C-ODO: an OWL meta-model for collaborative ontology design

Aldo Gangemi
Laboratory for Applied Ontology
National Research Council
(ISTC-CNR)
Roma, Italy
aldo.gangemi@istc.cnr.it

Valentina Presutti
Laboratory for Applied Ontology
National Research Council
(ISTC-CNR)
Roma, Italy
valentina.presutti@istc.cnr.it

Carola Catenacci
Laboratory for Applied Ontology
National Research Council
(ISTC-CNR)
Roma, Italy
carola.catenacci@istc.cnr.it

Jos Lehmann
Laboratory for Applied Ontology
National Research Council
(ISTC-CNR)
Roma, Italy
jos.lehmann@istc.cnr.it

Malvina Nissim
Laboratory for Applied Ontology
National Research Council
(ISTC-CNR)
Roma, Italy
nissim@loa-cnr.it

ABSTRACT

The design and maintenance of ontologies is a complex social collaborative activity, and this is true especially for semantic-web ontologies. On the one hand, such activity calls for the availability of tools providing support to typical operations such as the *reuse* of existing ontologies and design patterns, the *re-engineering* of thesauri, lexicons, folksonomies, database schemas, and knowledge from corpora, or to the appropriate evaluation and selection processes which are needed in order to make an ontology functional to a given task. On the other hand, tools able to support the collaborative performance of all these operations, aiding e.g. the *discussion* and consensus-reaching processes on an ontology element and its *rationale*, should be provided too. Current tools substantially fail to address both types of need. In our opinion, this is partly due to the lack of both an adequate requirement analysis, which describes the actual processes and data that are usually managed during ontology-design activities, and a unifying conceptual framework, which puts together the several interrelated aspects of (collaborative) ontology design. In this paper we present a formal framework that represents the notions needed to express requirements for the development of collaborative ontology engineering tools. The framework is formalized as an OWL(DL) ontology named C-ODO (Collaborative Ontology-Design Ontology), and is being used within the EU NeOn project.

1. INTRODUCTION

Collaborative ontology design cannot be specified unequivocally, e.g. as an OWL class, because the entities that are typically referred to by the term 'design' can be multifaceted. Usually, the talk about ontology design addresses several aspects that are highly interrelated: how the ontology does look like; its conceptualization; which principles have been used to build it; the process model that has been used to create it; and so on. In the context of the EU NeOn integrated project, collaborative design of networked ontologies in heterogeneous social contexts is a major issue. The work presented in this paper is the result of a preliminary requirements analysis for collaborative design that has been conducted in NeOn, and is extensively reported in [3].

WWW2007, May 8–12, 2007, Banff, Canada.

The main focus of the paper is a formal framework to be used as a requirement language for the 'social level' aspects of ontology design. We assume that the specification at the computational level should mirror the social requirements of ontology design, and that this can be done in one of two ways: i) by substituting social-level tasks with computational tasks, or ii) by assisting social-level tasks specified as proxies within a workflow of computational tasks. For example, a method for ontology evaluation can be described formally at the social level, and, when the evaluation is limited to the structural aspects of an ontology, most tasks can be accomplished by an implemented algorithm. On the other hand, if evaluation at some point requires an active decision role from a human agent, that role is proxied by the tool. Being clear about which (and how) social-level methods or tasks are substituted, and which methods, roles or tasks are proxied may improve the semantic interoperability between tools and social practices.

In the remainder of section 1, we introduce the basic organization of the framework and the notions treated. A unifying framework for describing ontology design should be general enough to express all the possible approaches to this activity, and should also be practical enough to be implemented without creating unnecessary complexity in local solutions, models, and tools. Moreover, it should not consist of a single particular methodology, but rather it should provide expressivity enough to describe different methods or aggregations of methods. We consider ontology design in terms of its objective, scope, components, and supporting functionalities.

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 the design of an ontology. In analogy with the blank page effect experienced by

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 which is 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 these kinds of data, the focus of this paper is on ontology design proper [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 rather 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, which are depicted in figure 1:

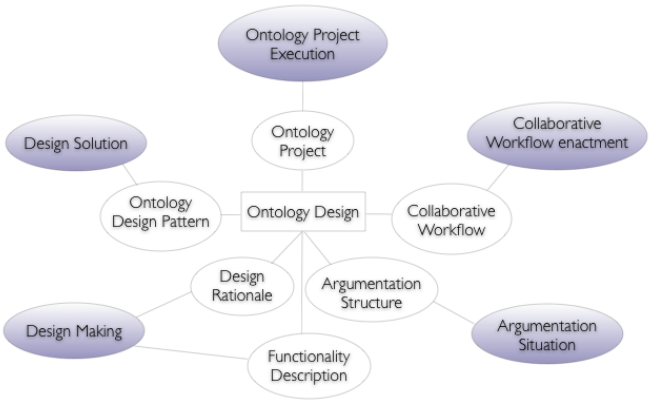

Figure 1: Ontology Design Components

- Ontology project: a project having the goal of influencing the lifecycle of a networked ontology
- Collaborative workflow: a special case of epistemic workflow, i.e. a relationship between rational agents that influences the knowledge of one or more agents in the relationship, according to a workflow shared by one, more or even all the agents. A collaborative epistemic workflow is characterized by the ultimate goal of designing networked ontologies and by specific relations among designers, ontology elements, and collaborative tasks
- Argumentation: a structure for discussing possible design solutions, based on rationales and dialectic rules
- Ontology design rationale: the motivations according to which an ontology is designed the way it is
- Functionality description: a description of a task to be accomplished by a design operation according to a method
- Design Pattern: a configuration of ontology elements that is relevant from the logical, architectural, or conceptual viewpoint.

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

- 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 platform developers with a requirement analysis.
- 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 platform.

Finally, we consider the functionalities that (are needed to) support collaborative ontology design. The list of functionalities from the NeOn project includes evaluation, selection, re-engineering, learning, upgrading database content, mapping, collaborative workflow, argumentation, provenance, data annotation, social network analysis, lexical domains, ontology localization, and multilingual ontology integration. Each of these functionalities is represented in two ways in C-ODO. On the one hand, these are all 'rich' functionality descriptions, in which roles, goals, parameters and complex tasks are put together to compose a 'method': e.g., a method to execute ontology selection, argumentation, or concept selection by means of lexical domain filtering and linking to other ontologies. On the other hand, these functionalities are also (complex) tasks, defined in the related functionality descriptions, to be be performed within an ontology project.

The rest of the paper is organized as follows: section 2 provides a brief overview of the existing work on requirements and tools for collaboration in cooperative knowledge communities. Section 3 describes the foundations of our conceptual framework, which is the main contribution of this paper and is encoded in the Collaborative Ontology-Design Ontology (C-ODO). Section 4 presents C-ODO with the help of a simple example. In section 5, a further example illustrates the way C-ODO can be used to model a collaborative ontology-design methodology. Finally, in section 6 some conclusions are drawn and lines for future work are sketched.

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

2. RELATED WORK

A huge amount of work has been conducted on requirements and tools for collaboration in cooperative knowledge communities. For an extended discussion of the state-of-art, the reader can refer to [3]. For space reasons, here we only cite some of the most relevant contributions.
Some works address social aspects to be considered when building tools [1], [15]. [1] clearly states the problems coming from the social-technical divide. [13] is a brief survey of the main studies on principles that seem to underline successful cooperative communities. The DILIGENT methodology provides several requirements for supporting collaborative workflows [31], [26], and deals with argumentation [25]. In [18], further requirements for collaborative ontology development are identified. Available tools include: Ontology Builder and Ontology Server (OBOS) [6], Hypertext-Augmented Collaborative Modelling (HACM) [24], Claimspotter [22], ClaiMaker [15], which is based on [16], and Co-OPR [23], which presents the integration of two existing tools (i.e., Compendium, and I-X). Furthermore, [14] is a source of interesting points with respect to requirements and tool support, while [21], and [4] analyze aspects of human-computer interaction (HCI). Finally, as far as coordination in collaborative environments is concerned, interesting suggestions are provided by the coordination models and workflow patterns presented, resp., in [19] and [28].

3. THE COLLABORATIVE ONTOLOGY DESIGN ONTOLOGY (C-ODO)

As pointed out in Section 1, the notions of collaboration and 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, we introduce here the Collaborative Ontology-Design Ontology (C-ODO). C-ODO formally specifies the components of collaborative ontology design. It allows formal expression of either social or computational requirements for tools that support ontology design. C-ODOs main components - as already informally presented 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 are briefly summarized in the following subsection.

3.1 Assumptions and reused ontologies in C-ODO

C-ODO's design has pivoted on three rationales: i) basing the whole framework on a reification mechanism, which is founded on the distinction between descriptions and situations; ii) breaking down the conceptualization modeled in the framework into six main layers (ontology project, collaborative workflow, argumentation, design rationale, functionality description, ontology design pattern); iii) reusing as many as possible existing ontologies as foundations for C-ODO.

Reification through Descriptions and Situations We have based C-ODO on a reification mechanism. The form of reification adopted in our framework makes it possible to talk in the same language of both a generic method and the elementary or complex entities that allow to perform that method, since both the method and its component entities are in the same domain of interpretation This allows, for instance, to design a set of operations like the composition of: {elicit knowledge from a colleague or an expert, find and specialize design patterns for that knowledge, validate the adapted patterns against competence questions}. This makes the language more expressive without making it computationally more complex as usual with reification. The form of reification adopted in C-ODO is based on the distinction between descriptions (e.g. a method) and the situations that satisfy a description (e.g. the configuration of operations that allow to perform that method). This architectural pattern is called DnS [10]. Descriptions 'use' concepts, e.g. the role of 'being an expert during elicitation', while situations are 'setting for' entities, e.g. an individual expert or an operation. Concepts are used to 'classify' entities within a situation.
For the purpose of intuition, the distinction between descriptions and situations can be understood as a generalization of the UPML (Unified Problem-solving Method Development Language) paradigm [17], 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 (as ontological entities) are the counterpart of a set of criteria in UPML, while situations are the counterpart of a solution in UPML. Furthermore, situations are settings for a set of entities and their relations, which satisfy the concepts devised in the description. Therefore, related entities in the setting of a situation may be considered the counterpart of UPML observables.

Plans, tasks and goals Extensions of DnS have been used to model several types of conceptualizations, such as: social agents like organizations, communities of agents [2], the information objects by which a description is expressed [7], the time-spans characterizing the situations, etc.
An important extension to DnS is the Plan ontology [7], which models plans as descriptions that represent sequences of tasks that lead from a given situation to a new, expected one. 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. The main components of plans are: task, goal and agent-driven role. Tasks and agent-driven roles are types of concepts, while goals are descriptions, proper parts of plans. Control constructs (e.g. choose between the following alternatives) from traditional planning and workflows are represented as control tasks, also defined within plans. Tasks may be connected to roles by a 'target' relation. Plans may also have situations as pre- or post-conditions. Plans as descriptions are different from plan executions: the latter are situations. Goal situations are situations that satisfy a goal.

Other reused ontologies Other reused ontologies include the Open Systems Ontology [7], which includes the object transformation pattern, involving roles for performers, resources, working items, and products; the oQual ontology [9, 8], which models ontology evaluation and selection as a diagnostic tasks over ontology elements, processes, and attributes; the Ontology Metadata Vocabulary (OMV) [12], which captures several notions conveying key-information on an ontology (ontology task, ontology engineering tool, location, party, ontology engineering methodology, ontology language, person, etc.); the notions of Network of Ontologies and Networked Ontology [12], i.e.: "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 Ontologies."

3.2 Six layers of ontology-design representation

As shown in Fig. 1 we have distinguished six layers that fraction the scope of design into ordered components, or modules: ontology project, collaborative workflow, argumentation, design rationale, functionality description, and ontology design pattern. The six layers are a componential structure, and do not imply a given direction in ontology design, i.e. they do not tattempt to prescribe a methodology that starts from project planning and ends with practical implementation, or the other way around. The six-layering is simply agnostic with respect to sequence of execution. Each of these six layers are modeled in terms of the distinction between descriptions and situations. For instance, if we take ontology design pattern as a starting point, we have to distinguish between its descriptions and situations. Design-pattern descriptions, or schemas, include roles, tasks, and parameters for encoding part of an ontology in a certain logical language. The situational counterpart of ontology design patterns are design solutions, i.e. the states of part of an ontology at some versioning point.

On their turn, design-pattern schemas and choices are motivated by a design rationale, and are applied by performing some functionality. Design rationales and functionality methods are descriptions, whose counterpart are actual design making situations.

Rationales and functionalities are made explicit during an argumentation situation, the latter being the situational counterpart of an argumentation structure. Argumentation situations typically occur during collaborative workflow enactments (situations) that follow some collaborative workflow (description).

Such workflows are part of an ontology project (description), whose counterpart is an ontology project execution (situation).

C-ODO ontology is an OWL(DL) ontology, way too *large* (127 classes and 58 properties) to be completely presented in a conference paper. From [5] C-ODO can be downloaded for extensive browsing. Here we use a guiding example to provide an intuition of what C-ODO is and how it can be used. We also provide some sample formal definition for the main C-ODO's components. For a full textual presentation of C-ODO, the reader can refer to [3].

4. DESCRIBING ONTOLOGY DESIGN WITH C-ODO

As an example, consider a team of designers that are collaboratively designing an ontology from a flat list of ten classes, including {Dog, Canine, ...}, and that are willing to apply the *subsumption* logical pattern to Dog. The implicit choice space is made of ten choices (nine possible superclasses, or being a top class). How to decide what class subsumes Dog? Reusing existing artifacts is a practice that advantages designers of any field (e.g., software, business, etc.) in undertaking their tasks. The same applies to ontology designers that can reuse previously produced ontologies, parts of them, or best practices in ontology representation. In our example, the decision on subsumption should be based on an explicit rationale suitable to subsumption. For example, the most typical one is 'extensional semantics'. In practice, it consists in assuming that choosing e.g. *Canine subsumes Dog*, depends on the fact that Dog's instances are all Canine's instances. Of course, the very same design solution can be motivated by a different rationale, for example, assuming dictionaries like WordNet (where Canine is a hypernym of Dog) to be a good motivation for making subsumption relations. In other words, the design pattern (in the example, the *subsumption* logical pattern) represents the 'quality' of a certain solution, whose rationale is 'extensional semantics', which leads to the design solution *Canine subsumes Dog*, on the basis that all Dog's instances are also Canine's instances. One aim of our work is to provide people involved in the design of ontologies with a way to represent the rationales that characterize their design decisions (e.g., design patterns chosen for a certain version of an ontology), and how these rationales are discussed within an argumentation session. We distinguish between solutions adopted and reasons behind such choices, and assume that design rationales represent reasons while design patterns are the solutions adopted. In C-ODO, the *subsumption* pattern can be represented by instantiating the class design-pattern schema. Design-pattern schemas are special cases of 'qoods', i.e. quality-oriented ontology descriptions which describe what is needed in an ontology and its use case(s) to be appropriate to some criterion. An ontology design-pattern schema is satisfied only by a situation that is the setting for some ontology element and their axioms (e.g., stating the axiom *Dog subClassOf Canine* using the OWL language). The design-pattern schema formalization and its situation counterpart are expressed by axioms 1, and 2.

$$\begin{aligned} DesignPatternSchema \sqsubseteq\ & oqual{:}qood \sqcap \\ \forall\ & edns{:}satisfiedBy(edns :: Situation \sqcap \\ & \exists\ edns{:}settingFor\ OntologyElement) \end{aligned} \tag{1}$$

$$\begin{aligned} DesignSolution \sqsubseteq\ & edns{:}Situation \sqcap \\ \exists\ & edns{:}satisfies\ DesignPatternSchema \sqcap \\ \exists\ & edns{:}settingFor\ FormalExpression \sqcap \\ \exists\ & edns{:}settingFor\ OntologyElement \end{aligned} \tag{2}$$

In general, design patterns [11] can be of several kinds: macro of logical languages, in the sense of [30]; architectural design patterns, which are formal expressions for solving structural issues (macros are simple examples of architectural patterns); content design patterns, which are typed (i.e., they refer to a non-logical vocabulary), and al-

low designers to solve domain specific issues; ontology anti-patterns, which identify wrong solutions to recurrent design issues. Design pattern schemas are used in order to guide designers in using ontology design patterns. As a matter of fact, the choice can be motivated by different rationales, e.g. the axiom *Canine subsumes Dog* 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 superclass. Still another rationale is 'best approximation semantics' to a repository of content design patterns, so that the best candidate as a superclass may be chosen from an existing content pattern that includes Dog and Canine, with a best similarity match. One of C-ODO's main layers is design rationale, which contains the classes: i) ontology design rationale, and ii) design making. According to [20], design rationales 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 artifact i.e., all the information that can be crucially valuable to various people who have to deal with the artifact. Design rationales are then explicit statements (or sets of statements) which represent the reasons why an object (product) has been (or is being) designed the way it is. 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. Axioms 3 and 4 formalize the concepts of ontology design rationale and design making, respectively.

$$\begin{aligned} OntologyDesignRationale \sqsubseteq\ & DesignRationale \sqcap \\ & \forall\ edns{:}satisfiedBy\ DesignMaking \sqcap \\ & \exists\ edns{:}usesConcept\ sys{:}PerformerRole \sqcap \\ & \exists\ edns{:}usesConcept\ edns{:}AgentDrivenRole \sqcap \\ & \exists\ edns{:}usesConcept\ sys{:}KnowledgeRole \sqcap \\ & \exists\ edns{:}usesConcept\ Functionality \end{aligned} \quad (3)$$

$$\begin{aligned} DesignMaking \sqsubseteq\ & edns{:}Situation \sqcap \\ & \exists\ edns{:}satisfies\ DesignRationale \sqcap \\ & \exists\ edns{:}satisfies\ FunctionalityDescription \sqcap \\ & \exists\ edns{:}component\ DesignSolution \sqcap \\ & \exists\ edns{:}settingFor\ edns{:}RationalAgent \sqcap \\ & \exists\ edns{:}settingFor\ DesignOperation \sqcap \\ & \exists\ edns{:}settingFor\ inf{:}InformationObject \end{aligned} \quad (4)$$

Design makings are realized by means of the implementation of some functionality. C-ODO defines the notion of functionality as a description which can either be a generic description of a goal, or an actual plan describing a cognitive or computational method to achieve that goal. A functionality, however, is also a (complex) desired task to be performed (e.g., ontology evaluation). Such task is accomplished through a design operation (i.e., an action in the context of a design making). Consider as an example the task of evaluating the goodness of two given ontologies against a set of competency questions. Such functionality can be performed by following different approaches, such as quantitative measures e.g., the number of terms in the questions that match the vocabulary of the two ontologies, or qualitative evaluations (see [9] for a comprehensive framework on ontology evaluation types), etc. Both approaches can be represented as functionality descriptions that define a method to accomplish the evaluation functionality. Axioms 5 and 6 formalize the notion of functionality.

$$\begin{aligned} FunctionalityDescription \doteq\ & edns{:}Plan \sqcap \\ & \exists\ edns{:}defines\ Functionality \end{aligned} \quad (5)$$

$$\begin{aligned} functionality \sqsubseteq\ & plan{:}Task \sqcap \\ \exists\ edns{:}definedIn\ & (FunctionalityDescription \sqcap \\ & \forall\ accomplishedThrough\ DesignOperation) \end{aligned} \quad (6)$$

Going back to our simple example, let us now consider the case when a team of designers actually agrees on how to apply the *subsumption* pattern between Dog and Canine. This agreement must be the result of a discussion between the designers from the team, based on argumentation activities. There are several ways of implementing an argumentation workflow. A scenario using argumentation theory for ontological support might follow the characterization proposed e.g. in [29]. In the framework proposed by these authors, an argumentation session (which is a kind of collaborative workflow in C-ODO) is a 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 a 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 following classes: i) argumentation structure (made up of dialectical rules: claiming, agreeing, disagreeing, refusing, etc.); ii) argumentation situation (any situation including designers arguing on ontology design solutions); iii) 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 iv) argumentation task (any task to be accomplished within an argumentation situation). As an example, the generic framework in [29] is represented in the class: argumentation session schema, which is a kind of collaborative workflow, and can be enacted as an argumentation session within an argumentation situation. Argument sessions include four tasks corresponding to the components of the framework: i) choice confrontation; ii) rationale declaration; iii) dialectic rule (application); and iv) argument resolution. Axioms 7 and 8 formalize the notions of argumentation structure and argumentation situation, respectively. Regardless to the method applied for achieving an agreement on a design solution, and what rationales have been provided to argument such choice, all the tasks described above are performed by means of some functionality in the context of some collaborative workflow, e.g. an ontology design methodology. Consider an example in which the team we are talking about is organized as a set of peers, working in a collaborative fashion with a non-hierarchical policy. All decisions have to be taken together by means of a given set of rules (e.g., the argumentation structure described above),

upon which peers must agree.

$$\begin{aligned} ArgumentationStructure \sqsubseteq\ & edns{:}Description \sqcap \\ & \exists\ edns{:}component\ DesignRationale \sqcap \\ & \exists\ edns{:}usesConcept\ sys{:}PerformerRole \sqcap \\ & \exists\ edns{:}usesConcept\ ArgumentationRole \end{aligned} \quad (7)$$

$$\begin{aligned} ArgumentationSituation \sqsubseteq\ & edns{:}Situation \sqcap \\ & \forall\ edns{:}satisfies\ ArgumentationStructure \sqcap \\ & \exists\ edns{:}component\ DesignMaking \sqcap \\ & \exists\ edns{:}settingFor\ (\exists\ isArguedBy) \end{aligned} \quad (8)$$

This behavior can be described as, and corresponds to, a kind of collaborative workflow. C-ODO does not prescribe a specific methodology, since it aims to provide the primitives needed in order to express any methodology for collaborative ontology design, and the requirements for developing supporting tools. To this end, C-ODO defines the classes: epistemic workflow and epistemic workflow enactment. Bear in mind that a collaborative workflow and a collaboration situation are special cases of epistemic workflow and epistemic workflow enactment, respectively. Any epistemic workflow includes at least one argumentation structure, and is conceived in order to achieving the goal of producing some knowledge (i.e., a knowledge production goal). At least two rational agents (which can be also collectives e.g., teams) have to participate as accountable performers (i.e., being classified by an accountable performer role). Furthermore, at least two knowledge resources are involved which are respectively, the knowledge resource which the two agents start working on, and the artifact resulting from their collaboration.

Epistemic workflow enactment is the situation counterpart of epistemic workflow. Axioms 9 and 10 formally define them.

$$\begin{aligned} EpistemicWorkflow \sqsubseteq\ & edns{:}Plan \sqcap \\ & \exists\ edns{:}component\ ArgumentationStructure \sqcap \\ & \exists\ plan{:}hasMainGoal\ KnowledgeProductionGoal \sqcap \\ & \exists\ edns{:}usesConcept\ AccountablePerformerRole \sqcap \\ & \exists\ edns{:}usesConcept\ KnowledgeProductRole \sqcap \\ & \exists\ edns{:}usesConcept\ KnowledgeResourceRole \sqcap \\ & \exists\ edns{:}usesConcept\ WorkingKnowledgeItemRole \end{aligned} \quad (9)$$

$$\begin{aligned} EpistemicWorkflowEnactment \sqsubseteq\ & \\ & edns{:}PlanExecution \sqcap \\ & \exists\ edns{:}satisfies\ EpistemicWorkflow \sqcap \\ & \exists\ edns{:}component \\ & edns{:}AgentCoParticipationSituation \sqcap \\ & \exists\ edns{:}component\ ArgumentationSituation \sqcap \\ & \exists\ edns{:}settingFor\ (edns{:}RationalAgent \sqcap \\ & \exists\ edns{:}classifiedBy\ accountablePerformerRole) \sqcap \\ & \exists\ edns{:}settingFor\ dol{:}TimeInterval \sqcap \\ & \exists\ edns{:}settingFor\ (edns{:}InformationObject \sqcap \\ & \exists\ edns{:}classifiedBy\ \{ontologyLifecycleProduct\}) \sqcap \\ & \exists\ plan{:}directSuccessor \\ & KnowledgeProductionGoalSituation \end{aligned} \quad (10)$$

Finally, the collaborative design of an ontology, say that of canine kinds, is the goal of a dedicated ontology project composed of a collaborative workflow supported by certain functionalities that allows designers to argument their design decisions by means of design rationales. The ontology project can be executed one or more time according to its description. C-ODO represents the notion of ontology project by defining the classes: ontology project, and ontology project execution. Axioms 11 and 12 formalize this notions.

$$\begin{aligned} OntologyProject \sqsubseteq\ & edns{:}Plan \sqcap \\ & \exists\ edns{:}component\ EpistemicWorkflow \sqcap \\ & \exists\ edns{:}usesConcept\ PerformerRole \sqcap \\ & \exists\ edns{:}usesConcept\ KnowledgeProductRole \sqcap \\ & \exists\ edns{:}usesConcept\ WorkingKnowledgeItemRole \sqcap \\ & \exists\ edns{:}usesConcept\ KnowledgeResourceRole \sqcap \\ & \exists\ edns{:}usesConcept\ Functionality \end{aligned} \quad (11)$$

$$\begin{aligned} OntologyProjectExecution \sqsubseteq\ & edns{:}Situation \sqcap \\ & \exists\ edns{:}satisfies\ OntologyProject \sqcap \\ & \exists\ edns{:}component\ EpistemicWorkflowEnactment \sqcap \\ & \exists\ edns{:}settingFor\ edns{:}RationalAgent \sqcap \\ & \exists\ edns{:}settingFor\ DesignOperation \sqcap \\ & \exists\ edns{:}settingFor\ KnowledgeCollective \sqcap \\ & \exists\ edns{:}settingFor\ edns{:}InformationObject \end{aligned} \quad (12)$$

According to this axiomatization and the definitions given above, we define an ontology project as a plan that is composed of some epistemic workflow and uses some working knowledge object, functionality, performer role, knowledge resource role, and knowledge product role. 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. Figure 2 summarizes the maximal functionalities (use cases) of collaborative ontology design, and how they relate to C-ODO layers.

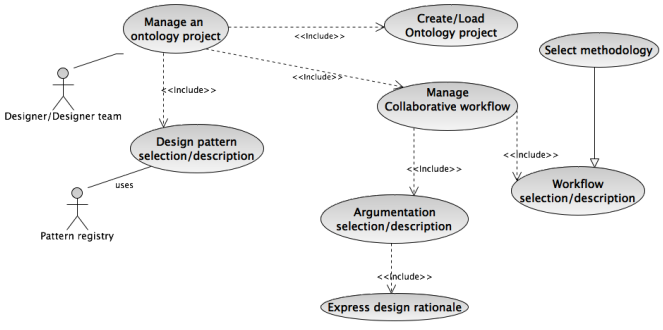

Figure 2: Maximal functionalities in collaborative ontology design

5. AN EXAMPLE OF C-ODO-BASED MODELING

In this section we show an example of C-ODO-based modeling. We model the DILIGENT [27] argumentation method. The model we show is represented by OWL-DL triples that instantiate classes and relations of C-ODO, and it formalizes the implicit ontology of the mentioned approach. However, we do not intend to provide here a complete modeling of its algebraic features.

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. 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. It also includes 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 and 4 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 3.1), which provides a unique domain of discorse for projects, methodologies, argumentation protocols, design rationales, design patterns, and functionalities.

Figure 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 `performerRoles` are used, which 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 `argumentationRoles` played by either a `design-solution` or an `information-object`. `Decision` and `position` roles are targeted to `solutions` or `information`.

Figure 4 shows another view of the `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-rationales` 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`.

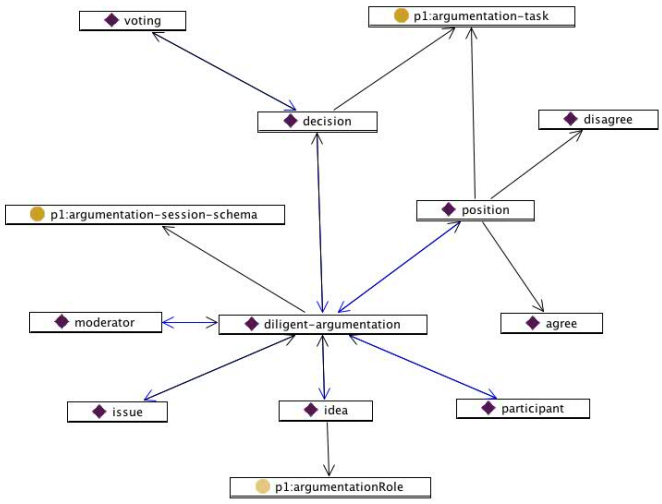

Figure 3: C-ODO-based model of DILIGENT argumentation (focus on tasks)

6. CONCLUSIONS

We have briefly discussed the state of art for collaborative work within cooperative knowledge communities and have shown that there is a lack of generality in this literature. This is due both to the lack of an adequate social requirement analysis, and to the social-technical gap between social requirements and technical feasibility. In particular, none of existing approaches attempts at providing a conceptual framework of the collaborative use of existing or forthcoming technology. The gap is even sharper when collaborative design is applied to ontologies. In order to fill that gap, we have introduced C-ODO (Collaborative Ontology-Design Ontology), the presentation of which is the main contribution of this paper. C-ODO is a framework that represents the notions needed to express requirements for the development of collaborative ontology engineering tools. The framework is formally described as an OWL(DL) ontology. Through a simple example, we have shown C-ODO's main components, their usage and relations, and some of the axioms which formalize the notions it contains. C-ODO can be factorized into six main layers: ontology projects, collaborative workflows, argumentation schemas, design rationales, functionality descriptions, and design patterns. Each layer is represented by assuming a distinction between descriptions (or schemas), and situations, and then adding roles and entity types that describe the world of ontology design. The context of C-ODO is the NeOn project, where a novel approach to ontology design is being defined, with reference to the networked and contextualized dimension of ontologies. The future NeOn platform for ontology design will include tools that allow a rich and highly customizable configuration of components, on the basis of the specific needs of a designer, a team of designers, or distributed teams that work together in an ontology project or in a part of it. In that generic perspective, ontology lifecycles can be much more complex and open-ended than it is usually perceived, and a precise modeling of requirements, at development time or at run time, will be a must. The early application of C-ODO has been realized for the modeling of NeOn functionalities and requirements.

7. ACKNOWLEDGEMENTS

We are grateful to the members of the NeOn consortium who contributed to the NeOn vision being funded by the Eu-

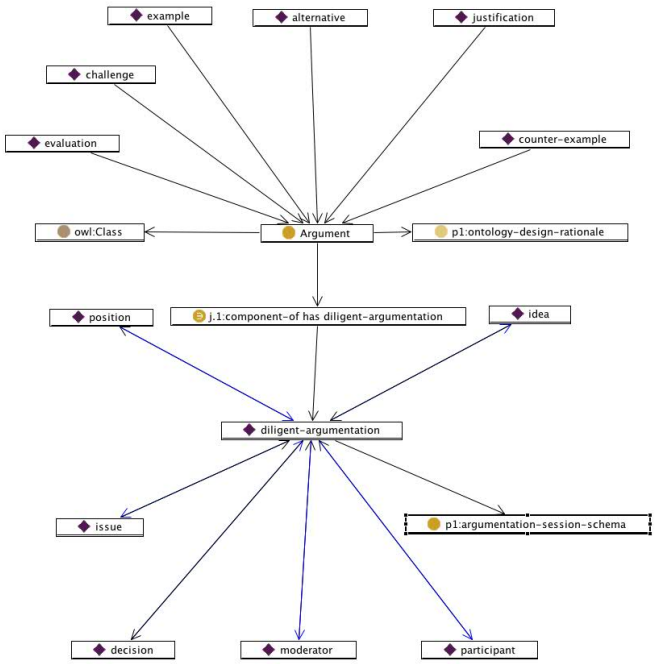

Figure 4: C-ODO-based model of DILIGENT argumentation (focus on ontology design rationale)

ropean Commission 6th IST Framework Programme. Further information on NeOn is available on http://www.neon-project.org.

8. REFERENCES

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

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

[3] Carola Catenacci, Aldo Gangemi, Jos Lehmann, Malvina Nissim, and Valentina Presutti. Design rationales for collaborative development of networked ontologies state of the art and the collaborative ontology design ontology. Deliverable d2.1.1 of the neon project, NeOn project, 2006.

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

[5] The collaborative ontology design ontology. http://www.loa-cnr.it/ontologies/OD/OntologyDesign.owl.

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

[7] A. Gangemi, S. Borgo, C. Catenacci, and J. Lehmann. Task taxonomies for knowledge content. Deliverable D07 of the Metokis Project, Laboratory for Applied Ontology ISTC CNR, 2005. `http://www.loa-cnr.it/Papers/D07_v21a.pdf`.

[8] 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.

[9] 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`.

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

[11] Aldo Gangemi. Ontology Design Patterns for Semantic Web Content. In *M. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference*, Galway, Ireland, 2005. Springer.

[12] P. Haase, S. Rudolph, Y. Wang, and S. Brockmans. Networked ontology model. Technical report, Universität Karlsruhe, 2006.

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

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

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

[16] W.C. Mann and S.A. Thompson. Rhetorical structure theory: Toward a functional theory of text organisation. *Text*, 8(3):243–281, 1988.

[17] 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, KMi, The Open University, UK, 2000.

[18] 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.

[19] D.E. Perry and G.E. Kaiser. Models of software development environments. *IEEE Transactions On Software Engineering*, 17(3):283–295, 1991.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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, Institut AIFB, Universitaet Karlsruhe (TH), Germany, 2006. `http://www.sekt-project.org`.

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

[27] 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 ESWC-2005 - European Semantic Web Conference*, Heraklion, Crete, 2005. Springer.

[28] 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.

[29] F.H. van Eemeren and R. Grootendorst. *A Systematic Theory of Argumentation: The Pragma-Dialectical Approach*. Cambridge University Press, Cambridge, 2003.

[30] D. Vrandecic. Explicit knowledge engineering patterns with macros. In C. Welty and A. Gangemi, editors, *Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005*, Galway, Ireland, 2005.

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