Interactive and Collaborative Ontology Development

Fan Bai, Zoulfa El Jerroudi

bai@interactivesystems.info, eljerroudi@interactivesystems.info

Abstract: Ontology developing and managing usually evolves a series of brain storming discussions, competitions, proposals, votings and other collaborative activities. These activities represent the main challenges of a platform, which enables users to develop and manage ontologies. In this paper we describe requirements for a collaborative ontology developing platform, and introduce an approach for collaborative ontology creation and management based on *Ontoverse* platform.

1 Requirements for Collaborative Ontology Developing

A typical reason for constructing ontology is to give a common language for sharing and reusing knowledge about phenomena in the world of interest. Concepts and their relationships are further described in terms of axioms and constraints that may be expressed formally. To design and develop ontologies is usually a joint effort reflecting experiences and viewpoints of several persons who intentionally cooperate to develop it. Chances for relatively wide acceptance are enhanced if these persons are argeed in the contributions they made. This helps reduce blind spots in the ontology and enrich its content. However these co-working requirements increase the complexity of ontologies creation and management. The following requirements should be considered in the collaborative ontology working environment:

1. **Synchronous/Asynchronous Information Exchange**. Information exchange is always the most important fact in collaborative working. A system which supports collaborative ontology development should offer functionality so users can communicate with each other. Consider the distribution of area and time, both synchronous and asynchronous way of information exchange should be supported.

2. **Concurrency Control**. When several users are working on the same content, conflicts can be occured. A user's result can be deleted or covered by other users' modification. The system can manage it in different ways, either to prevent the happening of the conflicts or tolerate the conflicts and manage them automatically or manually.

3. **Private/Shared Workspace**. Users usually prefer to work in a private workspace to get their own results and share it to others later. It is necessary to separate private and shared workspace for team members.

4. **Group Awareness**. If a user is working in a collaborative environment, it is always useful that he knows what other team workers are doing now and to get an overview about the project work flow. The system should support such awareness which lets each team member understand the activities of others and recognize changes.

5. **History Tracing**. Nothing can be done by only one step. To recognize what happened in each step will give users the awareness of how they get the result. The system should give users the possibility to find out who did what and when.

6. **Version Control**. When some mistakes happened, users wish that they can rollback their result to previous version to start again. The system should contains each version and offer the possibility to manage them.

These requirements represent the major challenges for systems that support users in creating ontologies collaboratively.

There are several systems trying to offer an environment in which users can construct knowledge in structured or semi-structured way. The ontology editor Protégé has two approaches to support collaborative working with ontologies. One is Co-Protégé [DBC06], it is a plugin for Protégé which supports users to publish their own resources to a shared workspace. Another approach is Collaborative Protégé [TN07] which bases on client-server architecture and allows a client to access ontologies on the server synchronously. Both support basic communication means, e.g. a chat or a discussion forum are supported. However they do not support version control, and does not use any locking mechanism to control concurrency. The Collaborative Protégé does not support private workspace. Other approaches are using wiki systems, such as OntoWiki [ADR06]. It is possible to create, delete and edit ontology entities by using form based PHP websites. However since it based on the HTTP protocol, it can not offer real time group awareness.

Aiming at combining advantages of different exited approaches and providing additional collaborative features, the **Ontoverse**[1] project provides a platform focusing on the support of communities consisting of domain experts as well as ontology designers to collaboratively design, create and manage ontologies in an easy, convenient and flexible way. The community members may discuss, share and exchange rich information objects, meet people and manage their contacts in a web page forum environment. It also provides a web-based editor, that enables users to collaboratively work on a shared ontology. The editor supports a instance message tool, concurrency control and group awareness, it also supports sub-shared workspace in the near future. It is useful when large number of developers are working on the same project. Developers can work in several groups and each group works in one sub-shared workspace. Every time a group commits its result a new version ontology will be generated in the public workspace. In the following, we will describe an scenario of how a user works with the Ontoverse collaborative platform.

[1]The project is funded by the German Federal Ministry of Education and Research (project no. 01C5975).

2 Scenario: Collaborative Working in the Ontology 'Bio2Me'

A user, called Tim, were invited into an ontology project BIO2ME[2] in the Ontoverse platform. He logs in the platform and enter the BIO2ME (Bioinformatics Ontology for Tools and Methods) project section. He regards all topics about this project and reply some topics that belong to his own domain knowledge. After that, he writes a mail in the platform to a expert who is working in the same project and tell him that some classes may not proper. Then he load the editor from the platform shown in Fig.1. Tim selects a sub work space so called *Data*, his own domain knowledge. In the instance message tool at the right side of the Fig.1 he finds that there are several users are working on the ontology already. Some of them are discussing a class in a instance message tool embedded in the editor. Tim notices that class *NumericMetricData* is quite *hot* since in the left part of the editor it's font is larger than others, which means many operations have been done on it.

He want to edit the class *Data*, an *lock icon* is shown on before the class hierarchy, which means no modification can be done. Tim knows that someone else is working on it right now so he could add changes directly. He decides to skip it and starts to edit the sub class *NumericMetricalData*, a lock is automatically set on it so it cannot be edited by others to avoid conflicts. Some minutes later an exclamatory mark is shown on the class *StructureData* so that other users realize, that there are same changes on it and then a warning message is transferred by instance message tool, where another developer try to warn the developers that this class is not consistent . After several hours all developers agree that this is the first version for their domain area and the manager commit it into the public work space.

3 Ontoverse: Editor for Collaborative Ontology Editing

In order to support such collaborative environment, both backend and frontend support is needed. In the platform described above a loosely coupled agent system based on the TupleSpaces approach and a visualization based on the Java applet technology are used, both are described in this section. The *visualization* of the ontology editor is supported by an application based on the Java Applet technology. It supports several collaborative features, such as *concurrency control*, *group awareness* and *instance message tool*, etc. Cooperated with the web forum in the Ontoverse platform, it supports a collaborative environment for ontology edition and management.

Concurrency Control: In the editor we use a *locking* mechanism to support concurrency control. Locking mechanism will prevent the happening of the conflicts. There are two mode of locking. One is called automatic locking, that a lock will be set on the resource automatically if a user start to modify it and the lock will be released automatically when the user finished. Another one is called manual locking, that a user set a lock on a resource manually and the lock will be stay still unless the user release it manually again. If a lock is set on a resource, all modification functionality will be blocked and the ontology

[2]This ontology describes bio-informatical knowledge created by the Ontoverse project.

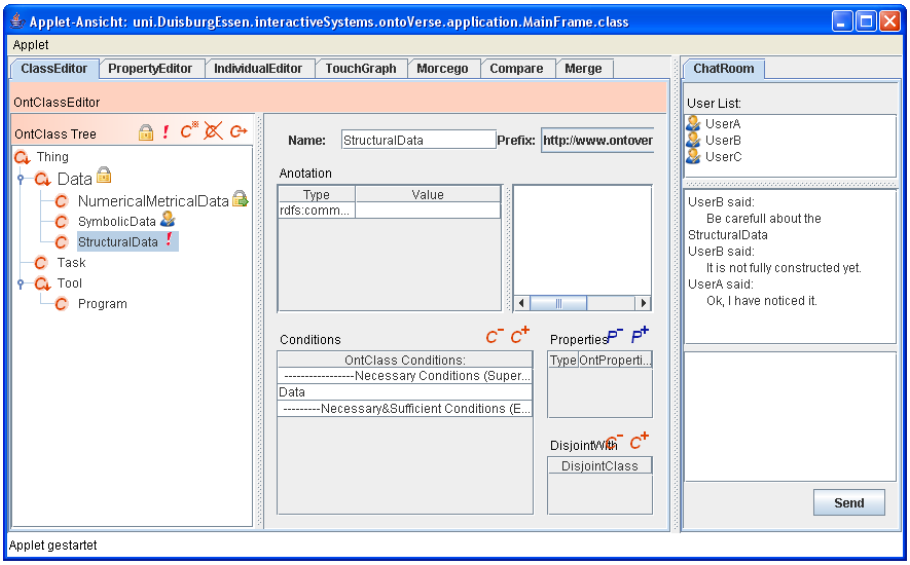

Figure 1: class editor of the Ontoverse ontology editor with some ontology-awareness features

resource can be read only. Locked resource will be marked a lock icon on the hierarchy tree interface, which shows in the Fig.1.

Group Awareness: The editor supports *Group Awareness* by immediately showing ontology resources, which have been created or edited or deleted by other users, in the user's editor interface. Once other users modified a resource, it will be shown in the hierarchy tree interface with a collaborative icon marked on it. The icon will be vanished when the user select the resource to see detail information. Notice that it is not necessary for a user to see every modification made on the ontology. Right now we use a simple mechanism that only the resources which the user see in the tree interface before will be marked an icon on it. Another possible alternative approach is let users define their own interesting area, and only resources in this area will show group awareness.

Synchronous/Asynchronous Information Exchange: An *Instance Message Tool* is also supported in the editor, usually it works with another awareness functionality called *Highlighting*, which supports users to capture other users' attention by setting an flashing exclamatory mark icon on a particular resource. Users can also find out who are working with them now in the instance message tool. Asynchronous collaborative information are not yet implemented though some of them are already stored in the application model such as modification history.

Ontology management & Version Control: *SWAT&SQLSpaceses* [MWH+07] are the backend of the Ontoverse. The main aim of this backend was to be as flexible as possible to integrate several features in a loosely coupled way into one main component, that provides the basic functionality of *ontology management*. The basic architectural idea is to use a *blackboard* [EHRLR80] system, which publishes relevant information in a space

accessible by several agents or processing units using read and write operations. It is implemented using the *TupleSpaces* approach, incorporated by Java and originating from Gelernter's Linda language [Gel85] in the Ontoverse project. *SQLSpaceses* is a implementation of *TupleSpace* and *SWAT* (SemanticWeb Application Toolkit) is a agent based on it. There are three main spaces in SWAT, ontology spaces (for each ontology one space) the actual ontological data is stored in form of RDF triples. The session space contains all process-related data like *log events, modification events, lock events*, etc, command space acts as a coordination channel for all participating agents. Version control will be combined with another functionality that enable users working on different group in one large ontology. Users can check out several sub-spaces from the main space and working in one of them. It is quite similar to the working group environment that several groups are working on the same large project and each group is working on certain sub project. Version control will help users to check out and check in the sub-spaces from main space. This will be implemented soon.

4 Summary and Outlook

In this paper we described an environment in which ontology developers and experts are working together in a synchronous and asynchronous way. With the help of forum and user management in web interface it offers a platform for all ontology developers and experts to exchange their minds and experiences. The embed ontology editor based on a Java applet supports several collaborative features to help ontology developers work closely together, such as *concurrency control* and *group awareness*. To support these activities a backend of SQLSpaces and SWAT is used. However which collaborative features should be supported and how to support them is always a big challenge for system designers.

Based on these results further work goes in the following two directions: First, how to give functionality to help users finding the edition history is still not finished right now. History tracing helps users to understand the procedure of the ontology developing and is quite important for a collaborative environment. Second, the sub-shared workspaces are not supported in the graphic user interface yet, and a fully version control mechanism based on it is not completely defined. We aim to complete these functionalities in the near future.

References

[ADR06] Sören Auer, Sebastian Dietzold, and Thomas Riechert. OntoWiki - A Tool for Social, Semantic Collaboration. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 736–749. Springer, 2006.

[DBC06] Alicia Diaz, Guillermo Baldo, and Gerome Canals. Co-Protégé: Collaborative Ontology Building with Divergences. In *DEXA '06: Proceedings of the 17th International*

Conference on Database and Expert Systems Applications, pages 156–160, Washington, DC, USA, 2006. IEEE Computer Society.

[EHRLR80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Comput. Surv.*, 12(2):213–253, 1980.

[Gel85] David Gelernter. Generative communication in Linda. *ACM Trans. Program. Lang. Syst.*, 7(1):80–112, 1985.

[MWH+07] Nils Malzahn, Stefan Weinbrenner, Peter Hüsken, Jürgen Ziegler, and H. Ulrich Hoppe. Collaborative Ontology Development - Distributed Architecture and Visualization. In *Proceedings of the German E-Science Conference*. Max Planck Digital Library, 2007. Open-Archive-Publikation.

[TN07] Tania Tudorache and Natasha Noy. Collaborative Protege. In *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at WWW 2007*, Banff, Canada, 2007.