A Generic Ontology For Collaborative Ontology-Development Workflows

Abraham Sebastian, Natalya F. Noy, Tania Tudorache, Mark A. Musen

Stanford Center for Biomedical Informatics Research, Stanford University, Stanford, CA, US
{abeseb,noy,tudorache,musen}@stanford.edu

Abstract. As knowledge engineering moves to the Semantic Web, ontologies become dynamic products of collaborative development rather than artifacts produced in a closed environment of a single research group. However, the projects differ—sometimes significantly—in the way the community members can contribute, the different roles they play, the mechanisms they use to carry out discussions and to achieve consensus. We are currently developing a flexible mechanism to support a wide range of collaborative workflows in the Protégé environment. In this paper, we analyze workflows for several active projects, and describe the properties of these workflows. We discuss an ontology that we developed to represent different aspects of workflows for collaborative ontology development. This ontology is a key component of the customizable workflow support in Protégé. We evaluate the coverage and flexibility of this ontology by using it to represent formally two different collaborative workflows described in the literature, DILIGENT and BiomedGT. This evaluation demonstrates that our workflow ontology is sufficiently flexible to represent these very different workflows.

1 Overview of Workflows for Collaborative Ontology Development

Until recently, ontology development was mostly the purview of ontology engineers and they developed ontologies individually or in small groups, with informal practices supporting their collaboration. As ontologies become prevalent domain descriptions in many application domains, several trends change the practice of ontology development.

First, domain experts become increasingly involved in developing ontologies. This development is a natural consequence of ontologies covering domains in more detail—thus, including information that ontology engineers simply do not know. On most large projects today, ontology development is a **collaborative effort**, involving both ontology engineers and domain experts. The number of users participating in development ranges from a handful (e.g., the Foundational Model of Anatomy [8]), to a couple of dozens (e.g., the National Cancer Institute's Thesaurus [9]), to the whole community contributing to the ontology in some way (e.g., the Gene Ontology [4]).

Second, with larger groups of users contributing to ontology content, many organizations define formal **workflows** for collaborative development, describing how project participants reach consensus on definitions, who can make proposals for changes, who can perform changes, who can comment on them, when ontology changes become public and so on. Some collaborative projects have been publishing and refining their workflows for years (e.g., the Gene Ontology). In other projects, ontology researchers are working actively with domain experts to make these workflows explicit and to provide

tooling for supporting the specific workflows (e.g., ontology development for the UN Food and Agriculture Organization (FAO) in the NeOn project [7]).

Third, these workflows **differ** from project to project, sometimes significantly. A workflow for a specific project usually reflects that project's organizational structure, the size and the openness of the community of contributors, the required level of rigor in quality control, the complexity of the representation, and other factors.

We are currently working on providing comprehensive support for collaborative ontology development in the Protégé system.[1] Protégé enables users to edit ontology in a distributed manner, to discuss their changes, to create proposals for new changes and to monitor and analyze changes. Integrating support for collaborative workflows in such a system would mean having the tool itself "lead" the user through the workflow steps. For example, the tool can enable or disable certain options, depending on the user's role, indicate to the user the current stage of the workflow and the actions expected or required from this user at this stage, or enable a user to initiate new activities. Furthermore, because the workflows differ from project to project, such support must be customizable, in terms of both the execution steps and the user interface.

This paper makes the following contributions:

- We *analyze* collaborative workflows across a wide variety of active ontology development projects and identify features of these workflows (Section 2).
- We develop an *ontology* to describe workflows for collaborative ontology development (Section 4).
- We *evaluate* the coverage and flexibility of our workflow ontology by representing two workflows described in the literature as a set of instances in this ontology (Section 5). Our evaluation shows that the representation that we developed is flexible enough to account for a wide variety of published workflows.

2 Analysis of Workflows for Collaborative Development

We start our analysis of workflows that ontology developers currently use by describing several projects for which published workflow descriptions exist. We then discuss the features of these workflows and identify the core components of describing a workflow for collaborative ontology development.

2.1 Use Cases for Collaborative Ontology Development

Each of the use cases that we mention in this section are ongoing ontology-development projects. We chose a cross-section of the projects that highlights different strategies for collaboration and consensus building.

Biomedical domain provides a rich sample of large-scale collaborative ontology projects. The **Gene Ontology (GO)** [4] is perhaps the best known example of such project. The GO provides terminology for consistent description of gene products in different model-organism databases. Members of the GO community constantly suggest new terms for this ontology. Three full-time curators examine the suggestions and incorporate them into GO on a continual basis. Developers use a sourceforge issue tracker to track suggestions from the community.

[1] `http://protege.stanford.edu`

The **National Cancer Institute's Thesaurus (NCI Thesaurus)** [9] is a reference ontology that covers areas of cancer biology, translational science, and clinical oncology. NCI regularly publishes baseline versions of Thesaurus. After a new baseline is published, multiple editors get assignments to edit their respective parts of the NCI Thesaurus. They use a client–server version of Protégé to perform their changes. During each cycle, an editor usually performs from a dozen to several dozens of ontology edits. At the end of the cycle, a lead editor reviews all changes, and accepts or rejects them to create and publish a new baseline.

The **Biomedical Grid Terminology (BiomedGT)**[2] is a terminology product that enables the wider biomedical community to participate directly in extending and refining the terminology. BiomedGT defines several roles for community members, such as *Subject Matter Expert (SME)*, *Alpha Curator*, and so on. A precise workflow (Section 5.2) describes what tasks a role has to execute in each steps of the editing process.

Researchers in the NeOn project are working with the domain experts from the **Food and Agriculture Organization (FAO)** to formalize their process of editing ontologies for large applications such as the *Fisheries Stock Depletion Assessment system* [7]. The FAO workflow focuses on the states that ontology changes go through: *draft*, *to be approved*, *approved*, or *published*. Users with different roles have the authority to change the state. The FAO workflow distinguishes between the maintenance version of the ontology, which is available only to the editors, and the published version, which is available to the public.

The DILIGENT methodology for collaborative development [10], which has been used in several European projects, focuses on the formal argumentation process in the ontology development. The discussions are centered around *issues*, and users take decisions using a voting mechanism. We describe the workflow in more detail in Section 5.1.

2.2 Features of Workflows for Collaborative Development

The main common thread for the workflows that we described in the previous section is that these are *human-centered workflows*: many steps in these workflows require actions from humans. Human-centered workflows are different from, say, service workflows that combine software services for automatic execution. Furthermore, most activities in the workflows are described in terms of the steps that a proposed change must go through in order to be incorporated into the ontology or in terms of the states that an ontology component goes through before it is published.

Our discussion in the previous section also highlights many differences between collaborative workflows that are deployed today.

Each project that defines a workflow formally, has a different set of *roles* for users and a different set of *activities* that users with these roles can perform. Not only the names for the roles differ, but also the way in which specific privileges are distributed between different users. For example, a *moderator* in DILIGENT is responsible for structuring the discussion rather than making changes. Deleting elements from an ontology has a special status in the FAO workflow and only *validators* can perform this action. Users who make suggestions in GO usually propose new terms rather than suggest changes to existing terms. The GO curators both make changes and perform quality control (i.e., there is no additional level of approval).

[2] `http://biomedgt.org`

The structure of the workflows differs significantly. For example, in the ontology development of FAO, the workflow process centers of the notion of *states* for changes and new concepts. Each change goes through several states, such as *Draft* and *To be approved*. Alternatively, the DILIGENT workflow focuses on the ontology-level *issues* and the formal argumentation process to address the issues. The NCI Thesaurus workflow is task- and process-oriented, with a lead editor assigning tasks to regular editors, and editors being driven by their assigned tasks. The NCI Thesaurus workflow also has a fixed temporal element to it: the lead editor reviews the changes and prepares a new baseline every two weeks. This temporal dimension is different from workflows where the next step starts after the previous one finishes, regardless of how long that step takes.

The way users *propose and make changes* also differs. For instance, the SMEs in BiomedGT create proposals for changes to property values. Curators then choose form the alternatives. By contrast, editors of the NCI Thesaurus make changes directly to the ontology and the lead editor then verifies and accepts or rejects the changes.

The more centralized projects, such as the development of the NCI Thesaurus, have the notion of *task* assignments. Managers can assign users areas of ontology to work on or specific tasks. Users must have access to the list of their outstanding tasks, be able to change the status of the task (e.g., set it to *complete*); managers must be able to monitor progress of tasks that they assigned.

A flexible tool that supports these types of workflows must account for these differences and enable users to describe their workflow in the terms that are relevant to them. This flexibility and wide coverage are the main requirements in defining our workflow ontology that we describe in Section 4.

3 Related Work

Research that affects our work and that we will discuss in this section comes from both the ontology community and the domain of business-process modeling.

C-ODO is an OWL meta-model for describing collaborative ontology design [3, 2]. The model focuses on describing design rationale, design decisions, and argumentation process. C-ODO also represents workflows, more specifically, *epistemic workflows*. Epistemic workflows describe the flow of knowledge from one rational agent to another. The focus of an epistemic workflow is the knowledge resource itself and the workflow is focused on the description of how the knowledge resource changes. In our work, we focus on the *execution workflows* describing the actions taken by the agents, which is a complementary view to the one taken by C-ODO. Because our goal is the design of the tool, an execution-based workflow description is more practical.

Researchers have used formal ontology languages to enhance web services with semantic information. Although the languages for describing Semantic Web Services, such as OWL-S [6], WSMO [6] and WSDL-S [1], have similar control structures to the ones that we need in representing workflows, their focus is on automating the discovery, invocation, and composition of services, which is not in the scope of our work. These languages focus on the interactions between software services and not humans.

We have discussed several workflows in Section 2. Researchers have developed tools to support some of these specific workflows. For instance, the BiomedGT work-

flow is supported by the BiomedGT environment.[3] This environment extends Semantic MediaWiki to support the specific roles in BiomedGT (e.g., *SME*, *Alpha SME*, *Curator*, etc.), the process for creating proposals and for suggesting changes. Similarly, coefficientMakna [10]— also based on a wiki platform—is designed explicitly to support the DILIGENT argumentation-based approach to ontology engineering. We use the insights provided by the authors of these tools and the specific solutions that they adopted in our approach. However, our goal is different from the goals of the authors of these systems: We want to develop a framework that works for a vast variety of *different* workflows, rather than for a specific one. We propose to have a general framework that is easily customizable for a new workflow description.

The approach of describing workflows declaratively and generating tools from this description has become a mainstay in business-process modeling, and there are several industry standards for describing process workflows. For example, Business Process Execution Language (BPEL) [4] is an executable process-modeling language developed by OASIS. BPEL focuses on describing interactions between services, such as Web services, and not human interactions. BPEL4People[5] is an extension of BPEL that models the types of human interaction and the roles that users play in a workflow as first-class objects. Business Process Modeling Notation (BPMN)[6] is a graphical process-modeling language developed by OMG. These languages model the notion of tasks and activities, and of control structures for process flow. Naturally, the tasks and activities modeled in these languages are very generic. In our workflow ontology, we focus on the tasks and activities that are specific to ontology development. In the future, when integrating our tools with a workflow execution engine, we will map the top level of our ontology to the process-modeling language required by the workflow execution engine.

4 Generic Ontology for Representing Collaborative Workflows

The workflow ontology that we describe here provides a formal language for describing workflows for collaborative ontology development. Figure 1 shows the key classes in this OWL ontology. The `Workflow` class represents the workflow object. Each instance of this class describes a workflow (e.g., an approval workflow or a voting workflow). Each workflow is associated with a set of initialization parameters, a workflow target, a partially ordered set of activities or states. We describe these components in detail in the rest of this section. A workflow may also have an `initiator` (a user who initiates a workflow); it usually has a `name`; and may have other parameters.

Typically, a workflow description contains a set of input parameters—represented as an instance of the `InitiationForm` class (e.g., a deadline for a voting to end). An initiation form contains a list of parameters and their types (e.g., a `deadline` which is a `date`, an `assignee` which is anyone with a specific role, etc.).

Each workflow for collaborative ontology development is associated with ontology elements, ontology changes, or discussions about an ontology (the latter are themselves linked to ontology elements or changes)—a workflow `target`. To represent

[3] `http://biomedgt.org`
[4] `http://www.oasis-open.org/committees/wsbpel/`
[5] `http://bpel4people.sourceforge.net/`
[6] `http://www.bpmn.org/`

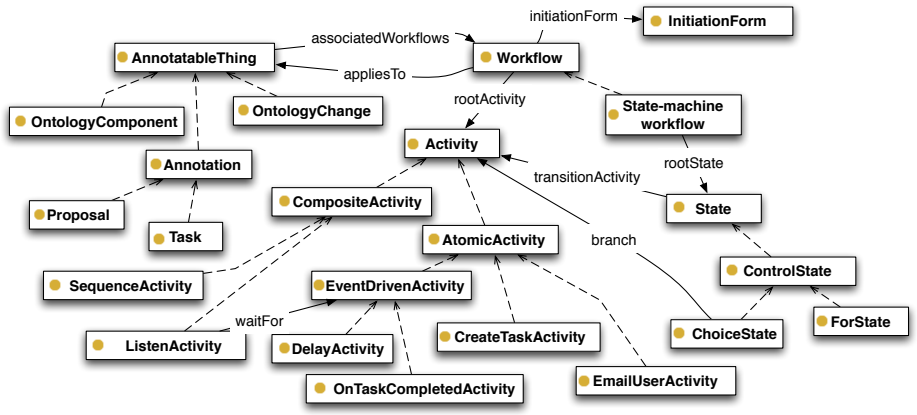

Fig. 1. A snapshot of the classes in the workflow ontology and relations between them. Rectangles represent classes. Dashed lines represent subclass-of relationship. Solid lines are `someValuesFrom` restrictions.

workflow targets, we reuse the Protégé Change and Annotations ontology (CHAO) that is used by several Protégé plugins that deal with change management. CHAO has a generic class `AnnotatableThing`, with subclasses representing `Ontology – Change`, `OntologyComponent`, and `Annotations`. The `Annotations` class contains representation of discussions, proposals, and comments. Thus, when we describe a specific workflow, we specify which of the subclasses of `Annotatable–Thing` serves as its `target` (e.g., the class `OntologyChange` for a workflow describing steps to accept a change).

We distinguish two types of workflows. First, there are workflows that are defined as a set of activities, with one activity invoking the next. Such a workflow consists of a partially ordered set of *activities* (instances of the `Activity` class). The second type are workflows defined as state machines, with states and transitions. Transitions from one state to another occur in response to events. An instance of the `State` class represents a state in such a workflow. A `State` contains one or more `EventDrivenActivities`, each of which listens for the occurrence of specific events. One possible action in response to an event is to execute a transition to another state. We provide an example instantiation of a state-machine–based workflow in Section 5.2.

Subclasses of the `Activity` class define specific activities in the collaboration process for both types of workflow. For example, a `CreateTask` activity invokes a task-management service and creates a task for a particular user or a set of users with a specific role. This task will contain a detailed description of the work item, a due date, a priority, and so on. A `VotingActivity` is an activity that sends notifications to users (via the task-management service) and then tallies the Yes/No votes from those users. The result of the tally determines the activity to which the process transitions next.

A set of `EventDrivenActivities` contains activities that wait for a particular event to occur, such as for some time to elapse (`DelayActivity`) or for a task to be completed (`OnTaskCompletedActivity`). A typical workflow-engine implementation would use a `ListenActivity` to suspend active execution of an instance of

a workflow and to wait for any of several EventDrivenActivities to occur. For example, a ListenActivity may refer to two EventDrivenActivities: one activity executes when some external event occurs (e.g., completion of a user-assigned task); the other activity executes when a timeout expires.

An activity can be a composite activity or an atomic activity. A CompositeActivity is an activity that has one or more sub-activities (i.e., substeps). For example, the SequenceActivity has an activities property that refers to an ordered list of activities that it executes in sequence. Composite activities include control-flow structures (states) such as the For, While and Choice states. These are analogous to the control structures in several workflow-description languages.

The participants in a workflow have assigned *roles*. Each role has a set of privileges associated with it. The privileges determine what operations a given user is allowed in the context of the workflow. For example, in a simple approval workflow for a change to an ontology any user who is an Editor may be allowed to insert or update an ontology element, but only a user who is a Manager may be authorized to approve such a change and to have it incorporated into the publicly available version of the ontology. Specific roles in a workflow are instances of the Role class. Each user then has a property role, which has one or more instances of the Role class as its values. When we describe workflow states and activities, we usually need to represent all users with a specific role (e.g., *"any editor can comment on this change"*). We use anonymous classes in OWL to define all users with the same role (cf. Figure 2).

5 Evaluation: Representing Published Workflows

We have represented a number of workflows in our ontology in order to evaluate its flexibility and coverage. In this paper we describe how to represent two quite different workflows as a set of instances in our ontology: the DILIGENT workflow focuses on a sequence of events to invoke and carry out the argumentation process among participants; and the BiomedGT workflow is a state-based workflow. Due to lack of space, we simplified some of the steps in the Figures 2 and 3.

5.1 DILIGENT

Tempich and colleagues [10] define several steps in the DILIGENT methodology for collaborative development, focusing on the argumentation process. First, the group chooses a moderator to structure the discussion and organize the decision. Second, they agree on a decision procedure (typically, a voting mechanism) to reach consensus on the issues under consideration. In the third step, the group identifies issues for discussion, and then discuss the issues and propose ideas. Finally, they analyze the results of the argumentation process and agree or disagree on an issue or postpone its resolution.

We model this decision procedure as a sequential workflow (Figure 2). A moderator initiates this workflow. The workflow applies to an Annotation (an issue is an instance of the Annotation class). We configure the VotingActivity to collect and tally votes from all users who have the role Editor. If the result of the voting round was in favor of the proposal, an appropriate notification is sent using the EmailUser activity, and the proposal is queued for implementation. If the activity timed out, the Moderator is notified—she may choose to reject the proposal or to start another round of argumentation in this event.

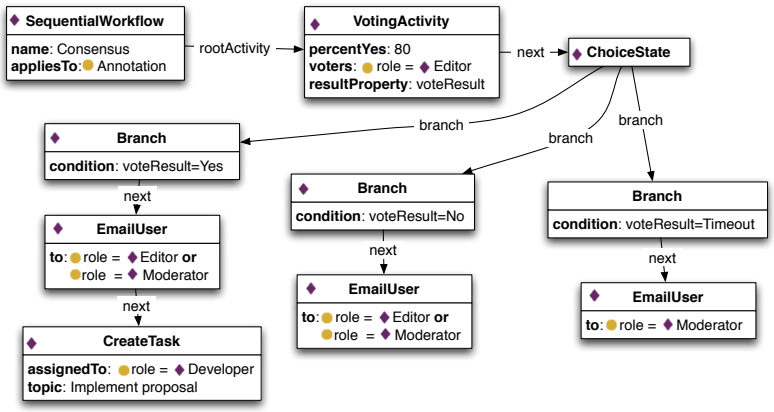

Fig. 2. Decision procedure in the DILIGENT **argumentation process.** Rectangles represent instances of classes in Figure 1. Some property values are classes, indicated by the round icon.

5.2 BiomedGT

The Biomedical Grid Terminology (BiomedGT) employs a well-defined process for incorporating change requests into their ontology. A *subject matter expert* (SME) initiates the process by making a recommendation in the form of a *proposal*. Once SMEs specify the proposal fully, an *alpha SME* reviews the proposal. Based on her review, the alpha SME either marks the proposal as *ready for implementation* or rejects the proposal. If she rejected the proposal, the proposal is either queued for further discussion or removed. A proposal that is marked as ready for implementation is sent to an *alpha curator*, who reviews the proposal and appropriately delegates work to his team of *content curators*. The content curators implement the changes according to the specification. The originators of the proposal including the alpha SME are then notified so that they may review the implementation.

Figure 3 shows the representation of the *design review* phase of the BiomedGT process. The workflow starts with an `EventDrivenActivity` listening for changes to the proposal (`OnProposalChangeActivity`). Once the proposal is completely specified, the SME changes the status of the proposal to `to be approved`. This event triggers a state transition to the *Approval* state. In this state, a notification is sent to an `AlphaSME` who may then approve or reject the proposal after review. The `Approval` state contains an `EventDrivenActivity` that waits for the `AlphaSME` to complete her review and makes a state transition based on the result of her review. If the proposal is rejected, the workflow moves to the `Specification` state where the `SMEs` may make corrections or other changes. If the proposal is approved, the workflow transitions to the *Ready for Implementation* state—the `Alpha Curator` is notified (via a task management system) that he must implement the proposed changes.

6 Discussion and Future Work

Our evaluation demonstrated that our ontology is flexible enough to describe different workflows. Using the generic workflow representation described in Section 4, we represented DILIGENT as a sequential workflow and BiomedGT as a state-based workflows.

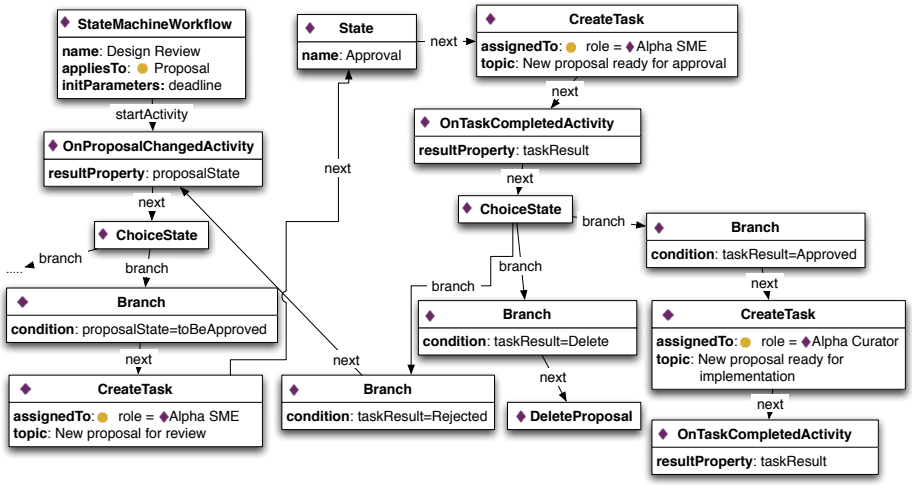

Fig. 3. Workflows for authoring and updating terminologies in BiomedGT. Rectangles represent instances of classes in Figure 1. Some property values are classes (shown as round icons).

Our main purpose in developing the workflow ontology is to create executable workflow specifications that can be run by workflow engines as part of a larger collaborative framework. There are several benefits in using an ontology to represent workflows. First, we are able to describe workflows with activities that are specific to ontology development, and that are applied to specific ontology components. This representation enables fine-grained control of the flow of operations at different stages of the development process. Second, we can represent workflows at different levels of abstraction, which enables the reuse of workflow descriptions. For example, we have defined a `VotingActivity` class that is common in many collaborative scenarios. Then, we have created specialized types of voting (`YesNoVotingActivity` and `FiveStarVotingActivity`) that inherit all properties and restrictions of the `VotingActivity`. The more specialized activities, however, can have different execution objects and user-interface components attached to them. Third, we can use the ontology to generate, configure, and control the user interface of the collaborative environment. Because we are describing all workflow activities with their associated user-interface components, we will be able to adapt easily the collaborative environment to changes in the workflow: For most cases, it will be enough to change the description of a workflow in the ontology, and to regenerate the collaborative environment.

When modeling the Workflow ontology, we were faced with the common trade-off between usability and reusability. We opted for reusability here, assuming that not all our users will have the resources to implement a custom workflow engine. Therefore, our ontology provides the generic building blocks for describing collaborative workflows, without prescribing any specific activity or operation. As a consequence, our framework can be adapted to virtually any type of workflow, with the trade-off that the initial work will go into describing a custom workflow using our ontology. To address this issue, we have already defined classes of activities that we consider to be common

in collaborative workflows (such as voting, proposals, creating tasks, etc.) that can be used "off-the-shelf" in the description of custom workflows.

There are some modeling challenges that we still must address. For example, we must represent and interpret the transition conditions between workflow states. The transition conditions may contain arbitrary mathematical expressions. We are considering using a scripting language, such as Jython, or JRuby. Another possibility is to use SWRL for this purpose.

As future evaluation we plan to determine how easy it is for project managers to describe their workflow as a set of instances in the Workflow ontology. For the current evaluation, we (the ontology authors) have described the workflows presented in Section 5. Our goal was not evaluating the ease of use, but the workflow ontology itself.

Adding more elaborate support to modeling argumentation and negotiations is a natural extension of our model. Researchers have developed formal argumentation frameworks [5] and represented some aspects of these frameworks in an ontology [10]. We have demonstrated a simple way to represent an argumentation protocol in our framework (Section 5.1) and we envision reusing the DILIGENT ontology to support a more elaborate argumentation process.

Acknowledgments

This work was supported in part by a contract from the U.S. National Cancer Institute. Protégé is a national resource supported by grant LM007885 from the U.S. National Library of Medicine.

References

1. R. Akkiraju. Web service semantics - WSDL-S. W3C Member Submission, World Wide Web Consortium (W3C), November 2005. http://www.w3.org/Submission/WSDL-S/.
2. C. Catenacci. Design rationales for collaborative development of networked ontologies state of the art and the collaborative ontology design ontology. NeOn Project Deliv. D2.1.1, 2006.
3. A. Gangemi, J. Lehmann, V. Presutti, M. Nissim, and C. Catenacci. C-ODO: an owl metamodel for collaborative ontology design. In *Workshop on Social and Collaborative Construction of Structured Knowledge at WWW2007*, Banff, Canada, 2007.
4. GO Consortium. Creating the Gene Ontology resource: design and implementation. *Genome Res*, 11(8):1425–33, 2001.
5. W. Kunz and H. W. J. Rittel. *Issues as elements of information systems*. Institute of Urban and Regional Development, University of California, Berkeley, 1970.
6. D. Martin. OWL-S: Semantic markup for web services. W3C Member Submission, World Wide Web Consortium (W3C, November 2004. http://www.w3.org/Submission/OWL-S/.
7. O. Muñoz García, A. Gómez-Pérez, M. Iglesias-Sucasas, and S. Kim. A Workflow for the Networked Ontologies Lifecycle: A Case Study in FAO of the UN. In *Conf. of the Spanish Assoc. for Artificial Intelligence (CAEPIA 2007)*, volume LNAI 4788. Springer, 2007.
8. C. Rosse and J. L. V. Mejino. A reference ontology for bioinformatics: The foundational model of anatomy. *Journal of Biomedical Informatics.*, 2004.
9. N. Sioutos, S. de Coronado, M. Haber, F. Hartel, W. Shaiu, and L. Wright. NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, 40(1):30–43, 2007.
10. C. Tempich, E. Simperl, M. Luczak, R. Studer, and H. S. Pinto. Argumentation-based ontology engineering. *IEEE Intelligent Systems*, 22(6):52–59, 2007.