collaborative workflow, argumentation, design rationale, functionality*, and *ontology design pattern*. Fig. 2 shows how these six layers work in terms of the distinction between *descriptions* (white balloons) and *situations* (grey balloons). Take for instance as a starting point *ontology design pattern*, which is a description that includes the roles, tasks, and parameters for designing a certain type of ontology The situational counterpart of *ontology design patterns* are *design solutions*. These have a *design rationale*, which is a *description* and which has been made explicit during an *argumentation situation*, this last being the situational counterpart of *argumentation*. *Argumentation situations* typically find place during *collaborative workflow enactments* (*situation*) that follow some *collaborative workflow* (*description*). Such workflows are part of an *ontology project* (*description*), either *a priori*, i.e. as proper parts of the *ontology project*, or *a posteriori*, i.e. as parts of the *ontology project execution* (*situation*). Finally, functionality descriptions are also executed as part of design making.

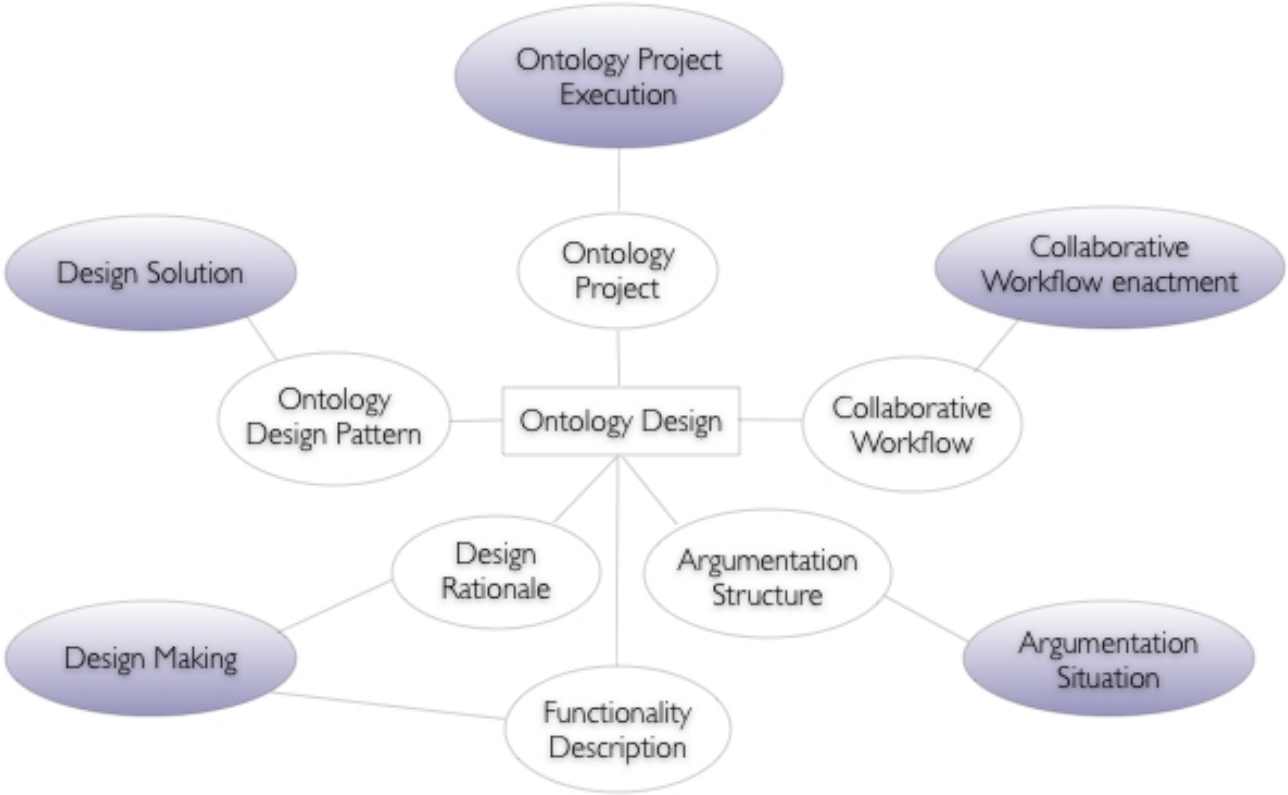

Figure 2: C-ODO's six layers as descriptions and as situations

Reuse Existing Foundations We have built C-ODO on existing ontologies, patterns and definitions. In particular, we have reused the ontology of Descriptions and Situations (DnS)

[GM03] to represent the distinction between *description* and *situation* introduced above. Moreover, we have founded C-ODO's five main layers on the following ontologies: the Plan Ontology (DDPO) [GBCL05], the ontology of Collections and Collectives [BCGL06], the Information Objects ontology (DDIO) [GBCL05], the Open Systems ontology, the Knowledge Content Objects ontology (KCO) [BGMR05], the ontology of Ontology Evaluation, Quality, and Selection (oQual) [GCCL06a, GCCL06b], the Ontology Metadata Vocabulary (OMV) [HRWB06], the definition of Network of Ontologies given in deliverable D1.1.1 "Networked Ontology Model" [HRWB06].

Section 3.2 provides an initial informal presentation of the notions for ontology design that have been formalized in C-ODO. This is meant to give an intuitive overview of the discourse on ontology design that C-ODO's conceptualisation supports. Sections 3.3 informally presents C-ODO's foundations. In both sections, we keep using the notation used in this introduction: we print in italics *notions that are defined in C-ODO or in its foundations*. This will hopefully make references to C-ODO and its foundations clearer while, at the same time, making it easier for reader to go through the informal presentation of the ontology.

3.2 C-ODO's conceptualisation

This section provides an initial informal presentation of the notions that in C-ODO represent collaboration and ontology design. We start from the notion of *ontology project* and show how by using C-ODO's conceptualisation one is able to support a discourse (both in terms of *descriptions* and *situations*) that touches on most salient aspects of collaborative ontology design.

3.2.1 C-ODO's concepts in terms of descriptions

From a descriptive perspective, an ontology project in C-ODO is a *plan* for (collaborative) work on a (networked) ontology. Such a *plan* enables the production of knowledge from knowledge and it is based on five elements:

1. At least one *epistemic workflow*
2. At least one *functionality*
3. At least one *knowledge-resource*
4. At least one *working-knowledge-item*
5. At least one *knowledge-product*

In the following we provide for each of these notions an intuition of the kind of discourse on collaborative ontology design in which it may be used.

Epistemic Workflow This is one of C-ODO's key-concepts, the super-type of *collaborative workflow*. It subsumes other types of workflows, like *interaction* or *usage*, for which the bonds between the agents involved in the workflow are weaker than in the case of collaboration. Just as ontology projects, workflows are *plans*, that are characterized by two facts: having a *knowledge-production goal* as their *main goal* and an *argumentation structure* as their component. Collaborative workflows are typically based on teams, that include *accountable agents*. All pieces of knowledge in a workflow (e.g. *resources*, *working-knowledge-items* etc.) have *knowledge creators*, which may be either *rational agents* or *knowledge collectives*. These last two may, though, execute other tasks or functionalities within an ontology project, thereby playing one of the roles subsumed by the generic *performer* role. If an agent adopts the main goal of the project, then it is an *accountable performer*. Like all plans, ontology projects and epistemic workflows can be expanded or refined, and their most complex functionalities (e.g. evaluation) can be treated as *subplans*. *Argumentation structures* are

descriptions for argumentation primitives, which define at least one *argumentation role*. Argumentation enables the definition of *design rationales*, which in turn are *descriptions* that describe the context assumed for reasoning over the definition of *design solutions*. Consider, for instance, two expert *knowledge creators* that hold two different opinions: expert A wants the class *fishing platform* to be subsumed by the class *fishing vessel*, while expert B wants them disjoint. The reasons behind A's opinion form a design rationale, and so do the reasons behind B's opinion. Finally, *ontology design patterns* are descriptions that describe the intension of an ontology. They may for instance encode logical invariants, like *being a class*, or content invariants, like *being an aquatic organism*.

Functionalities In C-ODO functionalities are tasks defined in some *functionality description* (that can be either generic descriptions of goals, or actual plans describing cognitive or computational methods to achieve those goals).

Knowledge Resource In C-ODO a *knowledge resource* is a role (defined in an *ontology-lifecycle description*) that classifies *information objects*. *Knowledge resources* include any piece of information that is used for developing ontologies. In particular, however, they include *formal expressions* such as architectural *design patterns*, queries, ontologies and *ontology elements* (i.e., formal expressions which are *proper-part* of an ontology, e.g. axioms, classes, individuals and relations).

Working Knowledge Item These are ontologies or *ontology elements* when seen as evolving objects on which a knowledge creator is working.

Knowledge products These are ontologies or ontology elements when seen as final results of an *ontology project*.

3.2.2 C-ODO's concepts in terms of situations

The situational counterpart of an ontology project is an *ontology-project-execution*. Executions can be more or less constrained according to the constraints, preferences, and resources declared in the project. An *ontology-project-execution*, which in C-ODO is classified as a *situation* (a concept taken from the Ontology of Descriptions and Situations introduced in section 3.3.1) is based on the following five elements:

1. At least one *epistemic workflow enactment*,
2. At least one *design operation*,
3. At least one *knowledge collective,*
4. At least one *rational agent,*
5. Some *information objects*.

Epistemic Workflow Enactment This trivially is the enactment of an *epistemic workflow*, as introduced in 3.2.1. Important elements of such situation are the *argumentation situations* that come with it and that are situational counterpart of the *argumentation structure*.

Design Operation In C-ODO a *design operation* is an action carried out to accomplish some *functionality*, according to the method represented by a *functionality description*. *Design operations* are the prominent *design solution*s. *Design solutions* are structural states that include only components (e.g. *ontology elements*) and their relations. In the requirement-specification-implementation cycle, ideally, each design operation should be performed, assisted, or approximated by a computational event. Additionally, design operations make *design makings* possible. These are the instantiations of a *design rationale* in the design of an ontology. In other words, a *design making* is the situation counterpart of a *design rationale*, where agents and teams make choices based on that rationale and create a new state of an ontology. A *design making* therefore is the *setting for* at least the following entities, that play roles in the rationale: a *rational agent*, one or more *information objects*, a *design operation* on

them, and a *time interval* at which the operation occurs. A *design making* satisfies both a *design rationale* (its context) and a *functionality description* (its method).

Knowledge Collective A *knowledge collective* in C-ODO is a community that is characterized by some practices of knowledge communisation. In line with the ontology of Collections and Collectives [Bot06], *knowledge collectives* are formalized as *collectives* that are *unified by* (i.e. identified by or kept together by) some *epistemic workflow*.

Rational Agent C-ODO adopts the notion of *rational agent* from DnS. An agent here is rational if able to master a *description* (e.g. understand, adopt, assemble, author it, etc.).

Information Object According to the Information Objects ontology [GBCL05], on which C-ODO is based, *information objects* are *social objects*, with the following features: they are realized by some entity; they are encoded within some system for information encoding; they are dependent on an encoding as well as on a concrete realization; they can express a description (the ontological equivalent of a meaning/conceptualization); they can be about any entity; they can be interpreted by an agent.

3.2.3 Examples of C-ODO-based modelling

In this section we show some examples of C-ODO-based modelling. We model the DILIGENT [PST04] argumentation method, the van Eemeren and Grootendorst [EG03] argumentation pattern, the IFCS taxonomy of types of interaction among software developers as defined in [PK91], and a collaborative workflow inspired by the Linda coordination approach [BCGZ01]. The models we show in this section formalize the implicit ontology of the mentioned approaches, but we do not intend to provide here a complete modelling of their algebraic features. All the example models are represented by OWL triples that instantiate classes and relations of C-ODO.

DILIGENT argumentation model The DILIGENT methodology is conceived for supporting distributed teams of domain experts that are involved in the process of creation and evolution of ontologies. DILIGENT also contains primitives for performing *argumentation sessions*. DILIGENT argumentation model has already been defined in a simple OWL ontology[4]. Here we model it in terms of C-ODO.
The DILIGENT argumentation model includes two actors, i.e. *argumentation roles* in C-ODO: 'participant' and 'moderator'. Some C-ODO rationales, called 'arguments' that can be expressed on either 'issues' or 'ideas'. Arguments can be of two types: 'justifications', which in turn can be either 'evaluations' or 'examples', and 'challenges', which in turn can be either 'alternatives' or 'counter-examples'.
Figures 3.2 and 3.3 depict the DILIGENT argumentation model in terms of C-ODO. The former focuses on *argumentation tasks*, while the latter on *ontology design rationale* aspects. Both tasks and rationales are modeled as instances of C-ODO classes, because C-ODO is based on *DnS* theory (see section 1, 3.3.1), which provides a unique domain of discorse for projects, methodologies, argumentation protocols, design rationales, design patterns, functionalities.

[4] http://diligentarguont.ontoware.org/2005/10/arguonto

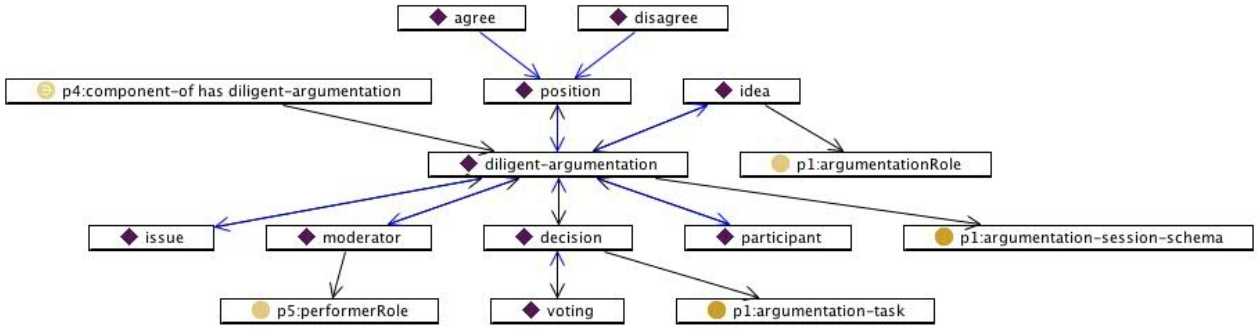

Figure 3: C-ODO-based model of DILIGENT argumentation (focus on tasks)

Fig. 3 shows the definition of *diligent-argumentation*, which is an instance of *argumentation-session-schema (i.e., a plan)*. In the context of *diligent-argumentation,* two *performerRole*s are used, they are: *participant* and *moderator*. Furthermore, two complex *argumentation-task* are defined: *decision*, which includes the *voting* task as a component, and *position,* composed of the *agree* and *disagree* tasks. *Diligent-argumentation* uses also the concepts *idea* and *issue,* which are *argumentationRole*s played by either a *design-solution* or an *information-object*. *Decision* and *position* roles are targeted to solutions or information.

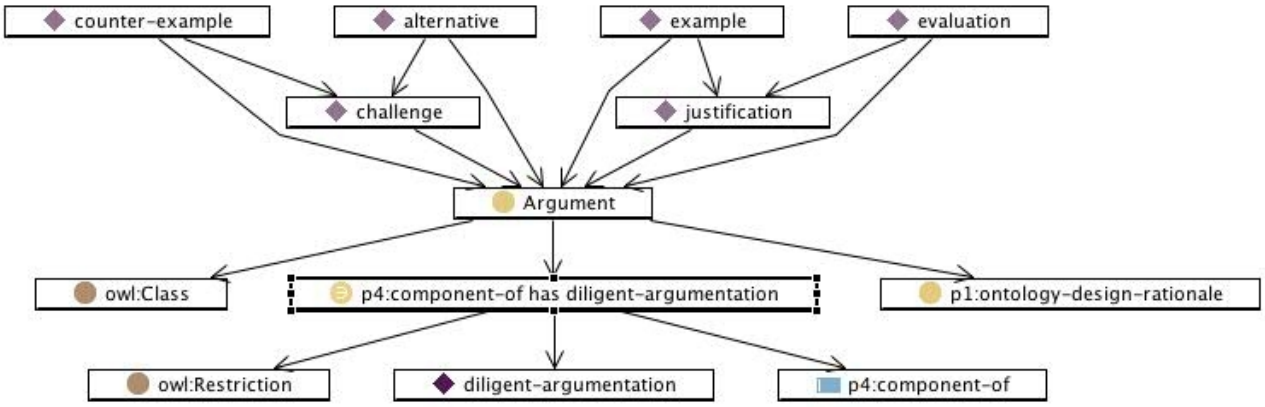

Figure 4: C-ODO-based model of DILIGENT argumentation (focus on ontology design rationale)

Fig. 4 shows another view of *diligent-argumentation* session. Here, the focus is on the design rationales that can be provided during a *diligent-argumentation* session. In DILIGENT terminology, *Argument* is synonym to the C-ODO concept of *ontology-design-rationale*, hence we have defined *Argument* as a subclass of it. *Arguments* can be of two kinds, *justification* and *challenge: example* and *evaluation* specialize (through the *specializes* relation) *justification,* while *alternative* and *counter-example* specialize *challenge.* An *ontology-design-rationale* is a component of some *argumentation-structure.* Notice that a *diligent-argumentation* can be composed only by *ontology-design-rationale*s of type *Argument*. In order to express this constraint, an OWL 'hasValue' restriction is introduced on the *component-of* relation for the class *Argument*, where the value is *diligent-argumentation.*

Pragma-dialectical argumentation framework This framework is discussed in [EG03]. The pragma-dialectic framework defines an argument session to be composed of four complex argumentation tasks: *choice confrontation*, *rationale declaration*, *dialectic rule* (application), and *argument resolution*. Fig. 5 depicts the pragma-dialectic argumentation model in terms of C-ODO

elements. Following the same approach as for DILIGENT argumentation, *pragma-dialectic-argumentation* is an instance of *argumentation-session-schema,* it uses (*uses-concept* relation) *choice-confrontation, rational-declaration, dialectic-rule,* and *argumentation-resolution*, and concepts modelled as *argumentation-task* instances, which are performed each as the *direct predecessor* to the next one.

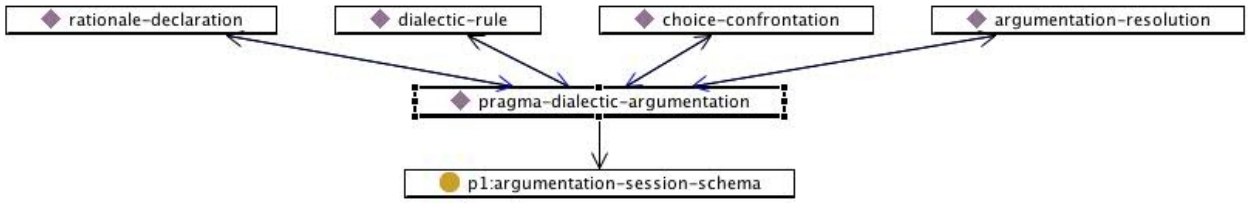

Figure 5: C-ODO-based model of pragma-dialectic argumentation framework

Collaboration patterns In [PK91] the possible models of software development environments (SDE) are discussed. SDEs are classified by means of what the authors call 'scale' dimension, i.e., the numbers of developers involved, and the size of the system being developed. In order to work out such classification, they use a sociological metaphor. The process of collaborative development of ontologies has many similarities with that of software systems. Furthermore, the models proposed in [PK91] are based on social aspects and are general enough to be suitably applied to the process of collaborative creation and evolution of any artifact. Therefore, we consider these models as examples of collaborative scenarios for the C-ODO concept of *epistemic influence*. Four kinds of configurations (organization of participants) are identified: 'individual', 'family', 'city', and 'state' (IFCS taxonomy). Furthermore, as in the social context, 'state' incorporates 'city', which in turn incorporates 'family', which in turn incorporates 'individual'. Although the authors do not describe the way e.g., workflow, agents involved in the identified scenarios should behave (they mainly focus on system requirements), for each element of the IFCS taxonomy we can associate an *epistemic workflow*.

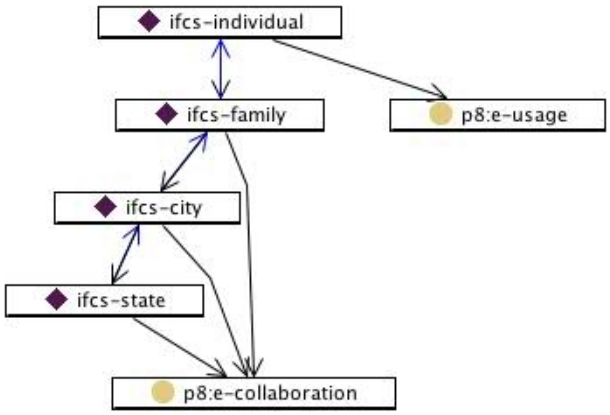

Figure 6: C-ODO-based model of ifcs taxonomy

Fig. 6 shows the IFCS taxonomy when applied to collaborative workflows, as it is modelled in terms of C-ODO. *Ifcs-individual* is a *e-usage*, which is a *proper part of ifcs-family,* of type *e-collaboration.* The other two elements are defined as instances of *e-collaboration*, and more specifically *ifcs-family* is *proper part of ifcs-city*, and *ifcs-city* is *a proper part of ifcs-state*. The

relation *propert-part-of* is a transitive object property that allows here to express the fact that a workflow is made up of sub-workflows. Such relation is defined in the Plan ontology (see section 3.3.2) together with its inverse property *proper part*.

Linda-like collaboration workflow Linda is a coordination model for agent interaction [BCGZ01]. It is based on very simple principles. A shared 'dataspace' is used as a place where messages are put on and get from. There are two roles: *sender* and *receiver*, and three possibile tasks: *out*, *in*, and *rd*. When executing the task 'out', the sender puts a message on the dataspace, and from that moment on it becomes equally accessible to all agents, while it is bound to none. The task 'in' allows a receiver to read and remove a message from the dataspace, finally the task 'rd' allows a receiver to read the message form the dataspace without removing it. This simple asynchronous policy is an example of how a collaborative work can be conducted.

In order to show an example of C-ODO modelling of collaborative workflow, consider the case when the *ifcs-family* is *specialized-by* another *e-collaboration* named *linda-diligent-familiy,* which follows a Linda-like protocol and includes the *diligent-argumentation* session schema as *argumentation-structure* (an *e-collaboration* is an *epistemic-workflow* and - as such - it has an *argumentation-structure* component).

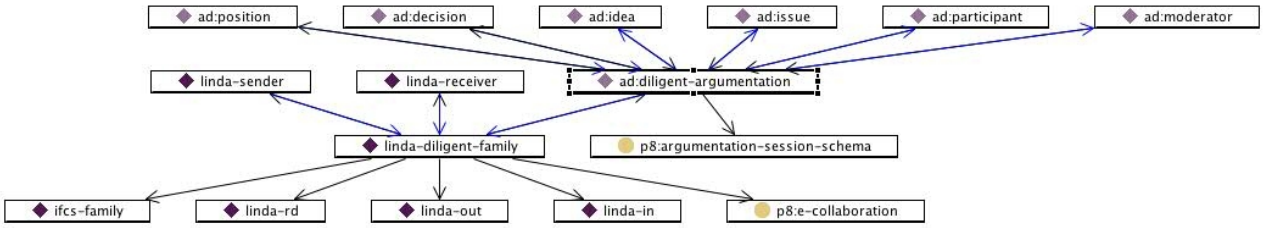

Figure 7: C-ODO-based model of linda-diligent-family example

Fig. 7 depicts the *linda-diligent-family e-collaboration*. It defines the tasks named *linda-out, linda-in,* and *linda-rd,* and uses the roles *linda-sender* and *linda-receiver.* Furthermore, *linda-diligent-familiy* has the *diligent-argumentation session schema* component.

3.3 Reused components from other ontologies

In order to identify entities and relations used in C-ODO, we reuse several existing ontological components:

- Ontology of Descriptions and Situations (DnS) [GM03],
- Plan Ontology (DDPO) [GBCL05],
- Ontology of Collections and Collectives [BCGL06],
- Information Objects ontology (DDIO) [GBCL05],
- Open Systems ontology[5],
- Knowledge Content Objects ontology (KCO) [BGMR05],
- Ontology of Ontology Evaluation, Quality, and Selection (oQual) [GCCL06a, GCCL06b],
- Ontology Definition Metamodel (ODM)[6],

[5] http://www.loa-cnr.it/ontologies/Systems.owl

- Ontology Metadata Vocabulary (OMV) [HRWB06],
- The definition of Network of Ontologies given in NeOn deliverable D1.1.1 [HRWB06].

3.3.1 Ontology of Descriptions and Situations (DnS)

The ontology of Descriptions and Situations (DnS) is a very general modelling pattern that provides a vocabulary for reification. Fig. 8 represents the most salient elements of DnS and should be read as a double-layered structure. The first layer, at the bottom of the figure, includes *description*, *concept* and *rational agent* and provides a way to model a conceptualisation. The second layer includes *situation*, which expresses the occurrences of states of affairs that comply with given *descriptions*. Extensions of DnS may be (and actually were) used to model various types of conceptualisations, such as: social and physical agents, *collectives* or communities which these agents are members of, the internal representations of descriptions by agents, the *information object* by which a *description* is expressed, the time-spans characterizing the situations.

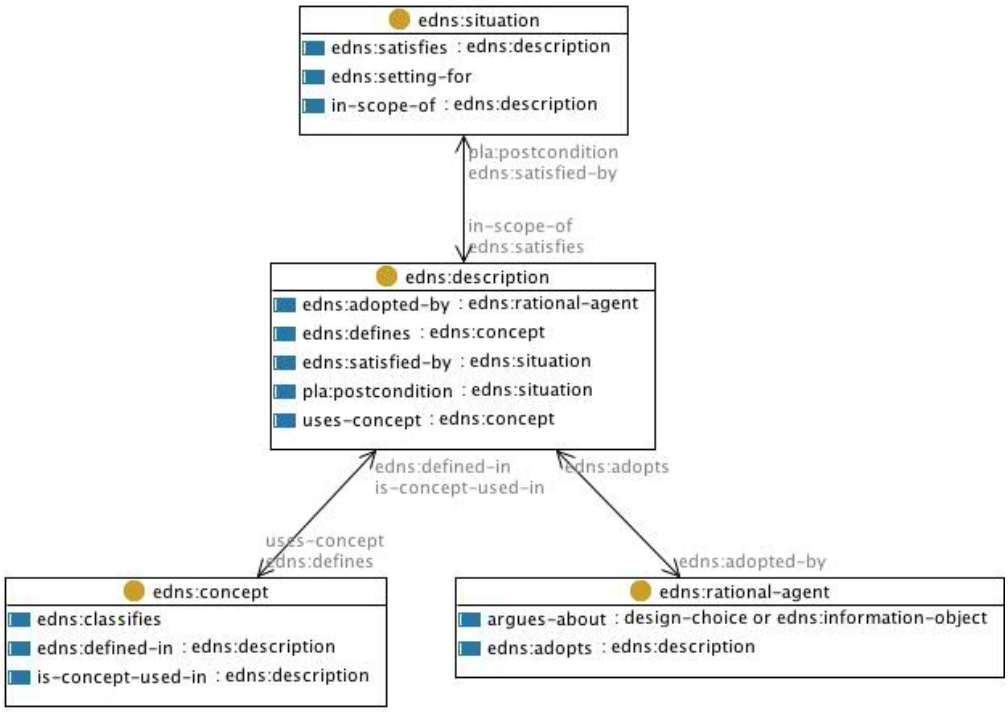

Figure 8: The Descriptions and Situations pattern as reused in C-ODO

Description This is a *social object* that represents a sharable conceptualisation, which must be communicable (i.e. expressed by means of *information objects*). *Descriptions* can be also seen as viewpoints on some state of affairs or context.

Concept This is a basic component of a description, which *defines* or *uses* it. In Fig. 8 the association *uses-concepts*/*is-concept-used-in* captures this part of the pattern by relating concepts to the description in which they are *defined* or *used*.

Rational Agent An agent here is rational if able to master a *description* (e.g. understand, adopt, assemble, author it, etc.).

Situation *Situations* express the occurrences of states of affairs that comply with a *description*. Again, *descriptions* are satisfied by *situations* or, equivalently, a specific *situation* satisfies a

[6] http://www.omg.org/ontology/

description (association '*satisfied-by*/*satisfies*'). Such satisfaction boils down to the fact that a *situation* is a setting for entities, which in one way or another comply with the *description*. Minimally, such entities are *classified* by at least one *concept* used by the *description* that the *situation* satisfies.

3.3.2 The Plan Ontology (DDPO)

Following [GBCL05], we model *plans* as *descriptions* that represent sequences of actions that lead from a given *situation* to a new one (see Fig. 9). These descriptions are abstract and independent from computational system design: they are reusable and easy-to-customize representations of the objects and activities involved in multiple action domains.

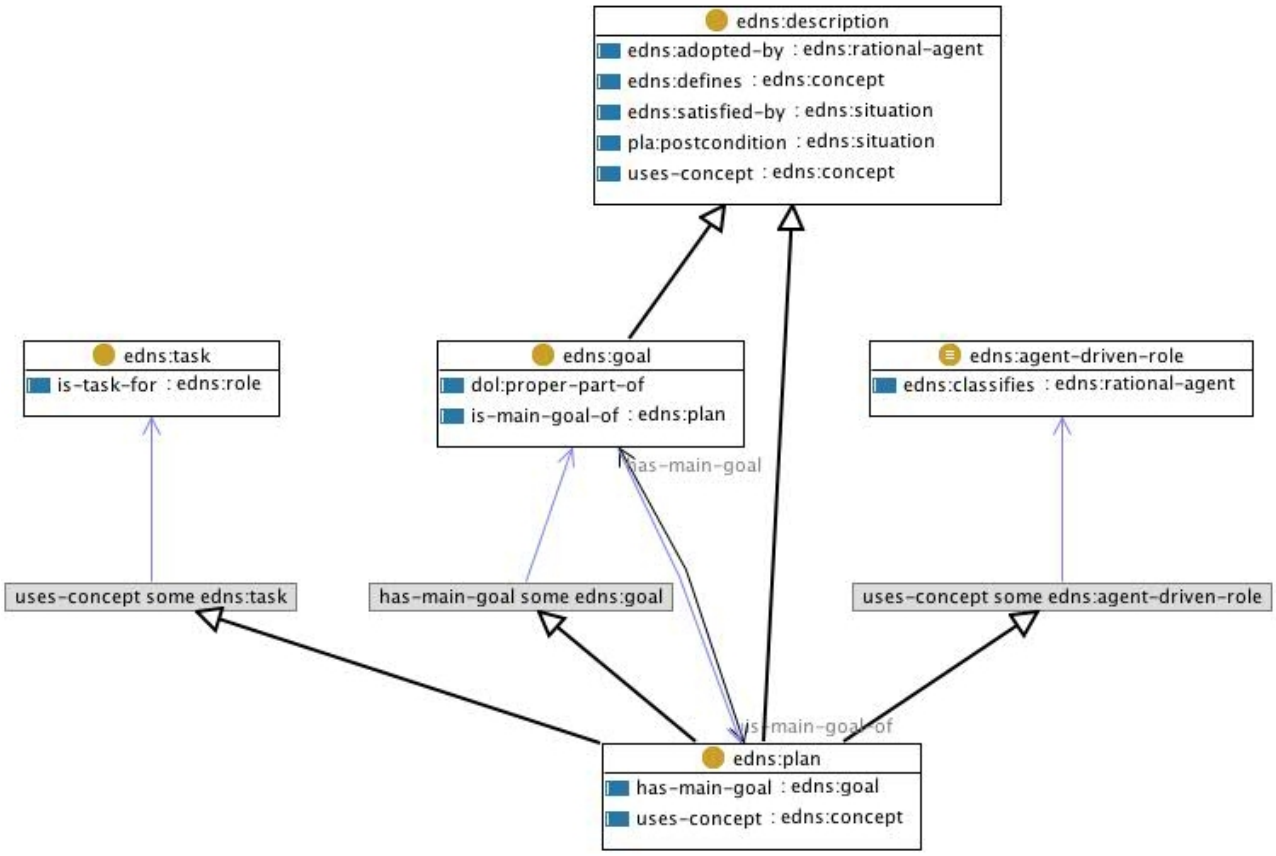

Figure 9: The structure of plan models as resued in C-ODO

Plan The main components of plans are: *task*, *goal* and *agent-driven-role*. In DnS terms, the main idea here is that a *plan defines* or *uses* at least one *task* and one *agent-driven-role* (which must *classify* an action and, respectively, an agent) towards a *goal* (which is usually *desired by* the creator or beneficiary of the plan). *Tasks* in DDPO provide instructions to execute (*classify*) *actions*. Tasks are treated as concepts defined within plans, which refer to actions (e.g. "write a deliverable"), and are organized in a subclass hierarchy. Control constructs (e.g. "choose between the following alternatives") from traditional planning and workflows are represented as *control tasks*, also defined within plans. Ordering of tasks is formalized by using *part* (mereological) relations, *control tasks* and a *successor* relation. Tasks may be connected to roles by a *target* relation, expressing the modalities (e.g. duties, obligations, or rights) that, in given plans, roles can have towards specific tasks. Moreover, a plan can have proper parts besides its goal (i.e., other descriptions). It can also include other plans, which are called *subplans*, and whose goals are called *subgoals* to distinguish them from the *main goal* of the overall *plan*. (The decision whether to consider a specific process or workflow description as an

atomic or composite plan (and its more complex tasks as subplans) is very much dependent on the needs of the formalization at hand.) Plans may also have situations as *pre-* or *post-conditions*. A situation is a pre-condition for a plan if it should preliminarily satisfy some description before executions of that plan occur. A situation is a post-condition of a plan if it should satisfy some description after plan executions of that plan occur. It often holds that the goal situation is a postcondition of plans, but this is not mandatory. Of course, every plan execution has predecessor and successor situations, but only some of them are pre- or post-conditions for the plan that the plan execution is supposed to satisfy.

Plan Execution As already seen for C-ODO, in DDPO *plans* as *descriptions* are different from *plan executions*: the latter are *situations*! Remember that a *satisfies* relation holds between situations and descriptions, implying that at least some components in a description must *classify* at least some entity in the situation setting. In the case of plans, a plan execution is a situation that (proactively) satisfies a plan description. *Goal situations* are situations that satisfy a goal.

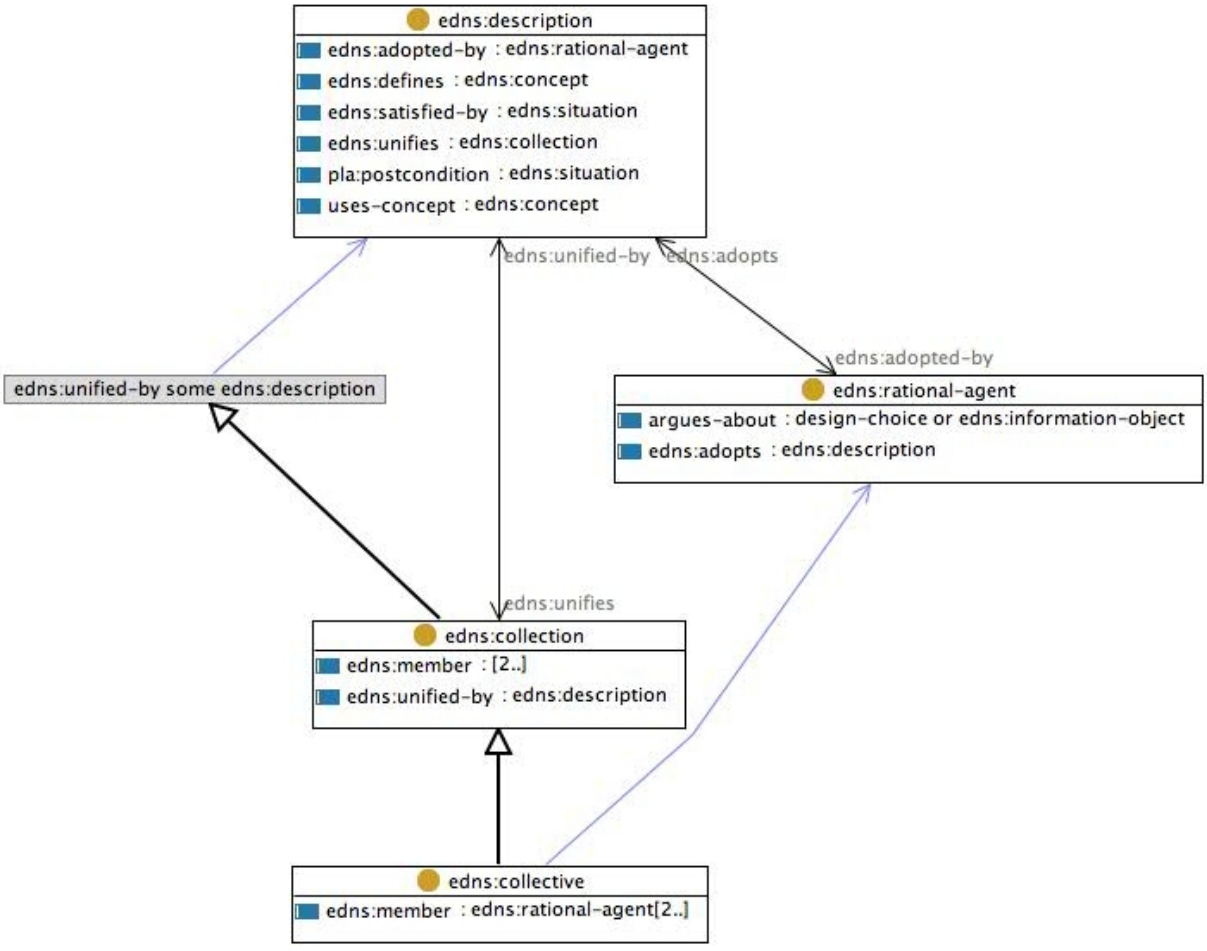

Figure 10: The Collections/Collectives pattern, as reused in C-ODO

3.3.3 Ontology of Collections and Collectives

Collections are *social objects* [BCGL06] that depend both on member entities and on some *concepts*, hence they indirectly depend on *descriptions*. We can talk of *collections* of any kind of

entities (events, objects, abstracts, etc.). As shown in Fig. 10, similarly to all *collections*, *collectives* are *covered* or characterized by roles and eventually *unified by* some *description*.

Note that, in order to model collective action (like for instance collaboration) a fine-grained set of criteria is needed to model collective-intentional aspects. To this purpose the Ontology of Collection and Collective makes use of notions taken from the Plan Ontology. In collectives, roles are played by agents. Since agents can participate in, and/or conceive, plans, roles can be assigned attitudes (participation modes) towards tasks that can sequence actions that are foreseen by the plan.

3.3.4 The Information Objects Ontology

Information Objects (hereafter, IOs) are social (hence, non-physical) products such as books, diagrams, thesauri, folksonomies or formal expressions [GBCL05]. An IO is characterized by a number of features (some of which are shown in Fig. 11):

1. It is *realized by* at least one *information realization* (i.e., a support)
2. It is *ordered by* one or more codes or languages (i.e. combinatorial structure/s or grammar/s, e.g. OWL)
3. *Expresses* one or more descriptions (i.e., conceptualisations, meanings, or viewpoints)
4. It *is about* one or more entities (i.e., the IO's reference), which are in the *settings* of one or more *situations*, which in turn *satisfy* the *description*/s *expressed by* the IO
5. It is *interpreted by* at least one rational agent.

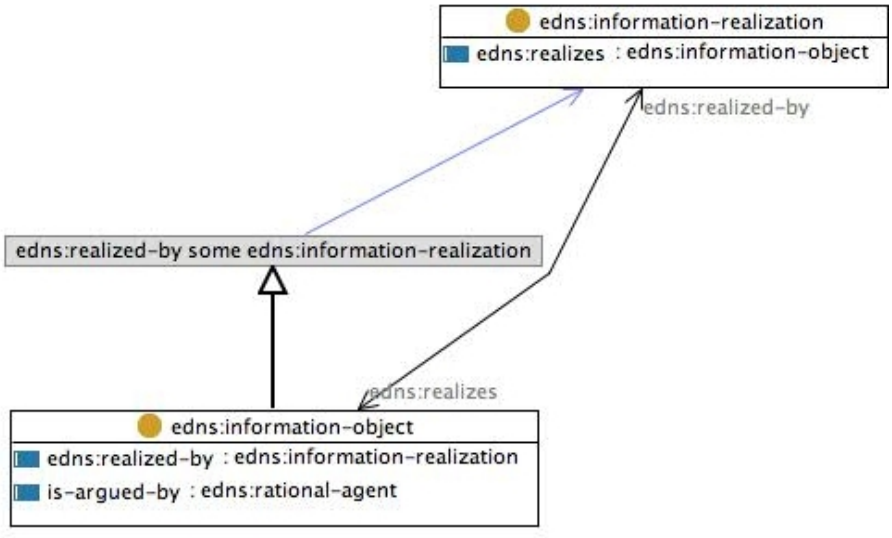

Figure 11: The Information Object and Realizations pattern, as reused in C-ODO

Hence, every IO is intrinsically provided with a viewpoint (the original *description* its author/s meant the IO to express), regardless whether this viewpoint is known to – or deemed interesting by - the agent/s that is/are currently considering the IO. The author/designer of a given IO can also be a team or organization acted by the members of a *knowledge collective*.

The relation *interprets* implies that an expressed *description* is *internally represented* by an agent; i.e., when an agent interprets an IO, it internally represents the description expressed by the IO; of course, two agents can represent different descriptions, then resulting in different interpretations for the same IO.

3.3.5 The Open Systems ontology

From the networked ontology: http://www.loa-cnr.it/ontologies/Systems.owl, inspired by von Bertalanffy's general systems theory [Ber71], we reuse here the *transformation pattern* (known in the literature as *input-throughput-output*), with the roles for *performers*, *resources*, *working items*, and *products*. In the DnS-based Open Systems ontology, *performer roles* are *superordinated to* the other roles, i.e. *resource roles*, *working item roles*, *product roles*.

3.3.6 The Knowledge Content Objects ontology

Knowledge Content Objects (KCOs, from the EU project Metokis: http://metokis.salzburgresearch.at) are knowledge containers used on a knowledge infrastructure called the Knowledge Content Carrier Architecture (KCCA) [BGMR05]. KCOs are based on the DOLCE foundational ontology and extensions to it (e.g. DnS).

KCOs have six semantic facets that help define the infrastructure implemented in the KCCA- these facets are:

1. *Content description*, the ontology through which the content can be described. This model covers all questions of the type: "What is this content about?" the ontology through which meaningful use of the content can be described. Sub-facets are user tasks, user roles, and usage history. This model describes who is intended to use this content (roles/communities) and how (tasks). It also allows knowledge content instances to be annotated by what has already happened to the content ("I have been read by all students of course 101 in computer science").
2. *Business Description*, the ontology through which contracts, pricing and negotiation about purchasing a KCO is described. This includes the ability to preview and enter into a business transaction with the owner of the KCO.
3. *Presentation Description,* the ontology through which the rendering and possible interaction with a KCO is described. For example, static web sites have very simple navigation semantics whereas semantics-based Learning Objects may have complex capabilities for context-sensitive navigation (e.g. depending on the results of self-assessment exercises).
4. *Trust and Security*, the ontology through which providers as well as consumers of KCOs can describe guarantees for the content they are selling or purchasing.
5. *Self-Description* - the ontology that describes the internal structure and semantics of KCOs.

3.3.7 The oQual ontology

oQual [GCCL06a, GCCL06b] is an ontology of ontology evaluation and validation, which models evaluation and validation as a diagnostic tasks over ontology elements, processes, and attributes. These tasks involve:

1. *Quality-Oriented Ontology Descriptions* (*qoods*), *descriptions* of an ontology, which provide the *roles* and *tasks* of the elements resp. processes from/on an ontology, and have elementary qoods (called principles) as their parts. For example, a type of qood is *retrieve*, which formalizes the requirement to be able to answer a certain competency question.
2. *Value spaces* (i.e., attributes) of *ontology elements*.
3. *Principles* for assessing the ontology fitness, which are modelled as elementary *qoods*, and are typically parts of a project-oriented *qood*.
4. *Parameters* (ranging over the attributes -value spaces- of ontologies or *ontology elements*), defined within a *principle*.

5. *Parameter dependencies* holding across *principles* because of the interdependencies between the *value spaces* of the measured ontology elements.
6. *Preferential ordering fun*ctions that compose *parameters* from different *principles*.
7. *Trade-offs*, which provide a conflict resolution *description* when combining *principles* with conflicting *parameters*.

3.3.8 C-ODO and its relation to OMV, the NeOn Networked Ontology Model and ODM

The Ontology Metadata Vocabulary (OMV) has initially been proposed in 2005 as a standard for annotating ontologies with metadata [HSHP05]. The current version OMV 2.0[7] introduced a distinction between a core part of OMV and possible extensions. The core part captures information expected to be relevant to the majority of ontology reuse settings (see **Error! Reference source not found.**). In order to allow ontology developers and users to specify task- or application-specific ontology related information, OMV extension modules can be developed. These modules are physically separated from the core scheme, but remain compatible to its element. Extensions are required to import the OMV Core ontology. While the current version of OMV does not allow referring to elements within an ontology, future work aims at allowing a more granular annotation. D1.1.1 [HRWB06] also relies on OMV as a vocabulary to describe ontology metadata in a network of ontologies.

Because OMV focuses on metadata about ontologies and not on its formal semantics or (collaborative) design, we will develop a C-ODO extension to OMV to enable these features. This way the benefits of an ontology intended to be used by applications storing ontologies (OMV) and an ontology that describes the design process and the formal semantics (C-ODO) can be combined.

In contrast to the recent Ontology Definition Metamodel (ODM) proposal by OMG's Ontology PSIG[8], the NeOn Networked Ontology Model as defined in D1.1.1 [HRWB06] does not try to cover all possible ontology languages and formalisms but is focussing on the real NeOn problems. For example, ODM does not provide a mapping or rule metamodel. It has a different scope than the NeOn Networked Ontology Model. Therefore, OMG's ODM should not be seen as a competing metamodel, but rather as a proposal with a focus not compatible with NeOn goals, although they seem to address similar problems.

C-ODO should be seen as an extension to both OMV and the NeOn Networked Ontology model. They all can coexist and benefit from one another to allow description of the whole ontology lifecycle. As a first step, we will link C-ODO to OMV (classes: ontology, author, task) and the NeOn Networked Ontology Model (classes: networked ontology, network of ontologies, contextual relations) in the next version of C-ODO. After that a proper extension to OMV will be developed.

[7] Ontology and further information available at http://omv.ontoware.org/

[8] http://www.omg.org/ontology/

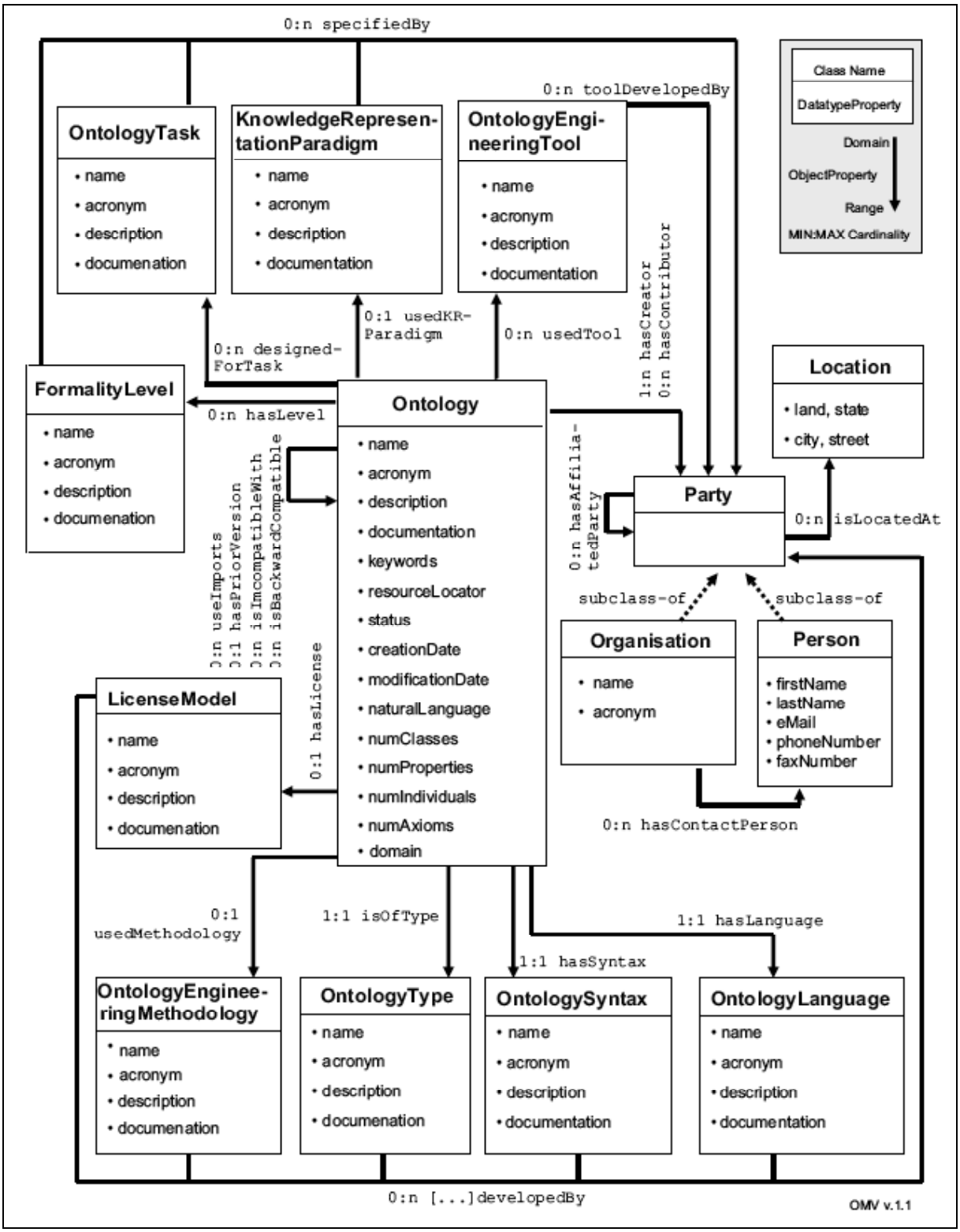

Figure 12: OMV Overview ([HRWB06], included here for convenience)

3.3.9 Network of Ontologies

We adopt here the Definition 1 of deliverable D1.1.1 [HRWB06] for Network of Ontologies and Networked Ontology:

"A *Network of Ontologies* is a collection of ontologies related together via a variety of different relationships such as mapping, modularization, version, and dependency relationships. We call the elements of this collection *Networked Ont*ologies."

Fig. 13 and Fig. 14 below show how the notions of Network of Ontologies and Networked Ontology have been included in C-ODO in terms of the concept of Collection and, respectively, Information Object. The relationships that can hold between networked ontologies are represented as `owl:subPropertyOf eConnectedTo`, whose name reminds of the proposed semantics for networked ontologies in work packages 1 and 3.

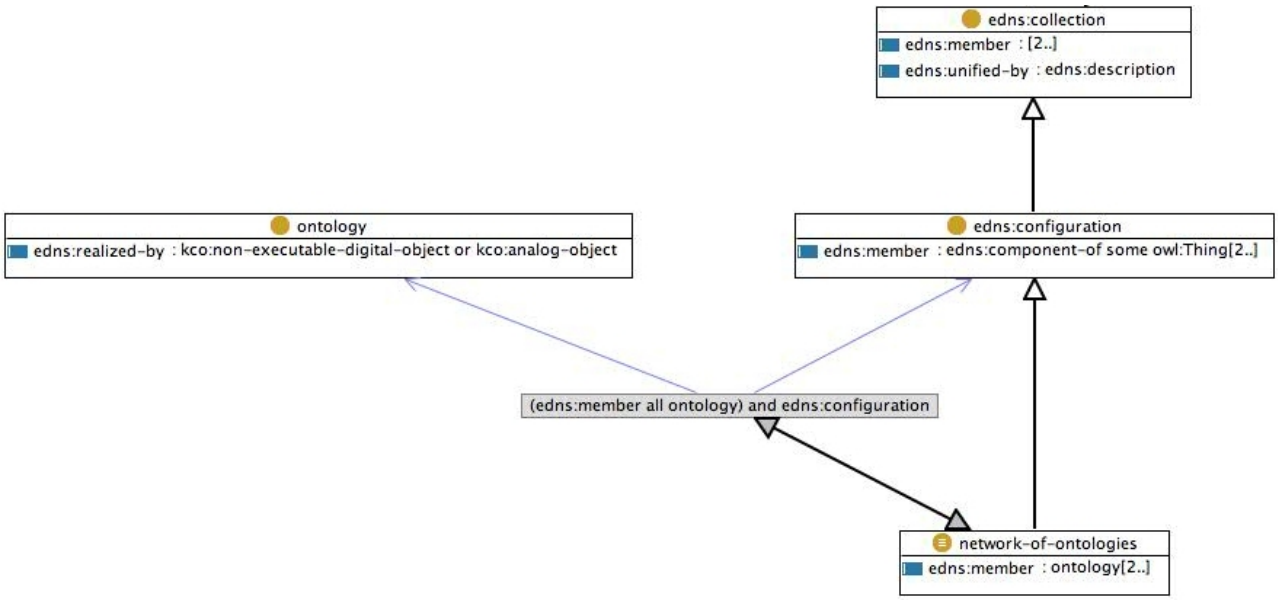

Figure 13: Network of Ontologies

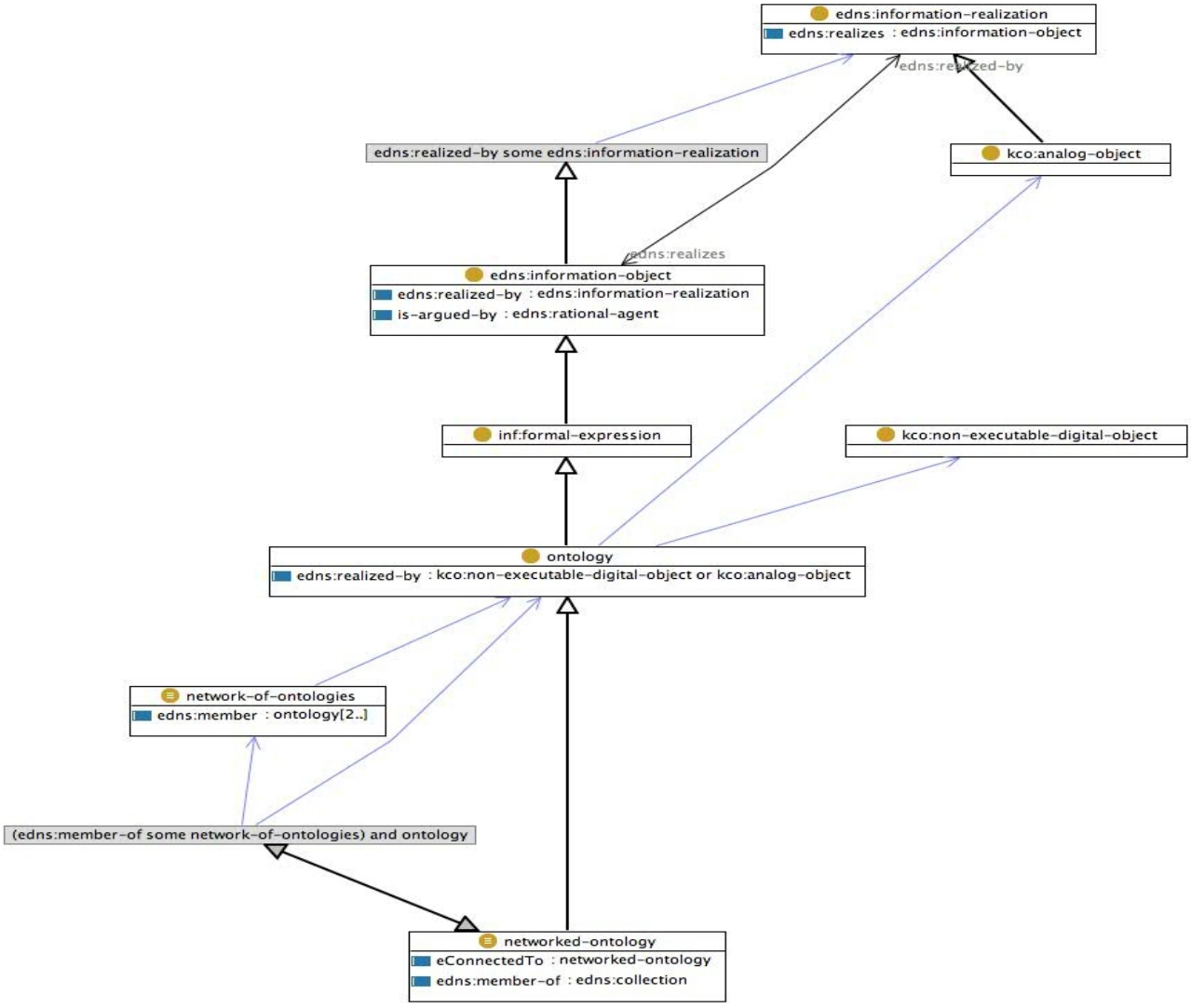

Figure 14: Networked Ontology

3.3.10 The modelling stack of C-ODO

C-ODO *traverses* the ontologies that have been described, which can be considered within a stack of languages for describing the domain of collaborative ontology design. That domain comprises social, formal, and computational entities: persons, roles, ontologies, tools, functionalities, goals, design patterns, workflows, etc. The extreme ontological heterogeneity of the domain has required a carefully designed reification vocabulary, which spans from concrete objects to abstract ideas without mixing up the entities described.

Fig. 15 shows the C-ODO stack. Ontology design domain is assumed to span across the following layers:

- the knowledge base layer (individuals and facts about a domain of interest), for example a knowledge base that contains relevant fishery objects, annotated or modelled by means of a fishery ontology;
- the ontology layer (ontologies and their elements), for example a fishery ontology;

- the ontology metadata layer (ontologies, individuals and facts about ontologies and their elements), for example the information about the authors, dates, subjects, and versioning data about the fishery ontology, expressed by means of OMV [HSHP05];
- the ontology metamodel layer (ontologies, individuals and facts about ontology formal languages), for example the formal description of representation primitives used by a formal language for ontologies such as OWL, expressed by means of ODM or the NeOn Netwoked Ontology Model (NOM) [HRWB06].

Ontology design is depicted as a vertical layer that traverses the four horizontal layers, because ontology design domain needs to talk about the relations between individuals from all those layers, e.g. between a designer, and its decision to use a certain OWL construct, or between an informal theory presented in a paper, and a design pattern encoded in UML.

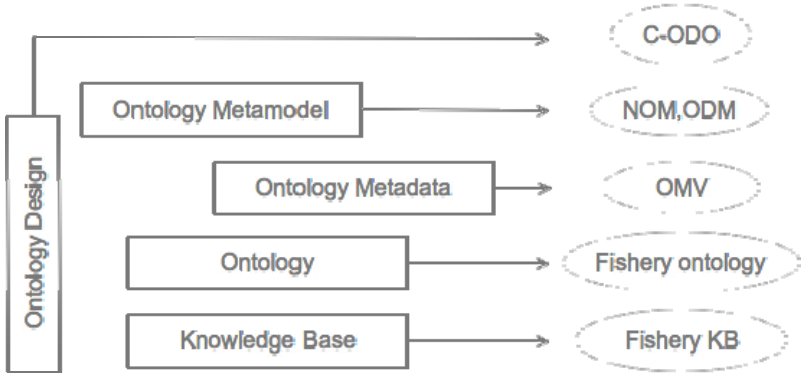

Figure 15: The C-ODO stack

Disclaimer. C-ODO is not intended as a complete language to be used by any NeOn user, for all aspects of ontology design, and in place of existing tools. On the contrary, it is an attempt to clarify issues related to different decision making procedures in collaborative ontology design: what language? What entities? What users? What functionality? What tool? are different issues that deserve a proper decision making. C-ODO helps to formalise what users, functionalities, and tools are needed in a particular context.

For example, a formal model of an argumentation method helps distinguishing it from other methods, and to express which functionalities are needed for which users, and which tools implement them. This is relevant for semantic interoperability: user requirements can be confronted to functionality specifications and to tool or service profiles. Confrontation and interoperability do not require a user to be aware of C-ODO modelling, since user aspects should be covered by appropriate user-ontology and user-tool interactions (cf. NeOn WP4).

As another source of evidence for C-ODO usability, its high expressivity in terms of the logic used (OWL-DL) and the amount of primitives employed (127 classes, currently) can be easily hidden from the final user, while exploiting its expressive power to generate very friendly collaborative ontology design environments. An experimental environment collaborative ontology design is being created by means of a tool, called SemanticFactory, which will be presented in the deliverable D2.3.1. The environment will create and maintain automatically a wiki that keeps updated an ontology and the design discourse around it. That environment will keep C-ODO hidden hidden from the user, while allowing all its semantic expressivity in an operational way.

4. C-ODO as a language for collaborative ontology design

In this section we introduce the formal definition of C-ODO. A complete presentation of the current state of C-ODO in OWL(DL) is available online in an OWLDoc-like format from: http://www.loa-cnr.it/ontologies/HTML/CODO/index.html.

The presentation follows here the six modules composing the ontology as they have been presented in section 3, i.e. ontology project, collaborative design workflow, argumentation, design rationale, design pattern, and functionality.

Each module has a dedicated section including: a description of background notions and rationales that are behind its ontological definition; a UML diagram (the used notation assumes a UML OWL profile), reflecting the module main concepts as they are defined and related in C-ODO; formal definitions of module main concepts, in OWL abstract syntax and English prose.

For the sake of readability we use the following prefixes in place of complete URI for namespaces (definition of prefixes is given in OWL abstract syntax), where ontologies are physically retrievable:

```
Namespace(rdf= <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Namespace(xsd= <http://www.w3.org/2001/XMLSchema#>)
Namespace(rdfs= <http://www.w3.org/2000/01/rdf-schema#>)
Namespace(owl= <http://www.w3.org/2002/07/owl#>)
Namespace(kco= <http://www.loa-cnr.it/ontologies/KCO/KCO.owl#>)
Namespace(inf = <http://www.loa-cnr.it/ontologies/InformationObjects.owl#>)
Namespace(edns = <http://www.loa-cnr.it/ontologies/ExtendedDnS.owl#>)
Namespace(sys= <http://www.loa-cnr.it/ontologies/Systems.owl#>)
Namespace(oqual = <http://www.loa-cnr.it/ontologies/EVAL/oQual.owl#>)
Namespace(pla= <http://www.loa-cnr.it/ontologies/Plans.owl#>)
Namespace(dol= <http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl#>)
```

The base namespace (corresponding to the empty prefix) is:

```
http://www.loa-cnr.it/ontologies/OD/OntologyDesign.owl#
```

4.1 Ontology projects

The generic notion of “project” is widely used in Software Engineering as a basis for Project Management Software (PMS). Even a trivial search on the Web[9] for reviews of this type of software makes clear what functionalities are usually provided by it, as well as the conceptualisations on which such software is based. Consider for instance the following list of features used for comparing to each other commercial pieces of PMS:

- Project Management, which may comprise: Task Management (e.g. Assign/Reassign Tasks, Task Progress Tracking, Task Feedback, Task Goals/Deadlines, Task Dependencies, Recurring Tasks), Scheduling (e.g. Calendars, Time Lines, Events), Gantt Charts, Reporting (e.g. Statistics, Work Load, Financials, Custom), Document Management, Budgeting, Critical Path Method, Project Templates, Scope, Milestones, Baseline.

[9] See e.g. http://project-management-software-review.toptenreviews.com

[9] This is partly taken from http://en.wikipedia.org/wiki/Project_management and http://en.wikipedia.org/wiki/Project_management_software

- Resource Management, which may comprise: Resource Details, Skill Sets, Timesheets, Materials/Supplies, Check In/Check Out, Import Resources, eMail Addresses, Costs, Resource Notes, Groups.
- Collaboration, which may comprise: Dashboard, Centralized Collaboration Centre, Team Calendars/Timelines, Issue Tracking, Print Reports/Documents, Forums, eMail Integration.
- Help/Support, which may comprise: Phone Support, Manual, FAQ, Knowledge Base.

The accent in the list above is operational: most notions are related to time and tasks, key elements of planning. Yet, the variables involved in a piece of PMS have a wider range than temporal variables only. A project does not simply have to do with scheduling activities, but also with managing resources, which are the structural elements of a project.

In more analytic terms the notion of "project", as studied for instance in fields like Project Management[10], tend to reduce the variables for describing a project to three main types:

- Time (the sum of the time required to complete each task);
- Cost (the sum of costs for labour, material, risk, plants, equipment, and profit);
- Scope (the end result).

Interestingly, the most complex of these three dimensions is Cost. This hints once more at the heterogeneity of resources, which are hardly modelled by a single variable, but only measured in terms of their financial cost.

Within these three dimensions, the constraints are set for the Project Management Activities (e.g. planning work or objectives, analysing and designing objectives, assessing and mitigating risk, estimating resources, allocating resources, quality management, issues management etc.) and for the Project Management Artefacts that support such activities (e.g., project charter, business case or feasibility study, governance model, risk register etc.).

Despite many similarities between the general notion of project presented above and the notion of Ontology Project modelled in this deliverable, our definition mainly concentrates on the aspects of a project related to its scope or end result: the ontology delivered by a project.

We do not consider issues of scheduling or cost-assessment, even though these aspects may easily be incorporated in our model. This follows from the fact that in our ontology - as shown in Fig. 16 and in the (OWL abstract syntax) definition reported in next section - Ontology Project is a plan.

The disengagement from temporal and material aspects illustrated above allows us to base our definition of Ontology Project only on the following five elements: Epistemic Workflow, Working Knowledge Item, Knowledge Resource, Knowledge Product, Functionality. These concepts are meant to convey the *epistemic nature* of an Ontology Project, i.e. the idea that an Ontology Project enables the *production of knowledge from knowledge*. In more detail:

- Epistemic Workflow: a relationship between rational agents that influences the knowledge of one or more agents in the relationship. Such influence takes place in a workflow involving some or all the agents. An epistemic workflow therefore is both an epistemic influence and a plan, having as its main goal the production of knowledge.
- Working Knowledge Item: a working item role played only by information objects.
- Knowledge Resource: a resource role played only by information objects.
- Knowledge Product: a product role played only by information objects

- Functionality: a task to be performed within an ontology project, e.g. 'evaluation'. As any other task, functionalities are 'defined-in' some description/schema, in this case in some 'functionality description'. Once defined, functionalities become available to be 'd-used-in' a certain ontology project (schema).

4.1.1 Ontology project

```
Class(ontology-project partial
  edns:plan
  restriction(edns:component someValuesFrom(epistemic-workflow))
  restriction(uses-concept someValuesFrom(workingKnowledgeItemRole))
  restriction(uses-concept someValuesFrom(functionality))
  restriction(uses-concept someValuesFrom(sys:performerRole))
  restriction(uses-concept someValuesFrom(knowledgeResourceRole))
  restriction(uses-concept someValuesFrom(knowledgeProductRole)))
```

Definition *An ontology project is a plan that is composed of some epistemic workflow (see section 4.2.1) and uses some concepts from the following classes: working knowledge object (section 4.1.2), functionality (section 4.1.3), performer role, knowledge resource role (section 4.1.5), and knowledge product role (4.1.6).*

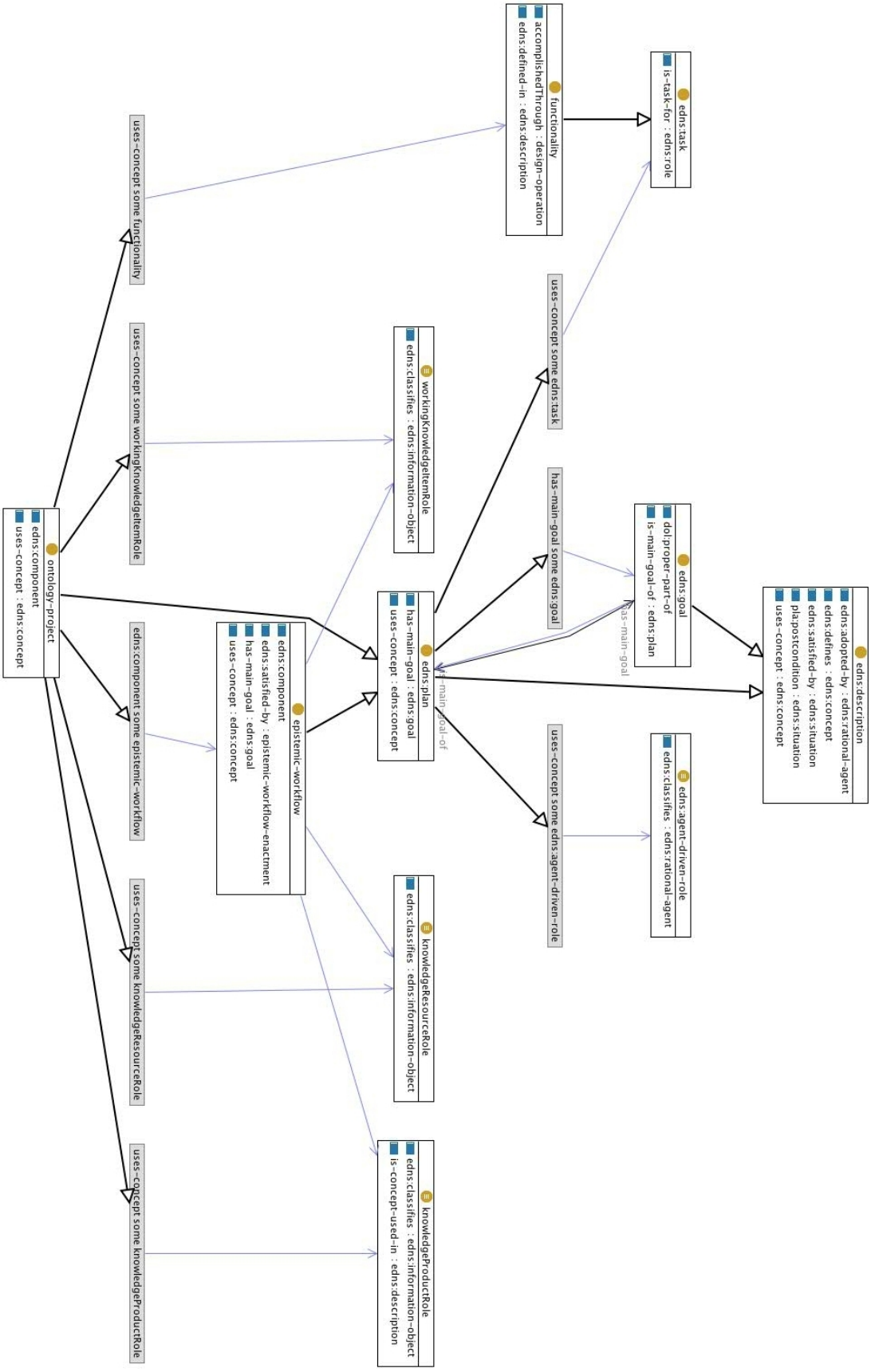

Figure 16: Ontology Project and its proximal classes

4.1.2 Knowledge role

```
Class(knowledgeRole complete
  intersectionOf(
       edns:role
       restriction(edns:classifies allValuesFrom(edns:information-object))
       restriction(is-concept-used-in someValuesForm(epistemic-influence))))
```

Definition *A knowledge role is equivalent to a role that both is classified by an information object (section 4.1.10) and is concept used in some epistemic influence.*

4.1.3 Knowledge resource role

```
Class(knowledgeResourceRole complete
  intersectionOf(knowledgeRole sys:resourceRole))
```

Definition *A knowledge resource role is equivalent to a role that is both a knowledge role (section 4.1.2) and a resource role.*

4.1.4 Knowledge product role

```
Class(knowledgeProductRole complete
  intersectionOf(knowledgeRole sys:productRole))
```

```
Class(knowledgeProductRole partial
  restriction(is-concept-used-in someValuesFrom(knowledgeProductionGoal)))
```

Definition *A knowledge product role is equivalent to a role that is both a knowledge role (section 4.1.2) and a product role. Furthermore, a knowledge product role is a concept used in some knowledge production goal.*

4.1.5 Working knowledge item role

```
Class(workingKnowledgeItemRole complete
  intersectionOf(knowledgeRole sys:workingItemRole))
```

Definition *A working knowledge item role is equivalent to a role that is both a knowledge role (section 4.1.2) and a working item role.*

4.1.6 Functionality

```
Class(functionality partial
  restriction(accomplishedThrough allValuesFrom(design-operation))
  edns:task
  restriction(edns:defined-in someValuesFrom(functionality-description)))
```

Definition *A functionality is a task which is accomplished through only design operations (section 4.1.11) and defined in some functionality description (section 4.1.7).*

4.1.7 Functionality description

```
Class(functionality-description complete
  intersectionOf(
       edns:description
       restriction(edns:defines someValuesFrom(functionality))))
```

Definition *A functionality description is equivalent to a description that defines some functionality (section 4.1.6).*

4.1.8 Ontology project execution

```
 Class(ontology-project-execution partial
  edns:situation
  restriction(edns:setting-for someValuesFrom(knowledge-collective))
  restriction(edns:setting-for someValuesFrom(edns:information-object))
  restriction(edns:setting-for someValuesFrom(edns:rational-agent))
  restriction(edns:setting-for someValuesFrom(design-operation))
  restriction(edns:component someValuesFrom(epistemic-workflow-enactment))
  restriction(edns:satisfies someValuesFrom(ontology-project)))
```

Definition *An ontology project execution is a situation that unfolds through the enactment of some epistemic workflow (section 4.2.4) and satisfies some ontology project (section 4.1.1). Furthermore an ontology project execution is a setting for some entity from classes: knowledge collective (section 4.1.9), information object (section 4.1.10), rational agent, and design operation (section 4.1.10).*

4.1.9 Knowledge collective

```
Class(knowledge-collective partial
  restriction(edns:unified-by someValuesFrom(epistemic-workflow))
  edns:collective)
```

Definition *A knowledge collective is a collective that is unified by some epistemic workflow (section 4.2.1).*

4.1.10 Design operation

Design operations are the basic actions actually executed on ontology elements, and lead to design solutions (see 4.5). In C-ODO, design operations are accomplishments of some `functionality`. A functionality is then a generic `edns:task`, defined within a `functionality-description`. Functionality descriptions can be very generic (as from the class: `generic-functionality-description`), as e.g. in UML “use case diagrams”, which only specify their `edns:goal`, or they can include a method for the achievement of that goal (as from the class: `functionality-method`). Functionality methods can finally be at the social level (`social-method-specification`) or at the computational level (`software-method-specification`).

```
Class(design-operation complete
  intersectionOf(
       restriction(edns:setting someValuesFrom(design-making))
       restriction(isTheAccomplishmentOf someValuesFrom(functionality))
       edns:action))
```

Definition *A design operation is equivalent to an action that accomplishes some functionality (section 4.1.6), within some design making situation (section 4.4.3)*

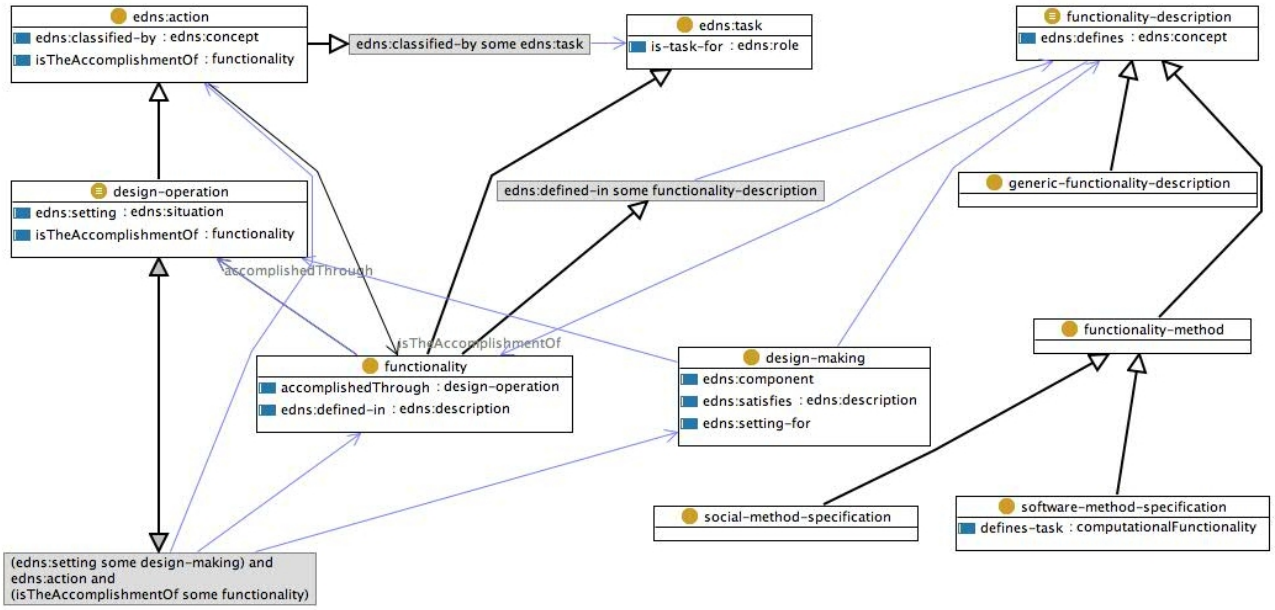

Figure 17: Design operations, functionalities, and design making situations

4.2 Collaborative design workflows

Analyses of collaboration found in the literature are rarely focused on ontologies (with the notable exception of [LU94] and [FAR96]).), and a general theory of collaboration is currently missing. For the present purposes, we will mainly focus on what directly relates to ontology design, or can be easily adapted to it.

Our notion/definition of collaboration within NeOn should:

- Cover all of the scenarios related to our use cases
- Be formalizable
- Be supportable by tools
- Be general enough to be reusable

The current subsection is therefore organised as follows. In 4.2.1 we present our definition of collaboration in the terms provided by C-ODO, hence with specific attention to collaboration towards ontology design and making. All examples will be concerned with collaborative ontology development, and wherever possible they will be taken from case studies provided by WP7 and WP8. We will then discuss typical requirements for collaborative activities (especially in online distributed environments) as found in the literature (4.2.2). Further, we will present a review of current tools that support (some of the types of) online collaboration (4.2.3), and finally discuss the so-called social-technical gap between requirements and support (4.2.4). The desired functionalities of collaboration support within NeOn (including state of the art) are then presented in Section 4.

Following intuition, a collaborative activity is concerned with the way a community carries out a complex task in a cooperative fashion.

Most approaches to collaboration identify as a necessary requirement the presence of two or more agents that work together towards a given goal. The collaboration frame in FrameNet [BFL98] is also defined around such concepts: “Partner_1 and Partner_2 or a group of Partners work together in some Undertaking"[11].

As far as knowledge-production goals are concerned, both partners are (implicitly) required to be rational agents (cf. [GAN05a]), with one of the two possibly having a more prominent role (this could be realized, e.g., as the subject of a clause). The undertaking element marks the expressions that refer to the project/plan/task on which the partners are collaborating.

When talking of knowledge communities like those targeted by ontology engineering and NeOn, we should assume, as a starting point, a very comprehensive notion of (social) knowledge relationship, including e.g. the following different situations:

1) Knowledge relationships whose participants are physically co-located, work in a same group and project and can communicate flawlessly in shared languages and with common tools. For example, two prospective NeOn users both working at UN-FAO, Rome, speaking a fluent English, both expert in their respective fields, competent on using the same communication tools, and collaborating on the design of an ontology for fishery regulations

2) Knowledge relationships whose participants are geographically distributed, work in a same community for related projects and can communicate with some fluency in a language and with some common tools. For example, two prospective NeOn users working at UN-FAO, Rome, and CABI, Oxford, speaking a decent English, both expert in their respective fields, somehow competent on using at least one communication tool, and collaborating on the design of two ontologies for fishery regulations and for fishery techniques respectively

3) Knowledge relationships, whose participants are not co-located in space, work in heterogeneous communities and cannot easily access their respective languages and communication tools, but have compatible goals. An example could comprise two users who work respectively for UN-FAO, and as an independent farmer in the Maluti Mountains, Lesotho. They have never met, have respectively inaccessible languages and, while being both experts in their respective fields, they are not necessarily competent on using the same communication tools. Moreover, they are not going to make any joint work on the design of an ontology for farming techniques; in principle, however, the knowledge owned by the independent farmer can be relevant for that task, and the UN-FAO user could involve the farmer in her project as a consultant

4) Knowledge relationships whose participants belong to heterogeneous communities, live in disjoint space-time, work on unrelated projects, and have no accessibility to their languages and communication tools. An example could comprise a UN-FAO user, and Aristotle. It is not completely absurd to think about a distinction by Aristotle to be reused by the UN-FAO user, e.g. that between matter and attribute (substance and accident in Aristotle's terms).

While the third and fourth situations cannot be considered typical collaborative scenarios, nonetheless, a form of knowledge (or ‘epistemic’) relationship can be generalized to occur even between agents living at disjoint space-time, and having apparently incompatible goals and communication means.

In order to comply with the most general and intuitive definition of collaboration, and at the same time account for all of the scenarios that might arise in the NeOn-related environments, we include the concept of `collaboration’ proper in a wider descriptive context, which we term *epistemic influence*.

[11] http://framenet.icsi.berkeley.edu/index.php?option=com_wrapper&Itemid=118&frame=Collaboration&

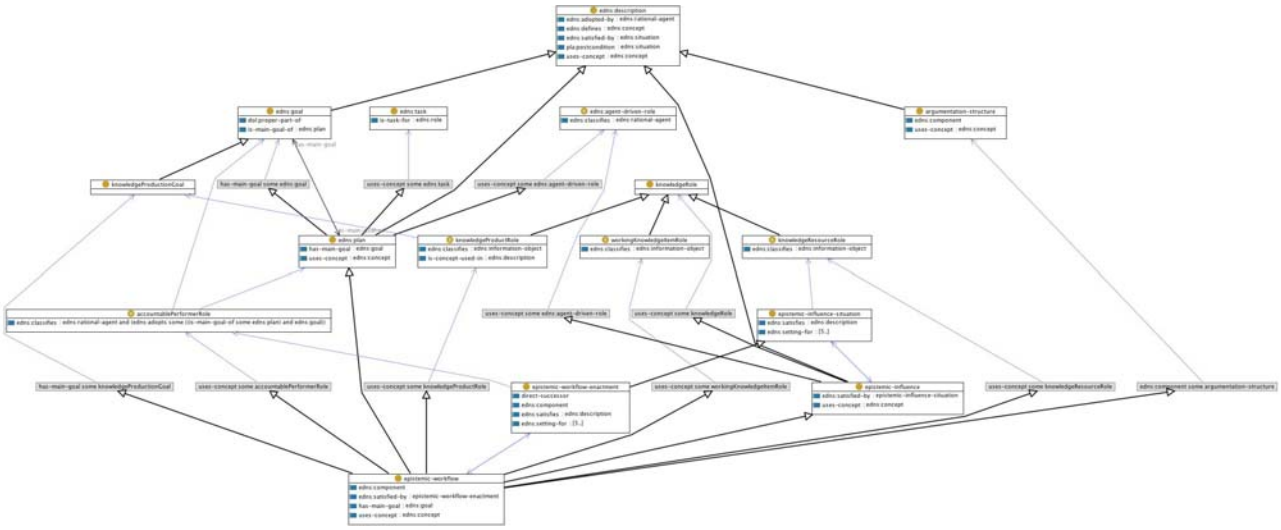

Figure 18: Epistemic workflow and its proximal classes

An epistemic influence is a relationship between rational agents that influences the knowledge of one or more agents in the relationship.

In DnS and C-ODO terms, the epistemic-influence relationship (`edns:description`) uses at least two roles:

- An agent-driven role (i.e., a role that must be played by an agent, `edns:agent-driven-role`), and
- A knowledge-resource role (`od:knowledgeRole`).

Epistemic-influence relationships are satisfied-by *epistemic-influence situations* (`edns:situation`), which are setting for at least two rational agents and one information object playing the role of knowledge-resource. They include the case in which there is only one rational agent that is 'influenced' by some information/knowledge that, in turn, has been produced at a different time by the very same agent (as when, e.g., revising one's own work).

Epistemic workflows are conceived of as a subclass of epistemic influence:

An epistemic workflow is both an epistemic influence and a plan (`edns:plan`), which:

- has a knowledge-production goal (`edns:goal`) as main goal,
- uses a working-knowledge-item role, a knowledge-resource role, a knowledge-product role (all `od:knowledgeRole`), and an accountable Performer role (`od:performerRole`),
- has an argumentation structure (`od:argumentation-structure`) as component.

Since, according to C-ODO, argumentation structures are themselves descriptions which have design rationale/s as components (see sec. 4.3), we postulate that any epistemic workflow (mereologically) includes at least one design rationale (see sec. 4.4). Rationales of the workflows themselves, on the other hand, will not be discussed in this deliverable.

Epistemic workflows, finally, are satisfied by *epistemic-workflow enactments*, which are situations that have an *agent-co-participation situation* and an *argumentation-situation* as components, a *knowledge-production-goal situation* (including in its setting the final *ontology-lifecycle product*) as direct successor, at least two rational agents/knowledge collectives, one of them necessarily playing the role of *accountable Performer*, and two information objects in their setting, playing respectively the roles of *working knowledge item*, and *ontology-lifecycle resource*.

Hence, any epistemic-workflow enactment includes the following elements:

A_1 = rational agent/knowledge collective

A_2 = rational agent/knowledge collective

K_1 = working knowledge item (expressing a *viewpoint/rationale*, e.g. a working hypothesis) of A_1

K_2 = ontology-lifecycle resource (expressing a *viewpoint/rationale*) of A_2

P = plan (e.g. ontology-design project), including at least one knowledge-production goal as *main goal* (G)

TK = at least one *task* (e.g. functionality), of which the plan consists of

R = at least one accountable performer *role*, played by one or both rational agents, that is targeted at one or more tasks

AR = an argumentation (that has a design-making component)

K_f = a final information object playing the role of ontology-lifecycle product (expressing an emerging *viewpoint/rationale*) resulting from epistemic influence, and that is in the setting of the knowledge-production-goal situation

T = at least one time interval, at which a rational agent interprets an ontology-lifecycle resource

The plan must be created or internally represented by at least one of the rational agents involved the epistemic workflow (although, as said above, in the typical collaborative case a project is 'shared' by all the involved agents; cf. [GAN05a] and [BOT06]).

Wrt the epistemic-influence relation, assigning different values of specific features applying to agents will yield different influence types, in a hierarchy ranging from the weakest influence to the highest, namely collaboration proper. More specifically, rational agents can *adopt* the plan's main goal or not, hence can be *accountable* for it or not. The second feature basically expresses whether an agent is committed (through an explicit agreement) to the plan or not.

Finally, K_f must be conceived as an element in the setting of the situation that satisfies the main goal of the plan (or, at a different level of granularity and recursively, one of its subgoal, e.g. the expected output of a task). In the case of collaborative ontology design, all the involved knowledge objects are typically *ontology object*s, i.e. bits of formal elements, annotations, or even natural language. The conceptualizations/ descriptions expressed by ontology objects can either be the result of design rationales (e.g. a certain modelling solution), or the representation of a design rationale (e.g. a comment, an argumentation tag, or an email argument).

Let us now consider some cases of epistemic workflow.

1. Let us take by way of example the following scenario (FAO use case 1.1.2 "Include a selection from an existing ontology"): an ontology expert needs to include a selection from an existing ontology into the ontology she is working with or building.

   A_1 = ontology expert, who is accountable knowledge creator (i.e., accountable performer and knowledge creator) of K_1

   A_2 = author of K_2 (could be A_1 at a previous time or any other ontology creator at any time), who, wrt P and K_f, is a non-accountable knowledge creator

   K_1 = working-ontology-lifecycle item (an information object expressing e.g. use-case requirements)

   K_2 = ontology-lifecycle resource (ontology to be selected, or take selection from)

   P = how to build ontology K_f

   G = having K_f built properly

TK = a selection from ontology K_2

K_f = ontology-lifecycle product (the final resulting ontology)

This is a type of epistemic workflow that we term *usage*. The main condition is the non-accountability of A_2. Note that this holds also in case $A_1 = A_2$, since in that case A_1 plays an accountable role when dealing with K_1, but A_2 does not, because she must have necessarily authored K_2 at a previous time, for a different task or a different execution of a same task. In other words, the identity of agents in a task of the use scenario does not require that that same agent is accountable for that task at all times of her lifecycle (this is formally represented by using admitting roles and tasks in the domain of quantification of our language).

Definition *Usage is a case of epistemic workflow that uses one additional role (with respect to epistemic workflow): non-accountable knowledge creator (for agents that have created the knowledge resource to be used in the workflow, but that are not actively involved in the current knowledge production plan).*

An usage situation is an enactment of a usage workflow. It is the setting for at least two rational agents: one accountable, the other a non-accountable knowledge creator.

2. Let us now consider another example: the same scenario as above, but A_1 asks A_2 to send her the ontology she needs in order to achieve her goal, and A_2 complies. This means that A_2 is assigned a role in a task of the project, but it does not imply that she is accountable for anything related to it. So, more specifically, elements are described as follows:

   A_1 = ontology expert, who is an accountable knowledge creator

   A_2 = author of K_2, who, wrt P and K_f, is a non-accountable knowledge creator and non-accountable performer

   K_1 = A_1's view on the ontology to be build

   K_2 = ontology to be selected (or take selection from)

   P = include a selection from ontology K_2

   K_f = final resulting ontology

This type of epistemic workflow we term *interaction*:

Definition *Interaction is a case of epistemic workflow that prescribes the proactive involvement of two rational agents. A rational agent can also be a team (a collective), provided that the members perform actions expected by a shared interaction plan. It is crucial, however, that only some of the activated rational agents adopt the goal of the interaction (some of them are non-accountable performers).*

An interaction situation is an enactment of an interaction workflow. It is the setting for at least two rational agents/teams: one an accountable performer, the other a non-accountable performer.

3. Let us now consider a third example: again, in the same scenario as above, A_1 asks A_2 to further develop her ontology in a specific direction, so that the final result will be best suited to the selection task A_1 has planned to perform in order to achieve her goal. If A_2 complies, this means that not only she is assigned a role in a task of the (new) ontology project, but also that she adopts the main goal of the project, thus becoming an accountable performer. Elements of this kind of workflow can be described as follows:

   A_1 = ontology expert, who is an accountable knowledge creator

   A_2 = author of K_2 , who, wrt P and K_f, is an accountable knowledge creator and an accountable performer

   K_1 = A_1's and A_2's shared view on the ontology to be build

K_2 = ontology to be selected (or take selection from)

P = include a selection from ontology K_2

K_f = final resulting ontology

In this case all involved agents are *accountable performers,*, and we can talk of *collaboration* proper:

Collaboration is a case of epistemic workflow that prescribes the proactive involvement of all rational agents, i.e. all the involved rational agents adopt the main goal of the collaboration plan. A rational agent can also be a team (a collective), provided that the members perform actions expected by a shared interaction plan.

A collaboration situation is an enactment of a collaboration workflow. It is a setting for at least two rational agents, both accountable.

This definition of epistemic workflow includes the 'Aristotle case' (the fourth in our initial list), which is a typical situation of 'usage'. The third of the above-listed examples, on the contrary, is a case of 'interaction', while we will talk of 'collaborative workflows' for the first two above-listed examples, which imply a sharing of both goals and tools, as well as the performance (by each of the collaborating agent) of (at least one) task defined in one and the same project. [The performed tasks should be, typically, different, but maybe it could be allowed for 'redundancy']

Based on this definition of collaboration, a network (of whatever kind) always implies a form of epistemic workflow, and vice-versa.

4.2.1 Epistemic Workflow

```
Class(epistemic-workflow partial
  edns:plan
  epistemic-influence
  restriction(edns:component someValuesFrom(argumentation-structure))
  restriction(has-main-goal someValuesFrom(knowledgeProductionGoal))
  restriction(edns:satisfied-by allValuesFrom(epistemic-workflow-enactment))
  restriction(uses-concept someValuesFrom(workingKnowledgeItemRole))
  restriction(uses-concept someValuesFrom(knowledgeResourceRole))
  restriction(uses-concept someValuesFrom(knowledgeProductRole))
  restriction(uses-concept someValuesFrom(accountablePerformerRole)))
```

Definition *An epistemic workflow is a plan the main goal of which is some knowledge production goal (section 4.2.2). Furthermore, it is an epistemic influence which is composed of an argumentation structure (section 4.3.1) and that is satisfied by some epistemic workflow enactment (section 4.2.4). An epistemic workflow uses some concept from the following classes: working knowledge item role (section 4.1.5), knowledge resource role (4.1.3), knowledge product role (4.1.4), and accountable performer role (4.2.3).*

4.2.2 Knowledge production goal

```
Class(knowledgeProductionGoal partial
  edns:goal
  uses-concept someValuesFrom(knowledgeRole))
```

Definition *A knowledge production goal is a goal whose expected situation includes new knowledge objects.*

4.2.3 Accountable performer role

```
Class(accountablePerformerRole complete
  intersectionOf(
       performerRole
       restriction(edns:classifies allValuesFrom(intersectionOf(
          restriction(edns:adopts someValuesFrom(intersectionOf(
             restriction(is-main-goal-of someValuesFrom(edns:plan))
             edns:goal)))
          edns:rational-agent)))))
```

Definition *An accountable performer role is a performer role. It is equivalent to a performer role, which classifies only those rationale agents that adopt a goal which is the main goal of some plan.*

4.2.4 Epistemic workflow enactment

```
Class(epistemic-workflow-enactment partial
  epistemic-influence-situation
  restriction(edns:component someValuesFrom(agent-co-participation-situation))
  restriction(edns:component someValuesFrom(argumentation-situation))
  restriction(direct-successor someValuesFrom(knowledge-production-goal-situation)))
  restriction(edns:satisfies someValuesFrom(epistemic-workflow))
  restriction(edns:setting-for someValuesFrom(dol:time-interval))
  restriction(edns:setting-for someValuesFrom(intersectionOf(
       edns:information-object
       restriction(edns:classified-by someValuesFrom (knowledgeProductRole))))
  restriction(edns:setting-for someValuesFrom(intersectionOf(
       edns:information-object
       restriction(edns:classified-by someValuesFrom (knowledgeResourceRole))))
  restriction(edns:setting-for someValuesFrom(intersectionOf(
       edns:rational-agent
       restriction(edns:classified-by someValuesFrom(accountablePerformerRole))))
```

Definition *An epistemic workflow enactment is an epistemic influence situation that is composed of some agent co-participation situation and some argumentation situation (section 4.3.3). It is the direct successor of some knowledge production goal situation and satisfies some epistemic workflow (section 4.2.1). Furthermore, an epistemic workflow enactment is the setting some entity from the following classes: time interval, information objects that are also classified by an ontology lifecycle product, information objects that are also classified by an ontology lifecycle resource, and rational agents which are classified by some accountable performer role.*

4.3 Argumentation

The two classical accounts on argumentation theory [Tou58, POT70] provide general schemas for representing argumentative processes. Toulmin's schema is based on four primitives: claims, datas, warrants and backings. Claims are derived from "datas" thanks to "warrants". The argumentative validity of this link depends on the acceptability degree of a "backing" supporting the "warrant". An important point is the field-relatedness of these "backings" and their "acceptability degree". They might range from jurisprudence in legal argumentation to experience results in biology. The soundness of the argumentation depends on the acceptability degree of the warrant within the competent community. In the case of ontology development, these "backings" can be seen as desired and undesired properties of the resulting ontology (consistency, coverage…). These ideas have been partly applied in [BSU+03]. However argumentation theory is lively subject

where more rigorous and formal frameworks (using informal logics and formal dialectics) are continuously developed [RN03]. The work in this direction, already recognized in the argumentation community [UBMSGL04], has to be pursued.

An important aspect development is to find the right balance between usability and formal expressiveness. A scenario using argumentation theory for ontological support might follow the characterization proposed in [EG03]. In their framework an argumentation session (a kind of collaborative workflow in C-ODO) is compound of (i) a confrontation where the problem is presented *(expression of design solution divergences)*, (ii) an opening where argumentation rules are established, including the closing conditions of the session *(the decision of acceptable ontology design rationales coming from our community best principles)*, (iii) the argumentation itself where the dialectical rules *(from a given argumentation structure)* are applied, (iv) a conclusion where the closing conditions are met.

C-ODO represents these notions by introducing the classes: argumentation structure (made up of dialectical rules: claiming, agreeing, disagreeing, refusing, etc.), argumentation situation (any situation including designers arguing on ontology design solutions), argumentation role (any role played by knowledge resources and ontology elements involved in the argumentation, e.g. preferred choice, debated choice, refused rationale, etc.), and argumentation task (any task to be accomplished within an argumentation situation). As an example, the generic [EG03] framework is represented in the class: argumentation session schema, which is a kind or collaboration workflow, and can be enacted as an argumentation session within an argumentation situation.

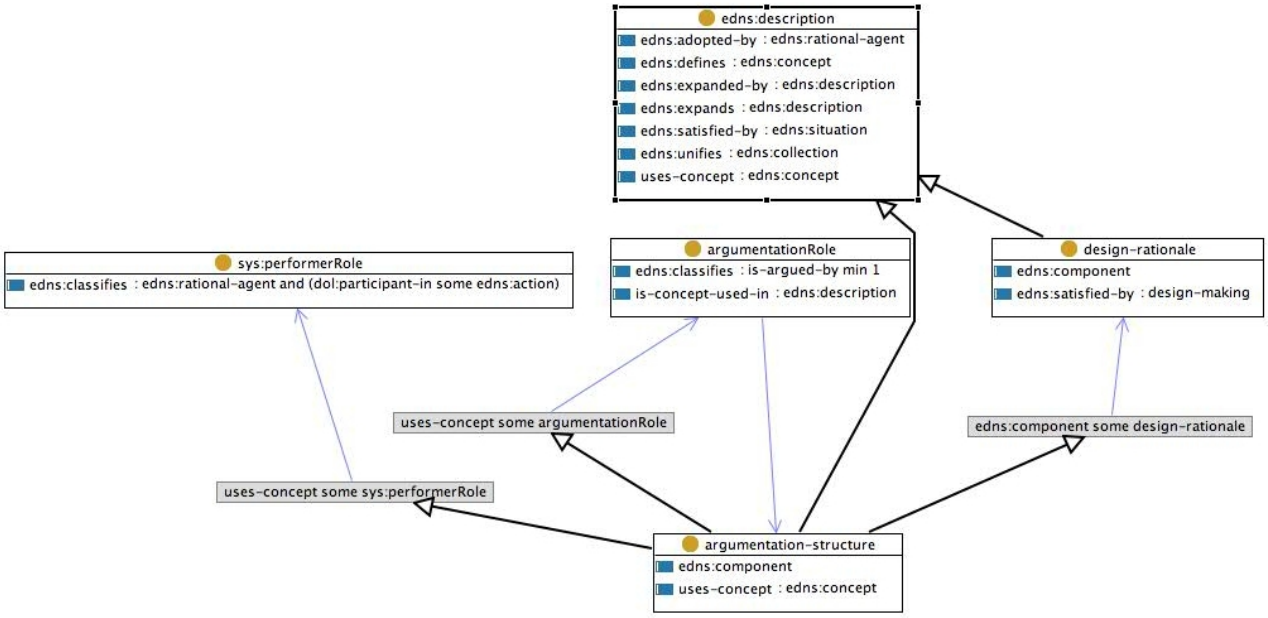

Figure 19: Argumentation structure and its proximal classes

4.3.1 Argumentation structure

```
Class(argumentation-structure partial
  edns:description
  restriction(edns:component someValuesFrom(design-rationale))
  restriction(uses-concept someValuesFrom(argumentationRole))
  restriction(uses-concept someValuesFrom(sys:performerRole))
```

Definition *An argumentation structure is a description composed of some design rationale (section 4.1.1) and uses some concept from the following classes: argumentation role (section 4.3.2), and performer role.*

4.3.2 Argumentation role

```
Class(argumentationRole partial
  edns:role
  restriction(is-concept-used-in someValuesFrom(argumentation-structure))
  restriction(edns:classifies
       allValuesFrom(restriction(is-argued-by minCardinality(1)))))
```

Definition *An argumentation role is a role that classifies only something that has been argued by a rational agent involved in an argumentation situation. Consistently with that, it is used as a concept in some argumentation structure (section 4.3.1).*

4.3.3 Argumentation situation

```
Class(argumentation-situation partial
  edns:situation
  restriction(edns:satisfies allValuesFrom(argumentation-structure))
  restriction(edns:setting-for
       someValuesFrom(restriction(is-argued-by minCardinality(1))))
  restriction(edns:component someValuesFrom(design-making)))
```

Definition *An argumentation situation is a situation that is composed of some design making, satisfies only argumentation structures and is the setting for some rational agent who has argued something.*

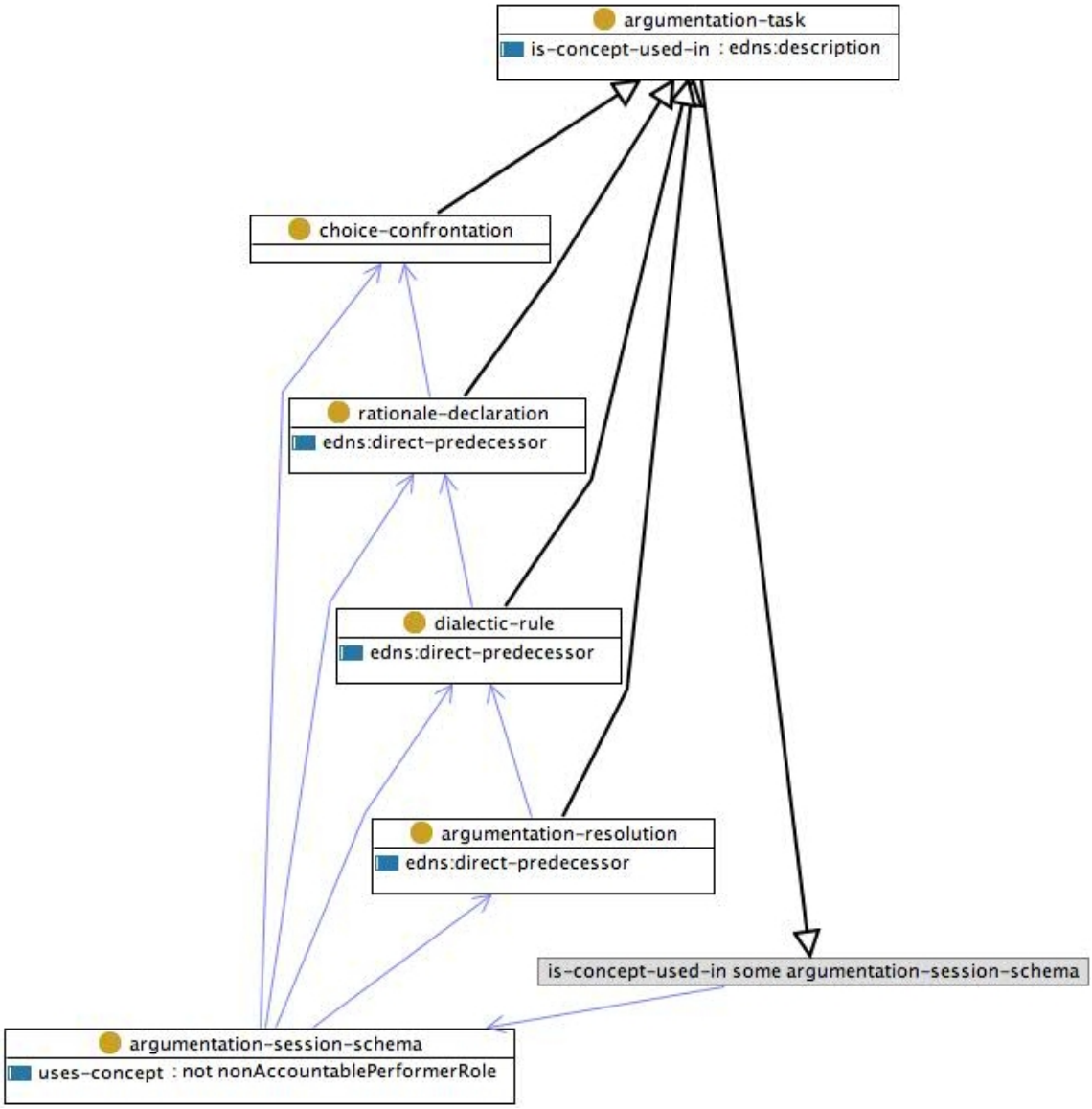

Figure 20: Argumentation session schema and its proximal classes

4.3.4 Argumentation session schema

```
Class(argumentation-session-schema partial
  edns:argumentation-structure
  e-collaboration
  restriction(uses-concept someValuesFrom(choice-confrontation))
  restriction(uses-concept someValuesFrom(rationale-declaration))
  restriction(uses-concept someValuesFrom(dialectic-rule))
  restriction(uses-concept someValuesFrom(argumentation-resolution)))
```

Definition *An argumentation session schema is an argumentation structure based on a collaboration workflow, which is composed of at least four tasks involving designers roles: a confrontation of respective choices, a declaration of the rationales assumed for those choices, an application of some dialectic rule, and a resolution of the argumentation. As a schema, it does not imply that the actual argumentation session will be carried out satisfactorily for both parties.*

4.3.5 Argumentation task

```
Class(edns:argumentation-task partial
  edns:task
  restriction(is-concept-used-in someValuesFrom(argumentation-session-schema)))
```

Definition *An argumentation task is a complex task to be achieved during the enactment of an argumentation session schema. Four typical classes of argumentation tasks have been singled out: choice-confrontation, rationale-declaration, dialectic-rule, and argument-resolution.*

4.4 Design rationales and design making

According to the historical survey provided in [REG00], design rationales (DRs) are related to the ways in which organisations capture, preserve and manage their intellectual capital. Currently, DRs constitute an intensively researched topic in a number of disciplines, spanning from mechanical design to software engineering, artificial intelligence, computer-supported cooperative work, and human-computer interaction.

In the literature, a DR has been defined as:

1. “A representation of the reasoning which has been invested in a design” ([CON91])
2. “An explanation of why an artefact, or some part of an artefact, is designed the way it is” ([LEE91])
3. "A representation for explicitly documenting the reasoning and argumentation that make sense of a specific artefact" [MYBM91]
4. “Design rationales are the solutions adopted to reach a given goal” (cf. Simon Buckingham-Shum, *Design rationale* [BSU+03])

According to [REG00], DRs should include all the background knowledge of a creation process, such as deliberating, reasoning, trade-off and decision-making in the design process of an artefact – i.e., all the information that can be crucially valuable to various people who have to deal with the artefact. In the 1980s, this information used to be expressed in terms of specifications and parameters to describe the way the artefact worked, but did not include *a description of why it had been designed the way it was*. Due to this incomplete documentation, work teams often needed (and still need) considerable amounts of communication in order to understand others’ previous work when performing maintenance and redesigning activities. Hence, research since then has mainly focused on ways of keeping track of DRs, so that they can be used both to record the history of design process (in order to modify and maintain existing designs, or to design similar artefacts) and to structure design problems (as a basis to explore new design options, and an aid to discussion and reasoning among collaborating designers). Under this respect, DRs are partially related to the 'provenance' functionality, as the latter has also the purpose of tracking the reasons why a change has occurred and to record the history of the design process.

Crucial attention must be paid to the notion of *argumentation*. According to [MYBM91], DR's analyses are not simply records of the design process, but rather they "are *co-products of design*, along with the target artefact being designed. That is to say, documented analyses are themselves artefacts - they are explicit representations that must be created and designed by designers…". A DR represents "*a structured space of design alternatives and the considerations for choosing among them* - different choices in the design space resulting in different possible artefacts". Hence, a DR "has to be constructed alongside the artefact itself… supporting both the original process of design and subsequent work on redesign and reuse… serving as a vehicle for communication".

One aim of our work is to provide people involved in the design of ontologies with a way to represent the rationales that characterizes their design decisions, and how these rationales are discussed within an argumentation session. On the basis of the definitions 1 to 4, we distinguish between solutions adopted and reasons behind such choices, and assume that design rationales represent reasons.

Definition *A design rationale (DR) is as an explicit statement (or set of statements), which represents the reasons why an object (product) has been (or is being) designed the way it is.*

At a very general level, we will then represent design activities on the basis of projects, which in turn include workflows (or coordination patterns) involving argumentation schemas that lead to explicitly formulate (by means of DRs) the motivations for the choice of a specific design-pattern schema.

DRs involve, and can be analyzed along, three dimensions:

- *Content*: deals with the objects used or created in order to design the product;
- *Task* (or *service*): is related to the requirements the product is meant to meet;
- *Sustainability*: "what use at what cost", cf. [SHU94]

When a rationale is applied to one or more ontology elements, a configuration of possible design solutions emerges. Hence, an ontology design rationale can be seen both as a description of the motivations behind specific ontology design solutions, and as the principle according to which those choices have been made available.

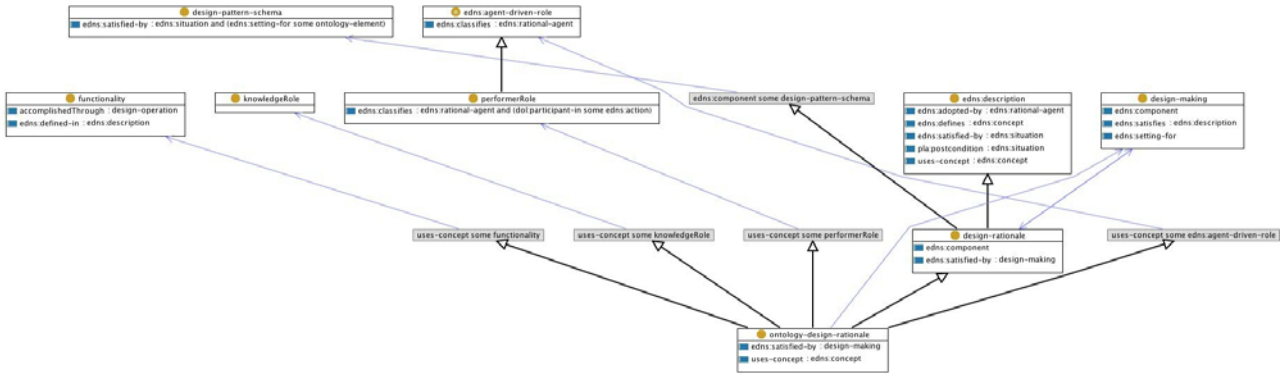

Figure 21: Ontology design rationale and its proximal classes

Although design solutions involve operations to materialize a design, we distinguish between design solutions concerning the structure of ontologies, and choices concerning their production, implementation, etc. The first set of choices is represented with reference to so-called 'ontology design patterns', while the second set is represented with reference to 'lifecycle design patterns'.

The first set is the main focus in the current version of this ontology, while the second will be possibly considered in future work. Note that this ontology currently includes the representation of production and implementation of ontologies, but not (explicitly) the representation of their design solutions (i.e. those related to 'lifecycle design patterns').

Ontology design can be motivated with reference to task or sustainability criteria (see below), but it is typically motivated with reference to structure and content, i.e. against a choice space created by the use of a pattern (either implicit or explicit) and a rationale that motivates its application.

For example, if a designer is designing an ontology from a flat list of 10 classes, including {Dog, Canine, ...}, and is willing to apply the `subClassOf` axiom pattern to Dog, the implicit choice space has a cardinality=10 (9 possible superclasses, or being a top class).

If the rationale of choosing e.g. the axiom `Dog subClassOf Canine` is that Dog's instances are all Canine's instances, the designer is applying an 'extensional semantics' rationale to the pattern employed.

In other words, the design pattern (in the example, the `subClassOf` axiom pattern) represents the quality of a certain solution (in the example, the actual axiom), whose rationale is 'extensional semantics', which leads to the design solution of the `Dog subClassOf Canine` axiom, because all Dog's instances are also Canine's instances.

As a matter of fact, the choice can be motivated by different rationales, e.g. the axiom `Dog subClassOf Canine` can be supported by a 'lexical semantics' rationale, which could take WordNet hyponymy relation as evidence.

Another rationale is the 'expertise semantics' rationale, which could use a poll system against a sample of domain experts voting on the best candidate as Dog's super-class.

Still another rationale is 'best approximation semantics' to a repository of content design patterns, so that the best candidate as a super-class may be chosen from an existing content pattern that includes Dog and Canine, with a best similarity match.

Arbitrarily complex rationales can be built to support decision on simple patterns like `subClassOf`, as well as on very complex ones, like content pattern composition.

As another, more complex example, a choice can involve two different patterns. E.g. a designer wants to create a relation between A and B, but she is undecided if the relation is `A subClassOf B` or `A partOf B`. The two possible choices are not two axioms, but two different axiom patterns, which have their own related choice spaces. The choice space of the designer is the union of those choice spaces, while the rationale is some combination of the rationales pertaining to the different choices.

In practice, the designer can apply:

- an *extensional* rationale to `A subClassOf B`: the relative choice space has a cardinality of 4: either all A's instances are all B's instances, or not all, or none (disjointness), or unknown, and

- an *intensional* rationale to `A partOf B`: the relative choice space has a cardinality of 4: either it is conceivable in the designer's world to think of all A's instances as being part of some B's instance, or it is conceivable in the designer's world for any A's instance to be part of any B's instance, or no A's instance can ever be part of any B's instance, or it is unknown.

The combined rationale is executed against a choice space with a cardinality of 4*4=16, i.e. any combination of one choice from the first set and one choice from the second set: first set: either `A subClassOf B`, or `A overlaps B`, or `A disjoint B`, or `unknown(A subClassOf B)`, and second set: `A partOf some B`, or `A partOf only B`, or `A partOf exactly 0 B`, or `unknown(A partOf B)`.

Choice spaces are defined here as collections of choice spaces, i.e. possible situations resulting from the application of a rationale to one or more design pattern, either architectural (*untyped*, see the class: `architectural-design-pattern`), or content (*typed*, see the class: `content-design-pattern`).

Ontology design rationales can be very general, as when they work as guidelines for a class of design makings, as well as very specific, as when they work as criteria for a particular design making.

Following the three proposed dimensions of analysis, ontology DRs can be of different sorts, and a specific rationale can result as a composition of these sorts:

- A first sort, including *content* rationales (exemplified above), is based on data, such as attributes, measurements, natural language evidence, etc. Content rationales motivate a choice within a choice space available for a design operation.
- A second sort, including *task* rationales, is based on queries (also called *competency questions*) that motivate a choice within a choice space available for a design operation.

Motivating queries can be represented as use case descriptions, and can be exploited as unit tests for ontologies [VG06]. Task rationales usually address complex choices, because they are matched to clusters of design solutions that can satisfy a query. Task rationales can be used at best together with a 'best approximation semantics' rationale, i.e. together with a repository of content design patterns (the repository can be either existing, or created on the fly).

- A third sort is *sustainability* rationales, usually pragmatic principles that motivate a choice. For example, the cost of a design solution, measured in hours, money, computational complexity, etc., is a typical sustainability rationale. Another one is provenance, by which authoritativeness of a solution is advocated for reuse.

When a design rationale is satisfied, this results into a *design making*, i.e. the situation in which discussed rationales are implemented in order to obtain certain *design solutions* out of the *choice space* allowed by the rationale. Design making is unfolded by executing *design operations* that accomplish a *functionality* by following a method, either social or computational. In the last case for example, a design operation can be assisted or made by a *software system* (i.e. a “tool”).

Consequently, design makings actually satisfy both a design rationale, and a *functionality method*.

4.4.1 Design rationale

```
Class(design-rationale partial
  edns:description
  restriction(edns:component someValuesFrom(design-pattern-schema))
  restriction(edns:satisfied-by allValuesFrom(design-making)))
```

Definition *A design rationale is a description that involves some design pattern schema (section 4.5.1) and is satisfied only by a design making (4.4.3).*

4.4.2 Ontology design rationale

```
Class(ontology-design-rationale partial
  design-rationale
  restriction(uses-concept someValuesFrom(functionality))
  restriction(uses-concept someValuesFrom(edns:agent-driven-role))
  restriction(uses-concept someValuesFrom(knowledgeRole))
  restriction(uses-concept someValuesFrom(sys:performerRole))
  restriction(edns:satisfied-by allValuesFrom(design-making))
```

Definition *An ontology design rationale is a design rationale that involves some concepts from the following classes: functionality (section 4.1.6), agent driven role, knowledge role (section 4.1.2), and performer role. Furthermore, an ontology design rationale is satisfied only by a design making (4.4.3). An ontology design rationale is also a component of an argumentation structure (section 4.3.1), and addresses some design pattern schema (section 4.5.1).*

4.4.3 Design making

```
Class(design-making partial
  edns:situation
  restriction(edns:satisfies someValuesFrom(design-rationale))
  restriction(edns:satisfies someValuesFrom(functionality-description)))
  restriction(edns:component someValuesFrom(design-solution))
  restriction(edns:setting-for someValuesFrom(edns:information-object))
  restriction(edns:setting-for someValuesFrom(edns:rational-agent))
  restriction(edns:setting-for someValuesFrom(design-operation))
```

Definition *A design making is a situation that involves some design solution and satisfies both some design rationale (section 4.4.1) and some functionality description (section 4.1.7). Furthermore, a design making is the setting for some entity from the following classes: information object (section 4.1.10), rational agent, and design operation (section 4.1.10). A design making is the instantiation of a design rationale in the design of an ontology, and it is a component of an argumentation situation (section 4.3.3).*

4.5 Design patterns and choices

Reusing existing artefacts is a practice that advantages designers of any field (i.e., software, business, etc) in undertaking their tasks. The same applies to ontology designers that can reuse previously produced ontologies or parts of them. In our framework we define the main concepts and relations useful for representing such design behaviour. Informally, ontology design patterns can be of several kinds:

- Logical constructs (axiom schemas), e.g. the following OWL constructs:
  - `<owl:Class> <rdfs:subClassOf> <owl:Class>`
  - `<owl:Restriction> <owl:someValuesFrom> <owl:Class> <owl:onProperty> <owl:ObjectProperty>`
- Macros of logical languages, in the sense of [Vra05], e.g. compositions of OWL constructs that are typical, such as the conjunction of the two logical constructs given above:
  - `<owl:Class>`
    - `<rdfs:subClassOf> <owl:Class>`
    - `<rdfs:subClassOf> <owl:Restriction> <owl:someValuesFrom> <owl:Class> <owl:onProperty> <owl:ObjectProperty>`
- Architectural design patterns are formal expressions for solving structural issues. Macros are simple examples of architectural patterns, while maintaining a *binary graph* structure all over an ontology is a complex one
- Content design patterns [Gan05] are “typed” macros, since they refer to a non-logical vocabulary. They help a designer solving domain specific issues. Examples include generic patterns for describing workflows, systems, components, plans, roles, as well as specific patterns for describing recipes, invoicing, soccer events, book subjects, medical diagnoses, etc.
- Ontology anti-patterns, which identify wrong solutions recurrent in recurrent design issues. The confusion between different partial orders (e.g. sub-class-of vs. part-of) is a typical anti-pattern.

Design pattern schemas are used in order to guide designers in using ontology design patterns.

Fig. 22 shows the C-ODO definition of *design pattern schema* and *design pattern skin.*

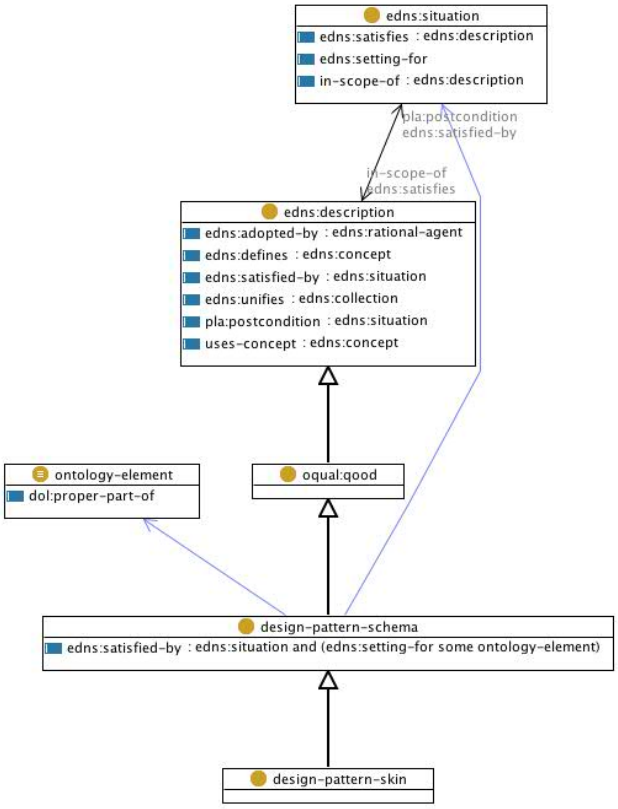

Figure 22: Design pattern schema and ist proximal classes

A *design pattern schema* (DPS) is a description that includes the roles, tasks, and parameters to implement a 'best' practice for an ontology design operation. For example, a DPS for content creation provides instructions to clone, choose, compose, or specialize an existing ontology or collection of ontology element. If the schema contains only non-procedural descriptions it is called here *design pattern skin*.

A skin is a schema for purely 'structural' design patterns, i.e. descriptions of the elements to be put into an ontology that is supposed to reuse an *architectural* or *content design pattern* (see Fig. 23 and its description below). For example, a skin for the `subClassOf` pattern will be defined with one role that classifies only a subsumption relation, and two roles for classes in that partial order. A skin for a `partOf` pattern will be defined e.g. with one role for a mereological relation, and three roles for two entities of a same category (e.g. either objects or events), and one role for a time interval.

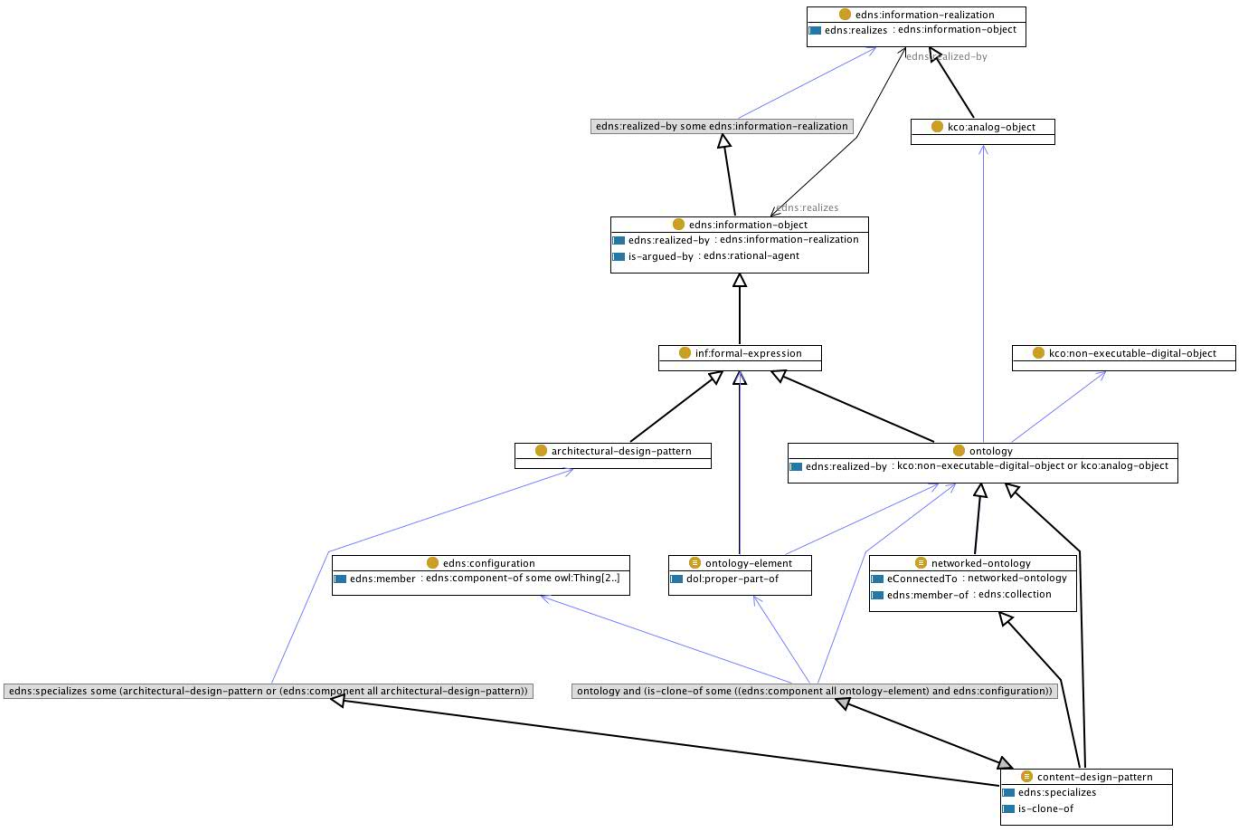

Figure 23: Content and architectural design patterns

Fig. 23 depicts how we define the concepts *architectural design pattern* and *content design pattern.*

Architectural design patterns are formal expressions, whose only parts are expressions from the logical vocabulary of a theory i.e., they are *untyped* ontology design patterns.

Architectural patterns can be either very simple, such as OWL macros [Vra05], or very complex. Among complex architectural patterns, the famous Aristotelian genus et differentia specifica for building taxonomies with explicit, incremental criteria, is an architectural pattern.

Other examples of architectural patterns are: matricial (based on specific values that distinguish the application of properties to different classes), free-style (creating axioms first, then trying to organize them into ordered sets), descriptive (creating two-layered models for making sense of reification and/or meta-modelling), split existential (splitting the complexity of reasoning with existential restrictions between two-layered models), etc.

A content design pattern (CODeP) [Gan05] is a clone of a configuration of ontology elements, which are usually from a same reference ontology. A CODeP is a *typed* design pattern, because it is a clone of a collection of ontology elements that have a 'type', i.e. a name from the non-logical vocabulary of a theory.

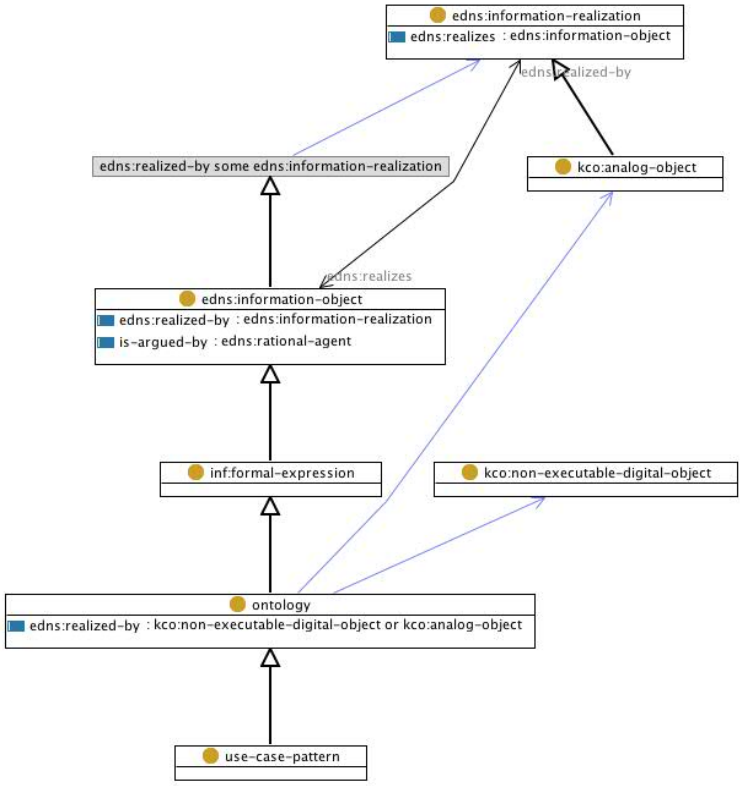

Figure 24: Use case pattern

Fig. 24 shows how the concept *use case pattern* is defined in our framework. Specifically, a *use case pattern* is a formal encoding of a use case. It can either be generic or specific, i.e. bound to a certain domain. Typical examples of use case patterns are competency questions.

The situations that satisfy a design pattern schema are called *design solutions*. A design solution is a purely structural situation, which represents a state of an ontology or a part of it, at time *t*. For example, the fact that version 36 (December 19th, 2006) of C-ODO contains the `owl:Class design-operation`, which is a `rdfs:subClassOf edns:action`, is a design solution.

A design solution is a `setting-for` some `ontology-elements`, the additional `edns:information-objects` that identify or annotate ontology elements, and the `dol:time-intervals` at which a configuration of those ontology elements holds.

Identifiers and (informal) annotations of ontology elements are not ontology elements, because they are not `inf:formal-expressions` (like ontology elements are), but they are constitutive parts of ontologies. For example, an ontology element can be `lexicalized-by` some `inf:linguistic-object` in a certain `inf:natural-language`. Multilingual lexicalization of ontology elements can be represented by using these primitives, and still preserving the identity of linguistic objects, which are not reduced to strings (although a string can still be used to add a datatype to them).

4.5.1 Design pattern schema

```
Class(design-pattern-schema partial
  oqual:qood
  restriction(edns:satisfied-by allValuesFrom(intersectionOf(
       edns:situation
       restriction(edns:setting-for someValuesFrom(ontology-element))))))
```

Definition *A design pattern schema is a qood (i.e., quality oriented ontology description) and is satisfied only by a situation that is the setting for some ontology element (section 4.5.7). Ontology design-pattern schemas are instantiated by design solutions (situations) that are the setting for ontology elements and their axioms.*

4.5.2 Design pattern skin

```
Class(design-pattern-skin partial
  design-pattern-schema
  restriction(uses-concept someValuesFrom(patternRole))
  restriction(uses-concept someValuesFrom(elementRole)))
```

Definition *A design pattern skin is a design pattern schema (section 4.5.2) that involves some concepts from the following classes: pattern role (setion 4.5.3), and element role (section 4.5.5).*

4.5.3 Pattern role

```
Class(patternRole partial
  designRole)
```

Definition *A pattern role is a design role (section 4.5.4).*

4.5.4 Design role

```
Class(designRole partial
  edns:role
  restriction(is-concept-used-in someValuesFrom(design-pattern-schema)))
```

Definition *A design role is a role and is a concept used in some design pattern schema (section 4.5.1).*

4.5.5 Element role

```
Class(elementRole partial
  designRole)
```

Definition *An element role is a design role (section 4.5.4).*

4.5.6 Design solution

```
Class(design-solution partial
  edns:situation
  restriction(edns:satisfies someValuesFrom(design-pattern-schema))
  restriction(edns:setting-for someValuesFrom(inf:formal-expression))
  restriction(edns:setting-for someValuesFrom(ontology-element)))
```

Definition *A design solution is a situation that satisfies some design pattern schema (section 4.5.1). Furthermore, it is the setting for some entity from the following classes: formal expression (section 4.5.7), and ontology element (section 4.5.8). For example, the occurrence of a subClassOf axiom (which is an ontology element) and its elements as included in a design solution complying to a subClassOf OWL macro*

4.5.7 Formal expression

```
Class(inf:formal-expression partial
  restriction(representedInLanguage someValuesFrom(inf:formal-language))
  edns:information-object)
```

Definition *A formal expression is an information object (section 4.1.10) that is represented in some formal language.*

4.5.8 Ontology Element

```
Class(ontology-element complete
  intersectionOf(
       restriction(dol:proper-part-of someValuesFrom(ontology))
       inf:formal-expression))
```

Definition *An ontology element is equivalent to a formal expression that is a proper part of some ontology, e.g. axioms, classes, individuals and relations. Furthermore, an ontology element can be lexicalized by some linguistic object in a certain language: since some types of elements do not have necessarily a lexicalization, the relation between ontology elements and linguistic objects has only a 0..* cardinality in C-ODO.*

4.5.9 OWL macro

```
Class(OWL-macro partial
  architectural-design-pattern
  restriction(dol:proper-part someValuesFrom(intersectionOf(
       ontology-element restriction(representedInLanguage allValuesFrom(OWL-species))))))
```

Definition *An OWL macro is an architectural design pattern (section 4.5.10) that has some OWL ontology element as a proper part (section 4.5.8).*

4.5.10 Architectural design pattern

```
Class(architectural-design-pattern partial
  inf:formal-expression)
```

Definition *An architectural design pattern is a formal expression (section 4.5.7).*

4.5.11 Content design pattern

```
Class(content-design-pattern partial
  networked-ontology
  ontology
  restriction(edns:specializes someValuesFrom(unionOf(
       architectural-design-pattern
       restriction(edns:component allValuesFrom(architectural-design-pattern))))))
```

Definition *A content design pattern (CODeP) is an ontology (section 4.5.14). Furthermore, it is a networked ontology (section 4.5.13) that specializes some architectural design pattern (section 4.5.10), which is composed of only architectural design patterns.*

4.5.12 Use case pattern

```
Class(use-case-pattern partial
  ontology)
```

Definition *A use case pattern is an ontology (section 4.5.14).*

4.5.13 Networked ontology

```
Class(networked-ontology complete
  intersectionOf(
       ontology
       restriction(edns:member-of someValuesFrom(network-of-ontologies))))
```

Definition *A networked ontology is an ontology (section 4.5.14). Furthermore, it is equivalent to an ontology that is a member of some network of ontologies.*

4.5.14 Ontology

```
Class(ontology partial
  inf:formal-expression
  restriction(edns:realized-by allValuesFrom(unionOf(
       kco:non-executable-digital-object
       kco:analog-object))))
```

Definition *An ontology is a formal expression (section 4.5.7) that is realized only by either analogical objects or non-executable digital objects.*

5. Tasks for the next deliverable

1. Mapping of OMV, ODM, and NeOn Networked Ontology Model to C-ODO
2. Suggesting guidelines for the formalization of the WP5 glossary of activities in C-ODO
3. Collecting C-ODO representations of requirements, functionalities and tools from the other deliverables of WP2, which will constitute the CODF (Collaborative Ontology Design Functionalities) model
4. Adding C-ODO descriptions of advanced requirements from case studies (WP7, WP8)

6. Bibliography

[AB05] H. Alani and C. Brewster. Ontology Ranking based on the Analysis of Concept Structures. InProceedings of the Third International Conference on Knowledge Capture (K-CAP 05), ACM,Banff, Canada, 2005.

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

[Ale79] C. Alexander. The Timeless way of building. Oxford University Press, New York, 1979.

[Axe84] R. Axelrod. The Evolution of Cooperation. Basic Books, New York, 1984.

[BADW04] C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data-driven Ontology Evaluation.In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC04), Lisbon, 2004.

[Bar06] J. Barrasa. Semantic upgrade and publication of legacy data. In C. Calero, F. Ruiz, and M. Piattini, editors, Ontologies for Software Engineering and Software Technology. 2006.

[BBC+ 99] N. Baker, A. Bazan, G. Chevenier, Z. Kovacs, T. Le Flour, J-M. Le Goff, R. McClatchey, and S. Murray. Design Patterns for Description-Driven Systems. In Proceedings of the 3rd Enterprise Distributed Object Computing EDOC'99 conference, Mannheim, Germany, 1999.

[BC99] M. Berland and E. Charniak. Finding parts in very large corpora. In Proceedings of ACL'99,1999.

[BCCV06] P. Bunemann, A. Chapman, J. Cheney, and S. Vansummeren. A Provenance Model for Manually Curated Data. In Proceedings of the International Provenance and Annotation Workshop,2006.

[BCGL06] E. Bottazzi, C. Catenacci, A. Gangemi, and J. Lehmann. From collective intentionality to intentional collectives: an ontological perspective. Cognitive System Research - Special Issue onCognition and Collective Intentionality, 7(2-3), 2006.

[BCGP03] J. Barrasa, O. Corcho, and A. Gomez-Perez. Fundfinder - a case study of database-to-ontology mapping. In Proc. ISWC Semantic integration workshop, Sanibel Island (FL US),2003.

[BCGP04] J. Barrasa, O. Corcho, and A. Gomez-Perez. R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In Second Workshop on Semantic Web and Databases (SWDB2004), Toronto, Canada, 2004.

[BCGP06] J. Barrasa, O. Corcho, and A. Gomez-Perez. A Semantic Portal for Fund Finding in the EU: Semantic Upgrade, Integration and Publication of heterogeneous legacy data. In Proceedings of the 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES2006), Bournemouth, United Kingdom, 2006.

[BCGZ01] Nadia Busi, Paolo Ciancarini, Roberto Gorrieri, and Gianluigi Zavattaro. Coordination Models: A Guided Tour. In Andrea Omicini, Franco Zambonelli, MAtthias Klush, and Robert Tolksdorf (eds.), Coordination of Internet Agents: Models, Technologies, and Applications, pages 6-25. Springer, 2001.

[BCM05] P. Buitelaar, P. Cimiano, and B. Magnini. Ontology Learning from Text: Methods, Applications and Evaluation. Frontiers in Artificial Intelligence and Applications. IOS Press, 2005.

[BED04] P. Buitelaar, T. Eigner, and T. Declerck. OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection. In Proceedings of the Demo Session at the International Semantic Web Conference, Hiroshima, Japan, 2004.

[Ber71] von Bertalanffy L. General System Theory. London, The Penguin Press, 1971.

[BFL98] Collin F. Baker, Charles J. Fillmore, John B. Lowe. The Berkeley FrameNet Project. Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics. Morgan Kaufmann Publishers, 1998.

[BFMP04] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta. Revising WordNet Domains Hierarchy: Semantics, Coverage, and Balancing. In Proceedings of COLING 2004 Workshop on “Multilingual Linguistic Resources", pages 101-108, Geneva, Switzerland, 2004.

[BGMR05] W. Behrendt, A. Gangemi, W. Maass, and Westenthaler R. Towards an Ontology-Based Distributed Architecture for Paid Content. In Proceedings of The Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC, volume 3532, pages 257-271, Heraklion, Crete, Greece, 2005. Springer-Verlag GmbH.

[BGP06] J. Barrasa and A. Gomez-Perez. Upgrading relational legacy data to the semantic web. In 15th international conference on World Wide Web WWW 2006, Edinburgh, UK, 2006. http://portal.acm.org/citation.cfm?id=1135777.1136019.

[BKKT06] D. Bourilkov, V. Khandelwal, A. Kulkarni, and S. Totala. Virtual Logbooks and Collaboration in Science and Software Development. In Proceedings of the International Provenance and Annotation Workshop, 2006.

[BKT00] P. Buneman, S. Khanna, and W.-C. Tan. Data Provenance: Some Basic Issues. In Proceedings of Foundations of Software Technology and Theoretical Computer Science, 2000.

[BKT01] P. Buneman, S. Khanna, and W.-C. Tan. Why and Where: A Characterisation of Data Provenance. In Proceedings of the International Conference on Database Theory, 2001.

[BOS04] P. Buitelaar, D. Olejnik, and M. Sintek. A Protege Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. In C. Bussler, J. Davies, D. Fensel, and R. Studer, editors, Proceedings of the First European Semantic Web Symposium (ESWS2004), volume 3053 of LNCS, Heraklion, Crete, 2004. Springer-Verlag.

[BSD05] N. Benn, S.B. Shum, and J. Domingue. Integrating Scholarly Argumentation, Text and Community: Towards and Ontology and Services. In Proceedings of the 5th International Workshop on Computational Models of Natural Argument (CMNA), IJCAI-05, Edinburgh, 2005. also published as Technical Report KMI-05-5, 2005.

[BSU+03] Simon Buckingham Shum, Victoria Uren, Gangmin Li, John Domingue, Enrico Motta, Visualizing Internetworked Argumentation, In: Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making. Paul A. Kirschner, Simon J. Buckingham Shum and Chad S. Carr (Eds), 2003. Springer-Verlag: London pp. 185-204

[CBY91] J. Conklin and K.C. Burgess Yakemovic. A process-oriented approach to design rationale. Human-Computer Interaction (Special Issue on Design Rationale), 6(3-4):357-391, 1991.

[CDPW02] F. Ciravegna, A. Dingli, D. Petrelli, and Y. Wilks. User-System Cooperation in Document Annotation Based on Information Extraction. In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), pages 122-137, Siguenza, Spain, 2002.

[CGR+05] M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric, and I. Rojas. Unsupervised Learning of Semantic Relations between Concepts of a Molecular Biology Ontology. In Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, UK, 2005.

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

[Cha02] S. Chakrabarti. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kaufmann Publishers, 2002.

[CHS04] P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In Proceedingsof WWW'04, 2004.

[Cir01] F. Ciravegna. Adaptive information extraction from text by rule induction and generalisation. InProceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001),2001.

[CLS05] P. Cimiano, G. Ladwig, and S. Staab. Gimme The Context: Context-driven automatic semantic annotation with C-PANKOW. In Proceedings of the 14th World Wide Web Conference, 2005.

[CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), 2002.

[CNM04] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large net-works. Phys. Rev. E, 70(066111), 2004.

[CTP00] P. Clark, J. Thompson, and B. Porter. Knowledge Patterns. In Proceedings of KR2000, 2000.

[CV05] P. Cimiano and J. Voelker. Text2Onto - A Framework for Ontology Learning and Data drivenChange Discovery. In In Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'2005), 2005.

[CW03] F. Ciravegna and Y. Wilks. Designing Adaptive Information Extraction for the Semantic Web in Amilcare. In S. Handschuh and S. Staab, editors, Annotation for the Semantic Web. IOS Press, Amsterdam, 2003.

[DDM04] J. Domingue, M. Dzbor, and E. Motta. Magpie: Supporting Browsing and Navigation on the Semantic Web. In N. Nunes and C. Rich, editors, Proceedings ACM Conference on Intelligent User Interfaces (IUI), pages 191-197, 2004.

[DEG+03] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In Proceedings of WWW'03, 2003.

[DPF+ 05] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and Ranking Knowledge on the Semantic Web. In Y. Gil, E. Motta, V.R. Benjamins, and M.A. Musen, editors, Proceedings of the 4th International Semantic Web Conference, volume 3729 of LNCS, pages 156-170. Springer-Verlag, 2005.

[DR04] W. Daelemans and M.L. Reinberger. Shallow text understanding for ontology content evalua-tion. IEEE Intelligent Systems, pages 1541-1672, 2004.

[DS06] Dobson, G., Sawyer, P., (2006) "Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web", International Seminar on "Dependable Requirements Engineering of Computerised Systems at NPPs", Institute for Energy Technology (IFE), Halden, 2006.

[DV05] T. Declerck and O. Vela. LabelTranslator: Multilingualism in Ontologies. In Proceedings of the4th Interantional Semantic Web Conference, 2005.

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

[EG03] FH van Eemeren, R Grootendorst, A Systematic Theory of Argumentation: The Pragma-Dialectical Approach, Cambridge University Press, Cambridge, 2003

[Fel98] C. Fellbaum, editor. WordNet: An Electronic Lexical Database. MIT Press, 1998.

[FFR96] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a Tool for Collaborative OntologyConstruction. In Proceedings of the 10th Knowledge Acquisition Workshop, Banff, Canada, 1996.

[Fle86] L. Fleck. The problem of epistemology [1936]. In R.S. Cohen and T. Schnelle, editors, Cognition and Fact - Materials on Ludwik Fleck, pages 81-82. Dordrecht, Reidel, 1986.

[FLGP02] M. Fernández-López and A. Gómez-Pérez. The Integration of OntoClean in WebODE. EON2002 Evaluation of Ontology-based Tools. In Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002), Sigüenza, Spain, 2002.

[FLGP04] M. Fernández-López and A. Gómez-Pérez. Searching for a time ontology for semantic web applications. In A.C. Varzi and L. Vieu, editors, Formal Ontology in Information Systems, pages 331-341. IOS Press, 2004.

[FMG06] B. Fortuna, D. Mladenic, and M. Grobelnik. Semi-automatic construction of topic ontologies. In Berendt et al., editor, Knowledge Discovery and Ontologies. Springer, 2006.

[Gal91] A. Galton. Reified Temporal Theories and How To Unreify Them. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1991.

[Gan04] A. Gangemi. Reusing semi-structured terminologies for ontology building: A realistic case study in fishery information systems. Deliverable of the EU FP5 WonderWeb project, 2004. http://wonderweb.semanticweb.org/deliverables/D16.shtml.

[Gan05] A. Gangemi. Ontology Design Patterns for Semantic Web Content. In Musen et al., editor,Proceedings of the Fourth International Semantic Web Conference. Springer, 2005.

[GBCL05] A. Gangemi, S. Borgo, C. Catenacci, and J. Lehmann. Task taxonomiesfor knowledge content. Deliverable D07 of the Metokis Project, 2005.http://www.loa-cnr.it/Papers/D07_v21a.pdf.

[GCB04] A. Gangemi, C. Catenacci, and M. Battaglia. The inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In Ontologies in Medicine, pages 64-80. IOS Press, Amsterdam, 2004.

[GCCL05] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Ontology evaluation: A review of methods and an integrated model for the quality diagnostic task. Technical report, Istituto di Scienze e Tecnologie della Cognizione, CNR, Roma, 2005. http://www.loa-cnr.it/Publications.html.

[GCCL06a] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Modelling Ontology Evaluation and Validation. In Proceedings of the Third European Semantic Web Conference. Springer, 2006.

[GCCL06b] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann. Qood grid. A metaontology-based framework for ontology evaluation and selection. In Proceedings of the EON'2006 Workshop, 2006. http://km.aifb.uni-karlsruhe.de/ws/eon2006/eon2006gangemietal.pdf.

[GFK+04] A. Gangemi, F. Fisseha, J. Keizer, I. Pettman, and M. Taconet. A Core Ontology of Fishery and its use in the Fishery Ontology Service Project. In A. Gangemi and S. Borgo, editors, CEUR-WS Proceedings (First International Workshop on Core Ontologies, EKAW Conference), volume 118, 2004.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995.

[GKRT04] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In Proceedings of the Thirteenth International World Wide Web Conference, pages 403-412, New York, NY, 2004.

[GM03] A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations.In CoopIS/DOA/ODBASE, pages 689-706, 2003.

[GM06] M. Grobelnik and D. Mladenic. Knowledge discovery for ontology construction. In J. Davies,R. Studer, and P. Warren, editors, Semantic web technologies: trends and research inontology-based systems, pages 9-27. John Wiley & Sons, Chichester, 2006.

[GN02] M. Girvan and M.E.J. Newman. Community structure in social and biological networks. InProc. Natl. Acad. Sci. USA, volume 99, pages 7821-7826, 2002.

[God94] M. Godwin. Nine principles for making virtual communinties work. Wired, 2(6):72-73, 1994.

[Gog93] J.A. Goguen J.A. Social issues in requirements engineering, In Proceedings of the 1st Intl. Symposium on Requirements Engineering, San Diego, Ca, pp. 194-198 (1993).

[GP06] A. Gangemi and V. Presutti. The bourne identity of a web resource. Architecture and Philosophy of the Web Identity, Reference, and the Web (IRW2006), WWW2006 Workshop,Edinburgh, Scotland, May, 23rd 2006. http://www.ibiblio.org/hhalpin/irw2006/presentations/-vpresutti.pdf.

[GPMM03] A. Gomez-Perez and D. Manzano-Mancho. A survey of ontology learning methods and tech-niques. Deliverable 1.5 OntoWeb project, 2003.

[GPS99] A. Gangemi, D.M. Pisanelli, and G. Steve. An overview of the onions project: Applying ontolo-gies to the integration of medical terminologies. Data & Knowledge Engineering, 31:183-220,1999.

[GSB+04] L. Gamberini, A. Spagnolli, L. Bua, P. Cottone, and M. Martinelli. Collaboration in virtual environments: The modulated exploitation of other people as resources. Cognition, Technologyand Work, 6(1):45-48, 2004.

[Guh03] R. Guha. Open rating systems. Technical report, Stanford University, CA, 2003.

[Hep06] M. Hepp. Products and services ontologies: A methodology for deriving owl ontologies fromindustrial categorization standards. Int'l Journal on Semantic Web & Information Systems(IJSWIS), 2(1):72-99, 2006.

[HRWB06] P. Haase, S. Rudolph, Y. Wang, and S. Brockmans. Networked ontology model. Technicalreport, Universität Karlsruhe, 2006.

[HS98] U. Hahn and K. Schnattinger. Towards text knowledge engineering. In Proc. of 15th NationalConference on Artificial Intelligence (AAAI-98), pages 524-531, Menlo Park, CA, 1998. MITPress.

[HS02] U. Hahn and S. Schulz. Turning Lead into Gold? Feeding a Formal Knowledge Base with In-formal Conceptual Knowledge. In Proceedings of 13th Conference on Knowledge Engineering (EKAW). Springer, Berlin, 2002.

[HSC02] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM — Semi-automatic CREAtion of Meta-data. In 13th International Conference on Knowledge Engineering and Knowledge Manage-ment (EKAW02), pages 358-372, Siguenza, Spain, 2002.

[HSV03] S. Handschuh, S. Staab, and R. Volz. On deep annotation. In Proceedings of the 12th Inter-national World Wide Web Conference, Budapest, 2003.

[IPV06] Angelo Di Iorio, Valentina Presutti, and Fabio Vitali. Wikifactory: An ontology-based applicationfor creating domain-oriented wikis. In Sure and Domingue

[IRE] Identity of resource and entity on the web ontology. http://wiki.loa-cnr.it/index.php/LoaWiki:IRE.

[JRS97] M. Jarke, C. Rolland, and A.G. Sutcliffe (eds.). The NATURE of Requirements Engineering. Springer-Verlag (1997).

[Kar99] H. Karsten. Collaboration and collaborative information technologies: A review of the evidence.Database, 30(2):44-65, 1999.

[KH01] P. Kogut and W. Holmes. AeroDAML: Applying Information Extraction to Generate DAML An-notations from Web Pages. In First International Conference on Knowledge Capture (K-CAP 2001), Workshop on Knowledge Markup and Semantic Annotation, Victoria, B.C., 2001.

[KN03] M. Klein and N. Noy. A Component-Based Framework for Ontology Evolution. In Proceedings of the Workshop on Ontologies and Distributed Systems, 2003.

[Kol96] P. Kollock. Design Principles for Online Communities. In Proceedings of the Harvard Confer-ence on Internet and Society, 1996.

[KR70] W. Kunz and H. Rittel. Issues as elements of information systems. Technical report, Instituteof Urban and Regional Development, University of California, Berkeley, 1970. Working Paper131.

[LL91] J. Lee and K. Lai. What's in design rationale. Human-Computer Interaction (Special Issue onDesign Rationale), 6(3-4):251-280, 1991.

[LMU06] V. Lopez, E. Motta, and V. Uren. PowerAqua: Fishing the Semantic Web. In Proceedings ofthe Third European Semantic Web Conference (ESWC06), 2006.

[Lou01] M. Louwerse. An analytic and cognitive parametrization of coherence relations. CognitiveLinguistics, 12(3):291-315, 2001.

[LSNM06] H. Lewen, K. Supekar, N.F. Noy, and M.A. Musen. Topic-Specific Trust and Open Rating Systems: An Approach for Ontology Evaluation. In Proceedings of the 4th International Workshop on Evaluation of Ontologies for the Web (EON2006) at the 15th International World Wide Web Conference (WWW 2006), Edinburgh, UK, 2006.

[LTA04] A. Lozano-Tello and Gómez-Pérez A. Ontometric: A method to choose the appropriate ontol-ogy. Journal of Database Management, 15(2), 2004.

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

[MB05] A. Miles and D. Brickley. SKOS Core Vocabulary Specification. Technical report, W3C WorkingDraft„ 2005.

[MBJK90] Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: a language for representing knowledge about information systems. ACM Transactions on Office Information Systems, 8(4):325-362 (1990).

[MC00] B. Magnini and G. Cavaglià. Integrating Subject Field Codes into WordNet. In Proceedings the Second International Conference on Language Resources and Evaluation (LREC 2000), pages 1413-1418, Athens, Greece, 2000.

[MG04] D. Mladenic and M. Grobelnik. Mapping documents onto web page ontology. In Berendt et al., editor, Web mining: from web to semantic web, volume 3209 of Lecture notes in artificial inteligence/Lecture notes in computer science, pages 77-96. Springer, 2004.

[MGG+04] C. Masolo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. The WonderWeb library of foundational ontologies. Deliverable D18 of the EU FP5 WonderWeb project, 2004. http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf.

[MHG02] D. Maplesden, J.G. Hosking, and J.C. Grundy. Design Pattern Modelling and Instantiation using DPML. In Proceedings of the Tools Pacific 2002, Sydney, 2002. CRPIT Press.

[Mil06] S. Miles. Electronically Querying for the Provenance of Entities. In Proceedings of the Inter-national Provenance and Annotation Workshop, 2006.

[ML00] E. Motta and W. Lu. A library of components for classification problem solving. Ibrow project ist-1999-19005: An intelligent brokering service for knowledge-component reuse on the world-wide web, deliverable 1, 2000.

[MP01] Andrés Montoyo and Manuel Palomar. Specification marks for word sense disambiguation:New development. In CICLing, pages 182-191, 2001.

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

[MT88] W.C. Mann and S.A. Thompson. Rhetorical structure theory: Toward a functional theory of textorganisation. Text, 8(3):243-281, 1988.

[MTC+02] D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. Architectural Elements of Language Engineering Robustness. Journal of Natural Language Engineering - Special Issue on Robust Methods in Analysis of Natural Language Data, 8(2/3):257-274,2002. http://gate.ac.uk/sale/robust/robust.pdf.

[MVB+04] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social roles and their descriptions. In D. Dubois, C.A. Welty, and M.-A. Williams, editors, KR, pages 267-277. AAAI Press, 2004.

[MVVD+02] E. Motta, M. Vargas-Vera, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), pages 379-391, Siguenza, Spain, 2002.

[MYBM91] A. MacLean, R.M. Young, V.M.E. Bellotti, and T.P. Moran. Questions, options, and criteria: Elements of design space analysis. Human-Computer Interaction, 6:201-250, 1991.

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

[NGM05] N.F. Noy, R. Guha, and M.A. Musen. User ratings of ontologies: Who will rate the raters? InProc. of the AAAI 2005 Spring Symposium on Knowledge Collection from Volunteer Contribu-tors, Stanford, 2005.

[Noy04] N.F. Noy. Evaluation by ontology consumers. IEEE Intelligent Systems, pages 1541-1672,2004.

[OLG+06] D. Oberle, S. Lamparter, S. Grimm, D. Vrandecic, S. Staab, and Gangemi A. Towards ontologies for formalizing modularization and communication in large software systems. Journal of Applied Ontology, 2006. To appear.

[OMGS04] D. Oberle, P. Mika, A. Gangemi, and M. Sabou. Foundations for service ontologies: Aligning OWL-S to DOLCE. In S. Staab and P. Patel-Schneider, editors, Proceedings of the World Wide Web Conference (WWW2004), volume Semantic Web Track, 2004.

[Ost90] E. Ostrom. Governing the Commons: The Evolution of Institutions for Collective Action. Cam-bridge University Press, New York, 1990.

[PB04] R. Porzel and M. Baudis. The Tao of CHI: Towards Effective Human-Computer Interaction. InProceedings of HLT-NAACL, pages 209-216, 2004.

[PGS98] D.M. Pisanelli, A. Gangemi, and G. Steve. An ontological analysis of the UMLS metathesaurus. J. of American Medical Informatics Association, 5:810-814, 1998.

[PK91] D.E. Perry and G.E. Kaiser. Models of software development environments. IEEE TransactionsOn Software Engineering, 17(3):283-295, 1991.

[PKK+04] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM - SemanticAnnotation Platform. Natural Language Engineering, 2004.

[PM04] R. Porzel and R. Malaka. A Task-based Approach for Ontology Evaluation. In Proceedings ofECAI04, 2004.

[Poh93] K. Pohl. The three dimensions of requirements engineering. Proceedings of the 5th Intl. Conf. Advanced Information Systems Engineering, pp. 275-292, Paris, France (1993).

[POT70] C. Perelman, L. Olbrechts-Tyteca Traite de l'argumentation. La nouvelle rhetorique, Bruxelles, editions de l'Universite de Bruxelles, 1970.

[Pro78] P. Procter. Longman Dictionary of Contemporary English. Longman Group Limited. Harlowand London, 1978.

[PSL90] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs.SIAM J. Matrix Anal. Appl., 11:430-452, 1990.

[PSLP03] C. Patel, K. Supekar, Y. Lee, and E.K. Park. OntoKhoj: A Semantic Web Portal for OntologySearching, Ranking and Classification. In Proceedings of the Workshop On Web InformationAnd Data Management, ACM, 2003.

[PST04] S. Pinto, S. Staab, and C. Tempich. DILIGENT: Towards a fine-grained methodology for Distributed Loosely-controllled and evolvInG Engineering of oNTologies. In Proceedings of ECAI-2004,2004.

[RCC+03] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. Preprint cond-mat/0309488, 2003.

[RD04] M. Richardson and P. Domingos. Combining link and content information in web search. InM. Levene and A. Poulovassilis, editors, Web Dynamics, pages 179-193. Springer, New York,2004.

[Rei00] J.R. Reich. Ontological Design Patterns: Modelling the Metadata of Molecular Biological On-tologies, Information and Knowledge. In Proceedings of DEXA 2000, 2000.

[RHAS00] W.C. Regli, X. Hu, M. Atwood, and W. Sun. A survey of design rationale systems: Approaches, representation, capture and retrieval. Engineering with Computers, 16:209-235, 2000.

[RN03] CA Reed and T.J. Norman, Argumentation Machines, Kluwer Academic Publishers, 2003

[RR04] A.L. Rector and J. Rogers. Patterns, Properties and Minimizing Commitment: Reconstruction of the GALEN Upper Ontology in OWL. In A. Gangemi and S. Borgo, editors, CEUR-WS Proceedings (First International Workshop on Core Ontologies, EKAW Conference), volume 118,2004.

[RW73] H. Rittel and M. Webber. Dilemmas in a general theory of planning. Policy Sciences, 4:155-169,1973.

[SD06] York Sure and John Domingue, editors. The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings, volume 4011 of Lecture Notes in Computer Science. Springer, 2006.

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

[SH94] S.B. Shum and N. Hammond. Argumentation-based design rationale: What use at what cost? International Journal of Human-Computer Studies, 40(4):603-652, 1994.

[SLMU06] M. Sabou, V. Lopez, E. Motta, and V. Uren. Ontology Selection: Ontology Evaluation on theReal Semantic Web. In Proceedings of the EON'2006 Workshop, collocated with WWW'06,Edinburgh, 2006.

[SMD02] S.B. Shum, E. Motta, and J. Domingue. Augmenting Design Deliberation with Compendium: The Case of Collaborative Ontology Design. In Proceedings of the Workshop on Facilitating Hypertext Collaborative Modelling in conjunction with ACM Hypertext Conference, Maryland, 2002.

[Sos03] D. Soshnikov. Ontological Design Patterns in Distributed Frame Hierarchy. In Proceedingsof the 5th International Workshop on Computer Science and Information Technologies, Ufa,Russia, 2003.

[Spy05] P. Spyns. Evalexon: Assessing triples mined from texts. Technical report 09, STAR Lab,Brussel, 2005.

[SSM04] B. Sereno, S.B. Shum, and E. Motta. Claimspotter: An Environment to support Sensemaking with Knowledge Triples. In Proceedings of the Intelligent User Interfaces Conference, IUI2005, San Diego, CA, USA, 2004.

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

[SSV02a] L. Stojanovic, N. Stojanovic, and R. Volz. A Reverse Engineering Approach for migrating data-intensive web sites to the semantic web. In Proceedings of Intelligent Information Processing,Montreal, 2002.

[SSV02b] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In Proceedings of the 17th ACM symposium on applied computing (SAC), pages 1100-1107. ACM Press, 2002.

[STV+06] Y. Sure, C. Tempich, D. Vrandecic, S. Pinto, E. Paslaru Bontas, and M. Hefke. Sekt methodology: Initial framework and evaluation of guidelines. Sekt deliverable d7.1.2, 2006. http://www.sekt-project.org.

[SV04] C. Strapparava and A. Valitutti. WordNet-Affect: an affective extension of WordNet. In Proceedings ofthe 4th International Conference on Language Resources and Evaluation (LREC 2004), pages 1083-1086, Lisbon, 2004.

[Sva04] V. Svatek. Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In Proceedings of the 7th Conference on Business Information Systems, Poznan, 2004.

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

[The] The Description and Situation ontology. http://www.loa-cnr.it/ontologies/ExtendedDnS.owl.

[Tou58] S. Toulmin. The Uses of Arguments. Cambridge University Press, 1958.

[TPSS05] C. Tempich, S. Pinto, Y. Sure, and S. Staab. Argumentation Ontology for DIstributed, Loosely-controlled and evolvIng Engineering processes of oNTologies (DILIGENT). In Proc. of ESWC2005- European Semantic Web Conference, Heraklion, Crete, 2005. Springer.

[UBSMGL04] V Uren, S. Buckingham Shum, C. Mancini, and V.U. Gangmin Li. Modelling Naturalistic Argumentation in Research Literatures. In Proceedings of the 4th Workshop on Computational Models of Natural Argument, 2004.

[vAGS06] M. van Assem, A. Gangemi, and G. Schreiber. Conversion of WordNet to a standard RDF/OWL representation. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), Genova, Italy, 2006.

[vdAtHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. Distributed and Parallel Databases, 14:5-51, 2003.

[Vea06] D. Vrandecic and et al. Sekt methodology: Initial lessons learned and tool design. Deliverabled7.2.1 of the sekt project, 2006.

[VHSS04] R. Volz, S. Handschuh, S. Staab, and R. Studer. Ontolift demon-strator. Deliverable of the eu fp5 wonderweb project, 2004.http://wonderweb.semanticweb.org/deliverables/D12.shtml.

[VO06] Max Völkel and Eyal Oren. Towards a wiki interchange format (wif). In First Workshop onSemantic Wikis - From Wiki To Semantics, Budva, Montenegro, 12.06.06, 2006.

[Vos98] P. Vossen. EuroWordNet: a Multilingual Database with Lexical Semantic Networks. KluwerAcademic Publishers, 1998.

[Vra05] D. Vrandecic. Explicit knowledge engineering patterns with macros. In C. Welty andA. Gangemi, editors, Proceedings of the Ontology Patterns for the Semantic Web Workshopat the ISWC 2005, Galway, Ireland, 2005.

[VG06] Denny Vrandecic, Aldo Gangemi, Unit Tests for Ontologies, In: OnToContent Workshop. M. Jarrar (Ed). 2006. Springer-Verlag: Berlin [Wel05] C. Welty. Semantic web best practices and deployment working group. Technical report, W3C Task Force on Ontology Engineering Patterns, 2004-2005. Description of work, archives, W3C Notes and recommendations available fromhttp://www.w3.org/2001/sw/BestPractices/OEP/.

[WF94] S. Wasserman and K. Faust. Social Network Analysis: Methods and Applications. Cambridge University Press, UK, 1994.

[WG01] C. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. Data andKnowledge Engineering, 39(1):51-74, 2001.

[Wik] Semantic Media Wiki. Semantic MediaWiki Home Page.http://meta.wikimedia.org/wiki/Semantic_MediaWiki.

[WVS05] Steven Willmott, Gerard Vreeswijk, Matthew South (editors). AIF: Argumentation Interchange Format Strawman Model. Version 0.7, November 5th, 2005

[YOE05] H. Yao, A.M. Orme, and L. Etzkorn. Cohesion metrics for ontology design and application.Journal of Computer Science, 1(1):107-113, 2005.