Supporting Collaborative Ontology Development in Protégé

Tania Tudorache, Natalya F. Noy, Samson Tu, Mark A. Musen

Stanford University, Stanford, CA 94305, US
{tudorache, noy, tu, musen}@stanford.edu

Abstract. Ontologies are becoming so large in their coverage that no single person or a small group of people can develop them effectively and ontology development becomes a community-based enterprise. In this paper, we discuss requirements for supporting collaborative ontology development and present Collaborative Protégé—a tool that supports many of these requirements, such as discussions integrated with ontology-editing process, chats, and annotations of changes and ontology components. We have evaluated Collaborative Protégé in the context of ontology development in an ongoing large-scale biomedical project that actively uses ontologies at the VA Palo Alto Healthcare System. Users have found the new tool effective as an environment for carrying out discussions and for recording references for the information sources and design rationale.

1 Ontology Development Becomes Collaborative

Recent developments are dramatically changing the way that scientists are building ontologies. First, as ontologies are becoming commonplace within many scientific domains, such as biomedicine, they are being developed collaboratively by increasingly large groups of scientists. Second, ontologies are becoming so large in their coverage (e.g., NCI Thesaurus with 80K concepts) that no one user or small group of people can develop them effectively. Hence, organizations such as the NCI Center for Bioinformatics "outsource" some of their ontology development to the scientific community at large. Third, in the last one or two years, many users have become quite familiar and comfortable with the concept of user-contributed content, both in their personal and professional lives (cf. Web 2.0). Thus, domain experts need tools that would support collaborative ontology development and would include collaboration as an integral part of the ontology development itself.

Researchers are only now beginning to develop such tools. Last year, tool developers were invited to contribute their tools for collaborative construction of structured knowledge (which included not only ontologies, but also any structured data) to the CKC Challenge, which brought together developers and users in order to examine the state-of-the-art and to understand the requirements for new tools [10]. In general, the participants in the CKC Challenge agreed on several key points. First, the notion of collaborative development of ontologies and most of the tool support was in its infancy. Second, the spectrum of tools even in the relatively small set of the challenge participants (from tools to organize tags in a hierarchy to full-fledged ontology editors) demonstrated that no single tool is likely to fill the niche completely. Third, the

requirements for such tools to support collaborative development in any specific setting were still poorly understood. The challenge participants started identifying these requirements. Starting with the initial set of requirements identified as the result of the CKC workshop, we continued the requirements-gathering phase in the context of extending the Protégé ontology editor to support collaborative ontology development. To gather specific requirements, we conducted interviews with representatives of several groups that currently use Protégé for ontology development and that were trying to adopt a more formal process for development. These projects included the development of the NCI Thesaurus [17], the ontologies for the ATHENA-DSS project at the VA Palo Alto Healthcare System [7], the Ontology of Biomedical Investigations (OBI) [2], the RadLex ontology for annotating radiological images [14], and many others. As the result of this process, we collected a set of requirements for an ontology editor supporting collaboration. Note that we focused on the projects that need a full-fledged ontology editor and where ontologies are fairly rich in structure and large in size. For example, the NCI Thesaurus is an OWL DL ontology with more than 80K classes, several thousand of which are defined classes. Both RadLex and ATHENA-DSS ontologies are frame-based ontologies that use different types of constraints on properties extensively. We then developed Collaborative Protégé by extending the Protégé tool with a set of features to support these requirements. We have performed the formative evaluation of Collaborative Protégé in several different projects in order to evaluate the usability of the tool and to understand what users like and do not like about it, how they use it, and what other features they need to support their work.

More specifically, this paper makes the following contributions:

- We identify a set of requirements for developing expressive ontologies and knowledge bases collaboratively (Section 2).
- We present Collaborative Protégé—an ontology editor that supports collaboration through integration of features such as discussions, chats, and annotations in the ontology editor (Sections 4, 5, and 6).
- We perform the formative evaluation of Collaborative Protégé in the context of representing formally clinical practice guidelines in the ATHENA-DSS project (Sections 7, 8, and 9).

2 Requirements for Support of Collaborative Ontology Development

We have identified our requirements for tool support for collaborative ontology development through interviews with many institutional Protégé users. The requirements that we identified significantly extend the set of requirements from the CKC workshop, and focus on the requirements of ontology developers for domains such as biomedicine. These developers are usually domain experts rather than knowledge engineers.

In most of these projects, users have already used Protégé in a client–server mode that enabled distributed users to edit the ontology simultaneously, immediately seeing the changes that others make. Thus, we focused on the features that would explicitly support collaboration. Furthermore, by the nature of projects already having chosen

Protégé for their ontology development, most of them had to work with expressive ontologies and knowledge bases. In some cases, users worked collaboratively to extend the ontologies themselves (e.g., the NCI Thesaurus or OBI), and in others they additionally used an expressive ontology to create a knowledge base of classes and instances (e.g., ATHENA-DSS). The overarching theme of these interviews was the disconnect between the produced ontology on the one hand and all the thought and discussion that went into producing this artifact on the other hand. The former was captured in Protégé, but the latter was captured in myriads of email messages, forum posts, phone conversations, and MS Access databases. When someone browsed the ontology, it was often impossible to understand the rationale that went into the design decisions, to find which references were relevant, to find the external resources that informed the modeling decisions. Conversely, when developers read a mailing list post discussing a modeling issue, they do not see the context for that post.

The specific requirements for supporting collaborative ontology development that our users identified included the following:

Integration of discussions and annotations in ontology development Almost by definition, an ontology is an artifact that requires its authors to reach consensus. At the same time, our experience demonstrates that developing an ontology is not a straightforward task and the developers can disagree on the best way to model concepts in the ontology or, in fact, on which concepts to model. Thus, tools that support discussion, such as forums and chats, are essential. However, these discussions happen in email messages and similar venues and are completely separated from the resulting artifact. For example, one of our collaborators (the OBI developers) reported recently that they found themselves in a heated discussion on the definition of a specific term (namely, analyte), something they thought they have resolved several months before. However, that discussion was not captured in the class definition and was not available when the question arose again. In fact, linking interactions among users and their comments and annotations directly to the artifacts they are producing, carries several advantages. First, when browsing the ontology later, developers and users can understand better why certain design decisions were made, what alternatives were considered, which group advocated a certain position, and so on. Second, when carrying out the discussion itself, if it is integrated in the development tool, the participants can immediately see the context for the components being discussed, they can examine the corresponding definitions and relations. Thus, the requirement for integrating discussion tools into ontology development environment is two-fold: Make the discussions accessible from the ontology components that are being discussed and make the ontology components accessible when one examines or writes a discussion message.

Support for various levels of expressiveness The projects that use Protégé for collaborative development have rather expressive ontologies. For instance, one often comes across defined classes, complex restrictions, with intersections, in class definitions for the NCI Thesaurus. Thus, in the settings of these biomedical projects that heavily rely on ontologies, the collaborative version of the tools must ultimately have the same expressive power as a stand-alone ontology editor. It must support editing both ontology classes and instances.

User management and provenance of information With multiple authors contributing to the ontology and the corresponding discussion, it is critical for users to understand where information is coming from. Thus, users must be able to see who makes specific changes and when, who creates a new proposal for change, who votes on it, and so on. This information must also be searchable. One must be able to find all changes or comments made by a specific user, or all recent changes and comments.

Scalability, reliability, and robustness The traditional requirements of using tools in production systems include scalability (both in the size of ontologies and in the number of users), reliability (domain experts cannot afford to loose their data), and robustness (ontology-development tools should be no less robust than other tools that domain experts use). While several prototypes of collaborative tools have appeared recently, our experience shows that domain experts are usually reluctant to try a new tool until they are convinced the tools is ready to be used in production environment. Ontology development is not their primary task and they need tools that would help them perform this task quickly and reliably.

Access control We often hear from our users who develop ontologies collaboratively that one of the features that all ontology-development tools largely lack today is access control. Today, for the most part, any user with writing privileges can edit anything in an ontology. However, users need to have more fine-grained control, particularly in the development of large ontologies. For example, users with expertise in an area represented by some part of an ontology should be able to edit that part, but may be able only to browse other parts or link to them. In fact, many ontology-development projects today maintain separation between what different users can do: For instance, some users can make proposals for changes but not make the changes themselves; others can comment on these proposal, but not create new ones; another group of users can affect the changes in the ontology based on the discussion; yet others can perform quality control by reviewing and approving the changes. We need to extend access-control policies with a more detailed model of user roles and privileges [4]. Because in ontologies concept definitions are often intertwined and a change in one part can affect definitions in another part, making such separation is far from trivial.

Workflow support Many collaborative development projects have specific workflows associated with making changes. For example, there is a formal workflow for development of ontologies for the Food and Agriculture Organization (FAO) of the United Nations in the NeOn project [9]. The DILIGENT methodology for collaborative development [19], which focuses on formalizing the argumentation process, has been used in several European projects. A workflow specification may include different tasks that editors are charged with; the process for proposing a change and reaching consensus; roles that different users play, and so on. We are only beginning to understand different workflow models that collaborative ontology development requires [5]. Flexible support for these workflows must be an integral part of tools for collaborative development.

Synchronous and asynchronous access to shared ontologies Depending on the size of the group and the complexity of the ontology, users might prefer synchronous or asynchronous editing [16]. In some of the projects we studied, users wanted to have their changes seen by everyone as soon as they make them, without the additional step of

"checking in" their changes. In other cases, users preferred to have their own "sandbox" to test out the changes they are proposing before sharing them with everyone.

The core Protégé system supports some of the requirements listed here. Specifically, Protégé provides support for various levels of expressiveness, user management and provenance information, access control, and synchronous access to ontologies. It also addresses the requirement for scalability, reliability, and robustness. We describe workflow support elsewhere [15]. In this paper, we focus on the support for integration of discussion and annotations with ontology-development environment.

3 Related Work

A number of ontology editors support some aspects of collaborative development. For instance, OntoWiki [1] is a web-based ontology and instance editor that provides such capabilities as history of changes and ratings of ontology components. OntoWiki provides different views on instance data (e.g., a map view for geographical data or a calendar view for data containing dates). OntoWiki focuses on instance acquisition and provides only rudimentary capabilities for ontology editing. The Hozo ontology editor [18] enables asynchronous development of ontologies that are subdivided into multiple inter-connected modules. A developer checks out and locks a specific module, edits it locally, and then checks it back in. If the ontology is not modularized, however, a developer must lock the whole ontology preventing others from editing it while he makes his change—an approach that may not be practical in many circumstances.

Several wiki-based environments support editing ontologies and instance data. The adaptation of the wiki environments that are particularly suited for ontology editing usually support a specific editing workflow. For example, a LexWiki platform developed at the Mayo Clinic, which is based on Semantic MediaWiki, currently is at the core of community-based development of BiomedGT.[1] BiomedGT is a terminology from the NCI Center for Bioinformatics (the same group that develops the NCI Thesaurus). The goal of BiomedGT is to enable the wider biomedical research community to participate directly and collaboratively in extending and refining the terminology. LexWiki enables users to browse an ontology, to make comments or to propose changes to (usually text-based) definitions. The BiomedGT curators with the privileges to make changes then open this annotated ontology in Protégé and perform the actual edits there. Wikis provide a natural forum for discussions, and the provenance information for suggested changes is easy to archive. Wikis, however, are not intended for ontology development and users cannot easily edit class definitions using this kind of framework. For example, in BiomedGT, curators must switch to Protégé to make the actual changes.

The coefficientMakna and Cicero tools (also based on wikis) implement the DILIGENT methodology for collaborative development [19, 3]. The DILIGENT workflow focuses on the process of argumentation. The users discuss *issues*, which are usually specified at the ontology level (e.g., how should a particular classification be structured). The users present their arguments, suggest alternatives, agree and disagree with one another, and vote on the resolution. The editing environment explicitly supports these steps.

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

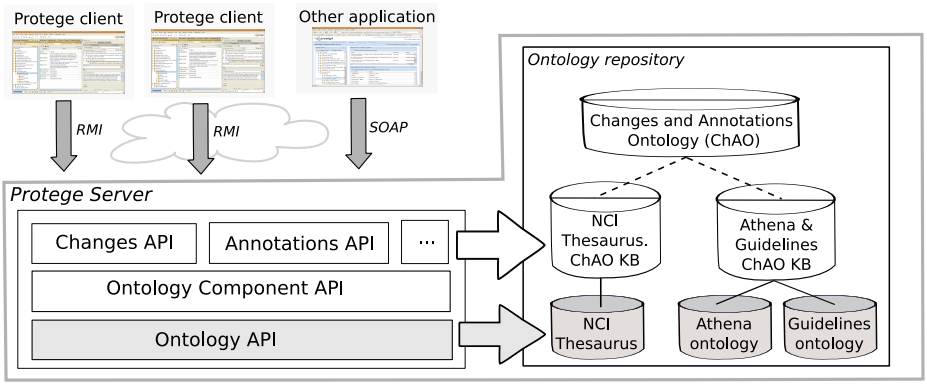

Fig. 1. The client–server architecture of Collaborative Protégé. The users work in Protégé clients or in other Protégé-based applications. All the changes made by a user in a client are sent to the server, and are immediately propagated to all other clients. The server has an ontology repository and several APIs to support the collaborative functionalities. Each domain ontology in the server repository has a *Changes and Annotations knowledge base* (ChAO KB) associated with it. This knowledge base contains instances of the ChAO ontology that describe the changes and annotations for the specific domain ontology.

Tools such as BiomedGT, Cicero, and coefficientMakna are designed to support specific workflows and could potentially work very well in the projects that use that specific workflow. The wiki-based tools have a simple interface that is best suited for making simple changes to the ontology. Wikis provide a natural forum for discussions, and the provenance information for suggested changes is easy to archive. However, these tools inherently cannot address the requirement of supporting ontology editing that conforms to a different workflow than the one for which they were designed. In the development of Collaborative Protégé, one of our goals is to make as few assumptions as possible about the editorial workflow that users will have and to develop mechanisms to make the tools customizable for different workflows.[2] Furthermore, these implementations do not provide structured access-control mechanisms.

4 Architecture of Collaborative Protégé

Our laboratory has developed Protégé—a widely used open-source ontology and knowledge base editor [13, 6]. At the time of this writing, Protégé has more than 100,000 registered users. Users can build ontologies in Protégé using different representation formalism ranging from *Frames*, to *RDF(S)* and *OWL*, and store them in file or database backends. Protégé is both robust and scalable and is being used in production environment by many government and industrial groups. The *ontology and knowledge base API* and *the plugin architecture* – one of the most successful features of Protégé, allow other developers to implement their own custom extensions that can be used either in the Protégé user interface or as part of other applications.

[2] We are currently working on adding customizable workflow support for Collaborative Protégé, but this work is outside of the scope of this paper.

Protégé can be run as a standalone application, or in a client–server setting. In the client–server mode, ontologies are stored on a central Protégé server. Users access the ontologies on the server to browse and edit them through desktop or web Protégé clients. The client–server mode uses the Remote Method Invocation (RMI) mechanism of Java.

We have developed Collaborative Protégé as an extension to the client–server Protégé. Collaborative Protégé enables users who develop an ontology collaboratively to hold discussions, chat, annotate ontology components and changes—all as an integral part of the ontology-development process. The key feature of Collaborative Protégé is the ability to create annotations. In this context, **annotations** are typed comments (e.g. example, proposal, question, etc.) attached to ontology components, or to the descriptions of ontology changes, or to other annotations. We define the structure of the annotations in the *Changes and Annotations ontology* (ChAO), which we describe in Section 5.

Figure 1 gives an overview of the main components of Collaborative Protégé. The Protégé server has an **ontology repository** that contains all the ontologies that Protégé clients can edit in the collaborative mode. The repository has ChAO knowledge bases (instances of the ChAO classes) for each of the domain ontologies in the repository. These instances represent the changes and the annotations for the corresponding ontology. Several related domain ontologies can share the same ChAO knowledge base. For example, in Figure 1, the *ATHENA-DSS* and the *Guideline* ontologies share the same ChAO knowledge base, while the NCI Thesaurus has its own ChAO knowledge base.

When a user edits the domain ontology in the Protégé client, each change that the user performs, is sent to the server. The server then performs several actions: (a) updates the central (server-side) ontology; (b) pushes the change to the other clients so that other Protégé users can see them immediately; and (c) creates one or several ChAO instances that represent the change [11]. The server also pushes the changes in the ChAO knowledge bases to the Protégé clients. When users create an annotation in the Protégé client, the Protégé server adds the corresponding instances to the ChAO knowledge base.

The server also provides several layered Java APIs for accessing the collaborative features. The **Changes API** provides methods for getting the structured log of ontology changes, to get detailed information about a change (like author and date of the change), and transactions – changes that are composed of several atomic changes, which are executed together as one single change. The **Annotations API** provides methods for adding annotations to ontology components and changes, for accessing the meta-data of an annotation (e.g. provenance information), to get the discussion threads, and so on. The **Ontology Components API** has common methods for both the Changes and the Annotations API and supports the access to the ontology components (e.g. classes, properties, individuals) stored as instances in the ChAO knowledge bases. The **Ontology API** has methods for accessing and changing the content of the ontologies and knowledge bases. It also provides support for transactions, caching, for multiple backends and support for the client-server architecture. The layered APIs can be used by other applications to access all domain ontologies as well as the collaborative information from the ChAO knowledge bases stored on the server side.

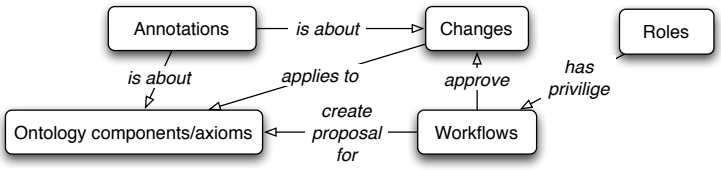

Fig. 2. Representation modules for collaborative ontology development. The *Ontology components* module represents the ontology elements. The *Changes* module captures declarative representations of changes to these elements. The *Annotations* module represents different types of annotations users can make about ontology elements and changes. The *Workflows* module represents activities and tasks in collaborative ontology development. The arrows in the diagram are labeled with sample relationships that may exist between classes in one ontology and another.

5 Ontologies for Supporting the Collaborative Development

Collaborative Protégé uses a set of ontology modules to drive the collaborative development process (Figure 2) .

The Roles module describes the users, roles, operations and policies that apply to a certain ontology. The Protégé server uses the *Roles* module for checking the users credentials at login time, and for determining whether a user is allowed to perform a certain operation based on the policies attached to an ontology instance. A user is represented as an instance of the `User` class and can play several roles (instances of `Group` class). For example, a user Ricardo can play the role of software developer and of editor. New roles can be easily added by creating new instances of `Role`, if a certain project requires them. To each ontology instance we associate a set of policies that define what operations are allowed for a role. For example, the NCI Thesaurus would be a represented as an instance of the `Project` class and would have associated to it a set of policy instances. One of the policies would allow editors to change the ontology. Because Ricardo is an editor, he will be allowed to write to the ontology, while for non-editor users the write access will be denied.

The Workflows module provides a formal language for describing workflows for collaborative ontology development. 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. For example, a workflow for a change proposal can be attached to a particular class in an ontology and would guide the flow of operations in the collaborative platform (e.g. first, start a proposal, then users votes, then count votes, then take a decision, etc.). We envision that future versions of Collaborative Protégé will provide flexible workflow support that would allow us just by changing a workflow description in the *Workflow* module to regenerate the collaborative platform to use the new workflow description.

The Ontology Components module provides a meta-language for describing representational entities in different ontology languages. For example, it contains classes such as `Class`, `Property`, and `Instance`. An instance of a `Class` represents a reified

object for a real class in an ontology (e.g. in the ATHENA-DSS ontology, we would have an instance of `Class`, called `Guideline`). *The Ontology Components* module provides classes for representing entities in OWL, RDF(S) and Frames. Collaborative Protégé uses this ontology, when users add comments to ontology components and also for change tracking. For example, if the user adds a comment to the `Guideline` class, the annotation instance will be attached to the corresponding `Class` instance (`Guideline`) in *the Ontology Components* module. This instance also references all the changes made to that class, and all other comments and annotations that users have attached to the class. For future versions, we are considering integrating the *Ontology Metadata Vocabulary* (OMV) [12] for the representation of OWL language constructs.

The Annotations module represents the different types of annotations that users make. The annotation types are extensions of the Annotea [8] annotations and contain concepts such as Comment, Question, Advice, Example, and so on. Each comment or annotation is linked to one or several ontology elements, or changes, which are represented in the ontologies describing `Ontology components` and `Ontology changes` [11]. If users need a new annotation type, they can simply extend this ontology by creating a new subclass of the `Annotation` class. In fact, users in our evaluation (Section 7) found this feature critical.

The Changes module contains classes representing different types of changes that can occur in an ontology. For example, an instance of the class `Class_Created` will represent a class creation event that references the `Class` instance from *the Ontology Components* module corresponding to the new class in the domain ontology. One of the challenges that we are facing is that each ontology language has its own types of changes. For example, in a Frames ontology, changing the domain of a slot will be recorded as a domain change event, while in OWL, the real change would actually be a remove and add domain axiom for a certain property. We plan to address this issue by defining a common layer for changes such as creating a class or adding a subclass and then creating subontologies for changes that are unique to each of the languages.

These service ontologies reference the components in the domain ontology. However, note that the domain ontology does not have references to the annotations, changes, and so on. Thus, the developers have the choice of whether or not to make their annotations public when they publish the ontology itself.

6 User Interface

The user interface of Collaborative Protégé (Figure 3) is implemented as a graphical extension of Protégé. Panel A in Figure 3 shows the class tree, Panel B shows the selected class information (in this case `Gene_Product`)—just like in the original Protégé user interface, while panel C displays the *collaborative tabs*. Each of the collaborative tabs supports one of the several collaboration features. For example, in the *Annotations* tab, the user can add comments to ontology components; in the *Changes* tab, the user may see the change history of the selected class and comment on a change; in the *Search* tab, the user can search all annotations on different criteria; in the *Chat* tab, the user may discuss with other online users, and so on.

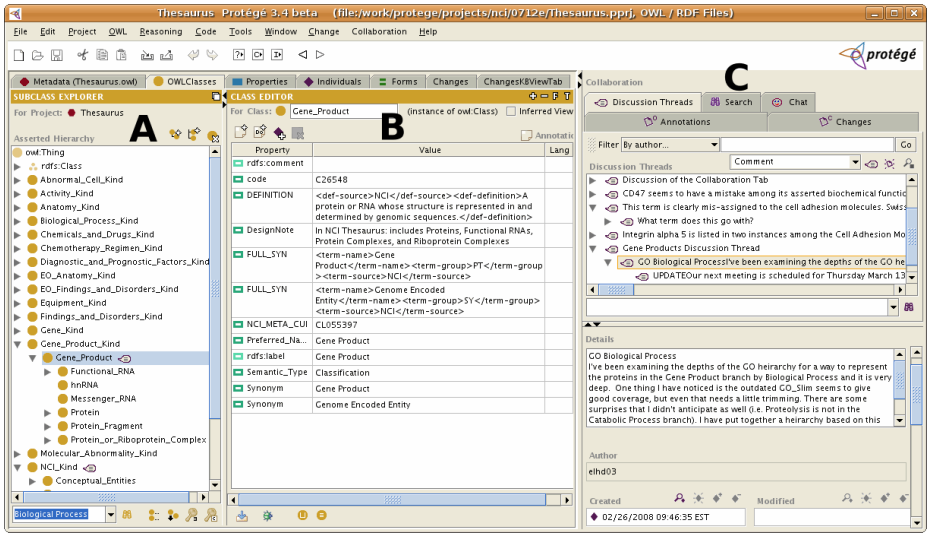

Fig. 3. The Protégé user interface, with the Collaborative Protégé plug-in. This screen capture shows the OWL Classes tab, in which the user edits and browses the classes that describe a domain ontology – here the NCI Thesaurus. Panel A shows the class tree; panel B displays the form for entering and viewing the description of the selected class `Gene_Product`, as a collection of attributes; and panel C shows the discussion among users about this class.

The **Annotations tab** is the default tab that users see when logging into Collaborative Protégé. The Annotations tab shows the annotations that are attached to the selected class in the tree (it also works for properties and individuals). The small callout icon shown in the class tree (Figure 3) next to the class name, indicates that the class has annotations. The lower part of the Annotations Tab shows the details of the selected annotation (e.g. the author, creation date, annotated entity, etc.). The annotations shown in the user interface are instances of the `Annotation` class. The user can create annotations of specific type (for example, `Comment`, `Question`, `Example`, `Proposal`, etc.). These types are defined in the *Annotations ontology* as subclasses of the `Annotation` class. Users can also reply to existing comments or notes—creating discussion threads related to a specific entity (Figure 3). The user may filter the displayed annotations by using one of the filtering criteria available at the top of the Annotations Tab. For example, she can filter by author, date, type and body of annotation.

Because the user interface takes the annotation types from the *Annotations ontology*—they are subclasses of the `Annotation` class—users can create their own types of annotation. To create a new annotation type, the user can edit the *Annotations ontology* itself, add the new type as the subclass of the `Annotation` class, define any additional properties that this custom-tailored annotation type should have, and the new annotation type will be available for use in Collaborative Protégé. In fact, in our evaluation (Section 7) users have defined their custom annotation type.

The **Discussion Thread tab** has a similar user interface and features as the Annotation tab (Figure 4). However, the annotations from the Discussion Thread tab are not attached to a particular ontology component, as the other annotations, but refer to the

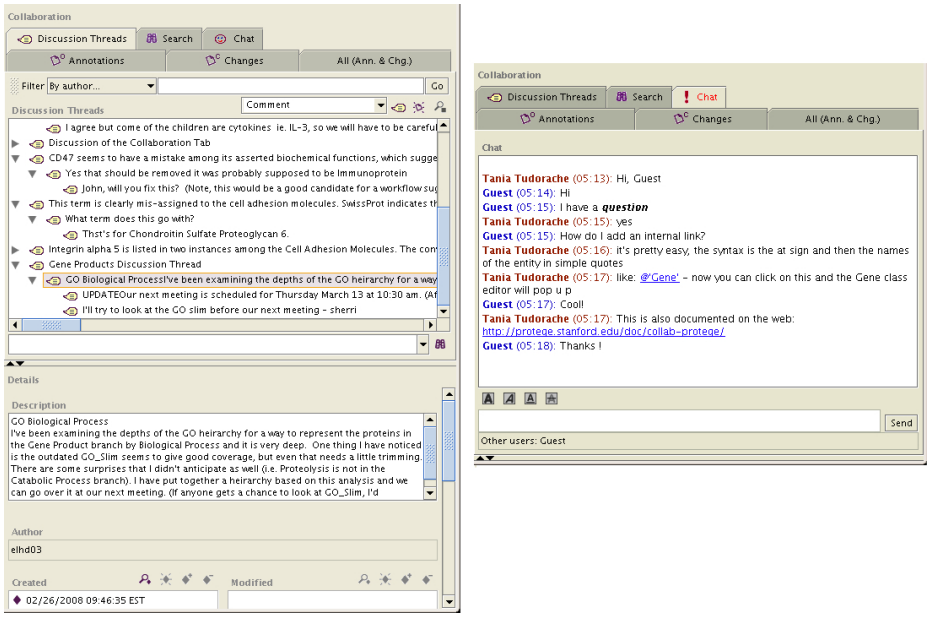

Fig. 4. Two of the collaborative tabs. The left screenshot shows the *Discussions Thread tab* where users can add comments on the ontology. The right screenshot is *the Chat Tab*, which allows users to chat and exchange internal and external links.

ontology itself. For example, users may discuss modeling patterns, or naming conventions that are broader in scope and that should apply to the whole ontology, rather than to individual ontology components.

The **Changes Tab** shows a chronological list of all the changes for the selected ontology component. For each change, the tab shows the change details (e.g. author, date, sub-changes, etc.). Users may also comment and have discussion threads related to a certain change as also shown in our example.

Users may also search all annotations based on different criteria in the **Search Tab**. For example, a user can search for all annotations of type `Comment` that have been made by an author `eldh` between `05/14/2007` and `05/14/2008`. The search result will show all the annotations that satisfy the criteria and will provide direct access to the annotated ontology elements or changes.

One of the popular features of Collaborative Protégé is the **Chat Tab** (Figure 4). Users connected to the Protégé server can exchange live messages. The chat panel supports HTML formatting of the message, such as bold, italics, highlight. One feature that sets the Collaborative Protégé chat functionality apart from other chat clients is the support for sending internal and external links. An internal link points to an ontology component. In the example in Figure 4, one of the users sends an internal link to the `Gene` class. The other user who is receiving the message can click on the internal link

and see the definition for the class mentioned in the chat. Thus, users can see the full context of the discussion in the chat.

7 Evaluation

We have performed the formative evaluation of Collaborative Protégé in the context of the ATHENA-DSS project. ATHENA (Assessment and Treatment for Healthcare: EvideNcebased Automation) [7] is a clinical decision-support system that generates guideline-based recommendations for the management of patients suffering from some clinical conditions. The system, developed as a collaboration between VA Palo Alto Healthcare System and Stanford University since 1998, is integrated with the VA's Computerized Patient Record System for a clinical demonstration, evaluation, and use. Initially developed for the management of hypertension, developers are extending it to include the management of chronic pain and diabetes, and the screening of chronic kidney disease. The end-users of the system are clinicians who are making decisions on the management of care for patients.

ATHENA-DSS developers use Protégé to build and maintain their knowledge base. The team of clinicians and knowledge engineers start with the narrative of a clinical guideline and distill this narrative into a set of related Protégé classes and instances that represent the guideline formally. Currently, the developers use an MS Access database to save the recommendation text and the associated annotations that they create. Thus, the information is spread across different tools and it is not linked. As the developers formalize medical concepts, such as diseases and drugs, and instantiate guideline recommendations as parts of flow-chart-like clinical algorithms, they have to work closely with one another, making sure that they do not overwrite one another's work. As the knowledge bases evolve, the developers have to ensure that the recommendations and annotations in the MS Access databases and Protégé knowledge bases are in synch.

As Collaborative Protégé became available, the team of one clinician and two knowledge engineers evaluated it over the period of one month. The three users actively used the tool during the evaluation period. They had access to the web pages that briefly describe the tool[3] but they did not have any training on how to use Collaborative Protégé. They were experienced users of the regular Protégé tool.

After the evaluation period, we conducted extensive interviews with the users to gauge their level of satisfaction with the tool, to understand how they used the it, to learn which features they liked and did not like, and to get new feature requests from them. In addition, we examined the annotations and the changes that the developers produced during the evaluation period to determine how they used the annotation and discussion feature, what was the nature of their posts, and how much of their time spent with the system was spent on collaboration activities compared to modeling activities.

8 Results

During the evaluation period, the developers entered 22 comments. All comments were comments on instances. There were three short discussion threads. We observed two

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

main uses for the comments in this project. First, the developers used the discussion feature to ask each other questions. For instance, the clinicians described some modeling problems and asked the knowledge engineers for the best ways to model the situation. Conversely, the knowledge engineers asked about some clinical concepts that they needed to represent.

Each clinical guideline has a narrative description and a set of qualitative parameters. The ATHENA-DSS developers represent each guideline as classes and instances in the ATHENA-DSS knowledge base. The developers found that annotations provided a good way to record the narrative and the parameters of the original guideline and to link them to the ontology components that represent the guideline. In a sense, the information about the original guideline provided the background information for ontology components, and annotations were a natural way to represent this link. The ATHENA-DSS developers currently store the information on the original guidelines in an MS Access database and they wanted this information to be accessible during ontology browsing. Because the reference guideline contains not only text, but a number of additional fields, we used the flexible design of Collaborative Protégé to produce a custom-tailored annotation type for ATHENA-DSS. We created a subclass of the `Annotatation` class, a `GuidelineComment` class. This subclass contained the fields specific to that type of annotation, such as quality of evidence and recommendation code. Because the Collaborative Protégé implementation simply displays the subclasses of the `Annotation` class as its available annotation types, we did not need to change any code to display the custom-tailored annotation. The ATHENA-DSS developers found this flexibility to be a particularly useful feature. They reported that they are now considering porting all the annotations from the MS Access databases to Collaborative Protégé as annotations. They cited several advantages of this approach in our interviews: First, they will be able to stay within one environment and not have to maintain the synchronization between the two sources. Second, they can see the reference source immediately as they browse the instances and can understand why the guideline was modeled the way it was. After we provided them with the new annotation type, about 25% of their comments were of this type.

In general, the members of the ATHENA-DSS team found Collaborative Protégé "very useful." They appreciated that the knowledge engineers could see the questions from the clinician in context of where the question was asked (rather than in an email, detached from the ontology). As one of the participants told us "It's just there, at the point where the problem is."

The ATHENA-DSS developers did not use the chat feature, mainly because they were never on-line at the same time. Another group that is currently evaluating Collaborative Protégé (the editors of the the NCI Thesaurus) found the chat to be one of the more useful features. The main difference between the two groups is that the second group is much larger and ontology development is their primary task. Thus, most editors are on-line editing the ontology during their workday.

In our interviews, the ATHENA-DSS developers indicated other potential uses that they see for the annotation features. These uses included recording detailed design rationale, having one developer explain to the others how he is approaching a specific

modeling problem in the context of the ontology, and having developers educate new users on the structure and intricacies of the ontology.

9 Discussion and Future Work

The analysis of the results, even from this fairly small evaluation period, points to several issues. First, users found Collaborative Protégé useful and did not require any special training to use it. We know that they did not find or use all the features that were available, and we expect that they would use the collaboration features even more extensively after a short training session (or with better documentation).

Second, the innovative use of Collaborative Protégé features points to the versatility of the tool. In fact, some of these uses prompted us to consider new features. For example, we might link the tool to an issue-tracker system, to enable users to see which task assignments have been made as part of the discussion, and to track their progress.

Third, the flexibility of the tool and the ease of extending it with new annotation types proved crucial in the ATHENA-DSS project. We envision that other users will create their own annotation types, with properties that are relevant in their settings.

One of the surprising findings for us (which we also observed in other settings) was that users do not add annotations to changes, but annotate only ontology components (in this case, instances). Even the rationale for changes themselves is recorded at the level of the ontology component, not the change or a group of changes. This observation suggests that users think in terms of ontology components rather than changes, even as they are closely involved in ontology editing.

In Collaborative Protégé, facilities for reaching consensus, recording design rationale, and noting outstanding issues are an integral part of the process of ontology browsing and editing. As users examine, say, a class in the ontology, they can immediately see all the discussion and questions pertaining to this class, whether there was any contention in its definition, alternatives that the authors considered. An editor, when coming upon a class that, he feels, must be changed, can post a request immediately, in the context of this class. This dual advantage of *context-sensitivity and archival* character of annotations adds the greatest value to Collaborative Protégé compared to discussion lists and issue trackers that are not integrated with an ontology environment.

Our infrastructure and the use of ontologies to represent many of the components that drive our software, enables other developers to reuse these components easily. Specifically, while Collaborative Protégé uses all the service ontologies described in Section 5, the service ontologies themselves are not specific to Protégé. We expect that other developers will reuse the ontologies in their tools, thus providing interoperability between the tools. For instance, different tools can implement their own mechanism for supporting or displaying discussions. If they use the same annotation ontology, then annotations created in one of the tools can be visible in the other tool.

There are many outstanding issues, however, that we must address in order to support truly collaborative ontology development.

In our original model, each annotation annotates a single object: a single class in the ontology, a single instances, a single other annotation. However, in the ATHENA-DSS use case a single guideline description could refer to different concepts such as

hypertension and diabetes. Thus, there must be a way of associating an annotation to several different objects. We do not currently have such support in the user interface. However, because annotations are simply instances, the `annotates` property can have more than one value and thus reference more than one object.

While we have a set of annotation types for proposals and voting, we do not have any workflow support for it. Our users (in ATHENA-DSS, and other projects) indicated that the proposals feature would be much more useful with such workflow support. For instance, when someone initiates a new round of voting, a workflow engine might inform other users that they are expected to vote, can tally the votes or wait for a certain period of time to elapse, and can produce the voting result.

Currently, Collaborative Protégé has only simple support for different user roles. In the future, we plan to adopt a policy mechanism that would enable us to describe privileges of users with different roles at different levels of granularity. For example, not all users in a project may have the privileges to create change proposals or to comment on the propsals. Some users may be able to edit only a part of the ontology. We plan to analyze the different scenarios and workflows that the biomedical ontology-development projects employ and add flexible support for roles and policies in future versions.

Finally, as we studied the different workflows that the projects described in the introduction to this paper used, one thing became clear: Developers of biomedical ontologies need tools that are flexible enough to work with different workflows. For instance, a group of users working together on developing an ontology in the context of a specific project will have different requirements compared to an open community developing a lightweight taxonomy that anyone can edit. In some cases, tools should support specific protocols for making changes, where some users can propose changes, others can discuss and vote on them, and only users with special status can actually perform the changes. At the other end of the spectrum are settings where anyone can make any changes immediately. Thus, tools need to support different mechanisms for building consensus, depending on whether the environment is more open or more controlled.

We are currently evaluating Collaborative Protégé in several other settings: the development of the NCI Thesaurus, the development of the Software Resource Ontology to be used by the NIH Roadmap's NCBCs, the development of the 11th revision of the International Classification of Diseases (ICD-11) at the World Health Organization, and other projects. These projects are all active ongoing projects and have different scope, workflow, the number of contributors, and so on. We expect to these evaluation to produce additional requirements for the tools and also to demonstrate innovative uses of the capabilities that we described here.

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 United States National Library of Medicine. Initial development of ATHENA-DSS for diabetes mellitus is supported by the Palo Alto Institute for Research and Education at VA Palo Alto Health Care System. Views expressed are those of the authors and not necessarily those of the Department of Veterans Affairs. We are indebted to the Susana Martins, Martha Michel, and Mary Goldstein of the VA Palo Alto Healthcare System for their help with the evaluation and for their insightful feedback on the tool.

References

1. S. Auer, S. Dietzold, and T. Riechert. OntoWiki–a tool for social, semantic collaboration. In the *5th International Semantic Web Conference, ISWC*, Athens, GA, 2006. Springer.
2. OBI Consortium. `http://obi.sourceforge.net/`.
3. K. Dellschaft, H. Engelbrecht, J. M. Barreto, S. Rutenbeck, and S. Staab. Cicero: Tracking design rationale in collaborative ontology engineering, 2008.
4. T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: Role based access control in owl. In *ACM Symposium on Access Control Models and Technologies (SACMAT 08)*, Colorado, US, 2008.
5. 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.
6. J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Interaction*, 58(1), 2003.
7. M. K. Goldstein, et. al. Translating research into practice: organizational issues in implementing automated decision support for hypertension in three medical centers. *Journal of the American Medical Informatics Association*, 11(5):368–376, 2004.
8. J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared web annotations. In the *10th International World Wide Web Conference*, pages 623–632, 2001.
9. 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 the *Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2007)*, volume LNAI 4788, pages 200–209. Springer-Verlag, 2007.
10. N. F. Noy, A. Chugh, and H. Alani. The CKC Challenge: Exploring tools for collaborative knowledge construction. *IEEE Intelligent Systems*, 23(1):64–68, 2008.
11. N. F. Noy, A. Chugh, W. Liu, and M. A. Musen. A framework for ontology evolution in collaborative environments. In the *5th International Semantic Web Conference (ISWC 06)*, Athens, GA, 2006. Springer.
12. R. Palma, J. Hartmann, and P. Haase. OMV: Ontology Metadata Vocabulary for the Semantic Web. Technical report, http://ontoware.org/projects/omv/, 2008.
13. Protégé. `http://protege.stanford.edu/`.
14. D. L. Rubin, N. F. Noy, and M. A. Musen. Protégé: A tool for managing and using terminology in radiology applications. *Journal of Digital Imaging*, 2007.
15. A. Sebastian, N. F. Noy, T. Tudorache, and M. A. Musen. A generic ontology for collaborative ontology-development workflows. In the *16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008)*, Catania, Italy, 2008. Springer.
16. J. Seidenberg and A. Rector. The state of multi-user ontology engineering. In the *2nd International Workshop on Modular Ontologies at KCAP 2007*, Whistler, BC, Canada, 2007.
17. 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.
18. E. Sunagawa, K. Kozaki, Y. Kitamura, and R. Mizoguchi. An environment for distributed ontology development based on dependency management. In the *2nd International SemanticWeb Conference (ISWC2003)*, Sanibel Island, FL, USA, 2003.
19. C. Tempich, E. Simperl, M. Luczak, R. Studer, and H. S. Pinto. Argumentation-based ontology engineering. *IEEE Intelligent Systems*, 22(6):52–59, 2007.