IADIS International Journal on WWW/Internet
Vol. 5, No. 2, pp. 86-99
ISSN: 1645 – 7641

A COLLABORATIVE KNOWLEDGE SHARING FRAMEWORK WITH DIVERGENCE SUPPORT

Alicia Díaz and Guillermo Baldo *Lifia, Fac Informática – UNLP, CC 11, 1900 La Plata, Argentina*

[alicia.diaz, guillermo.baldo]@lifia..info.unlp.edu.ar

Gérôme Canals *Loria, Campus Scientifique, B.P. 239, 54506 Vandœuvre-lès-Nancy cedex, France*

gerome.canals@loria.fr

ABSTRACT

The focus of this paper is Collaborative Knowledge Sharing systems and how their usability can be improved by supporting of knowledge divergence occurrences. This approach recognizes divergence occurrence as a natural source of new knowledge in knowledge sharing communities. This paper presents a framework for sharing knowledge based on an explicit process model that governs and coordinates users' actions. The process includes operations for externalizing new knowledge and making complementary or divergent knowledge public. This framework conceptualizes a knowledge sharing process-oriented groupware application which supports the development of a knowledge repository collaboratively. In order to show how this framework can be used, we have instantiated it by ontologies as a knowledge representation paradigm. Finally, Co-Protégé, which is a prototypical groupware application based on the framework, is introduced.

KEYWORDS

Knowledge sharing communities, collaborative edition, collaborative ontology edition

1. INTRODUCTION

In this paper, we focus on Collaborative Knowledge Sharing systems and how their usability can be improved by supporting the occurrence of knowledge divergence. Collaborative knowledge sharing (CKS) systems are groupware applications that support the development of a shared knowledge repository. This functionality is the core of any Knowledge-Based system in which a group of people, that share a domain of interest, develop a knowledge memory in

collaboration. There are many potential applications of such a system: on-line communities, project memory support, corporate knowledge portals within others.

We understand knowledge sharing activity as a collaborative activity through which a group of people develops a common understanding about a domain of interest. The idea of sharing knowledge goes hand in hand with the notion of cognitive conflicts: people sharing knowledge also express divergent opinions about a topic of interest. From a computational point of view, divergence occurrences correspond to the occurrence of contradictions or inconsistencies in the repository. Traditionally, CKS systems deal with inconsistencies simply by avoiding them or keeping them out of the scope of the system; only agreed knowledge is hosted by the system. However, we claim that conflicts and divergence occurrences promote more actively shared workspaces: conflicts are an important source of new knowledge; their resolution generates more collaborative interactions among peers and improves the motivation for participating in this kind of activity.

Consequently, we propose an approach based on the explicit representation of conflicts (as divergence perspectives of a topic) and their resolution process. Therefore, the shared knowledge-base does not only store knowledge about a particular domain, but also it stores knowledge about the way it was proposed, discussed, augmented and finally agreed.

There are some requirements to take this work to action. The first one is that *the process of sharing should be explicit and controlled*. This is to organize and control users' actions and interactions, but also to make possible the explicit representation of the conflict solving steps in the knowledge memory. In order to stimulate a dynamic knowledge sharing activity, the second one is to keep participants aware of the shared knowledge status and its evolution and of the process activity. Finally, a last requirement is the need of having a *unified formal model* which allows representing both domain knowledge artifacts, the conflicts and their solving steps.

Based on these requirements, we designed a process-driven framework for supporting the knowledge sharing activity. This framework is based on an explicit process that defines steps and basic operations for the manipulation of a shared knowledge memory, an *ontology-based knowledge model* that includes conflicts and discussion representation, and a *dedicated awareness mechanism.*

The paper is organized as follows. Sections 2 introduce the process which will drive the development of the knowledge sharing activity. Then, in section 3, it is presented the collaborative knowledge sharing framework which joins fundamental components (knowledge-sharing workspace and the divergence management and the awareness components) of a groupware application which supports the knowledge sharing process through which a knowledge repository which is developed in collaboration. The section 4 completes the domain knowledge with knowledge about the activity, people and discussion. Section 5 deals with the particular realization of the framework through an ontology formalism. Section 7 is dedicated to show a real implementation of this approach by the introduction of Co-Protégé. Finally, in section 8, we present some conclusions.

2. KNOWLEDGE-SHARING PROCESS WITH DIVERGENCES

In this section we will introduce the knowledge sharing process which describes the process that is carried out by a group of participants in order to build a knowledge repository in collaboration.

The knowledge sharing activity can be described as a spiraled process where knowledge keeps emerging in each cycle. This process describes an *augmentative* building of the common understanding through the contribution of "knowledge". People always add more knowledge in each contribution, whatever this contribution means. Let's see a typical example where people share knowledge. It is a small community that shares knowledge about tools, experiences and news in the CSWC field:

> *Ale has just found out tikiwiki, a wiki environment with a forum, e-mail, etc. He has sent an email to the community announcing his discovery; consequently, Rick has also said that tikiwiki is similar to JSPwiki3 (another groupware tool) since it has comparable functionalities to JSPwiki; whereas Diego has said that tikiwiki is not exactly similar to JSPwiki, because although they share many functionalities, they do not share all of them.*

By observing, this scenario, we can remark that this activity is a collaborative learning process by means of which the community accumulates knowledge and develops a common understanding [Díaz 2004, Stahl 2005]. This process is an iterative and incremental process, which shows how the knowledge is exchanged among the participants and how knowledge is converted from tacit to explicit knowledge. However, this process also shows the reflection among individuals which is fundamental to achieve a common understanding.

When the knowledge sharing activity is computer supported by the collaboratively development of a knowledge repository, it is possible to remark that knowledge moves from private knowledge contexts to the community one and comes back to individuals again. At the same time, knowledge is no longer tacit to become explicit and then becomes tacit again [Nonaka 1995]. In order to capture this, we suggest the knowledge sharing process as the one to describe how a group shares knowledge at the same time that it develops its own knowledge repository. This is a spiraled process, made up of 4 steps: externalization, publication, internalization and reaction.

Externalization - from tacit to explicit knowledge in the private context. Externalization is an individual and private activity through which a knowledge unit, which is tacit in the individual knowledge context, becomes explicit as a knowledge artifact. A *knowledge artifact* is the minimal unit of "explicit" and exchangeable knowledge (see details in section 3). An example of externalization is when Ale writes an e-mail to communicate he has found out tikiwiki. Here, an informal knowledge representation system was used.

Publication - from private to shared knowledge context. Publication is the act of making public some externalized knowledge. It corresponds to the submission (making an *augmentative contribution*) of a knowledge artifact from the private to the shared knowledge context. In the scenario, Ale has published his discovery by sending an e-mail to the community.

Internalization - from explicit to tacit knowledge and from shared to private context. Internalization is an individual and private process, which takes place when individuals realize and acquire the subject of a new contribution. Internalization makes knowledge go from

community knowledge context to the individual one. This is the case of Rick who had realized about Ale contribution before making his contribution.

Lastly, *reaction* is the act of giving some kind of response to a previous contribution. It always involves an externalization and an eventual publication. Reactions can be private, this means that it only produces some change at individual knowledge context; or it can be public when it is published. Public reactions involve contributions, which we call *contributions by reaction*.

There are many causes for a reaction occurrence. It can be either motivated to complement a previous contribution (like Rick) or to give a divergent point of view (like Diego) or just to provide arguments for the original contribution. Those reactions that provide other points of view enable the occurrence of divergences.

Depending on whether the reaction is private or public; there is either a private or public divergence. A public divergence implies the coexistence of divergent knowledge artifacts.

Any reaction is triggered from a previous contribution and the knowledge sharing process's cycle describes a *discussion thread*.

In figure 1, readers can see a schematic view of the knowledge sharing process.

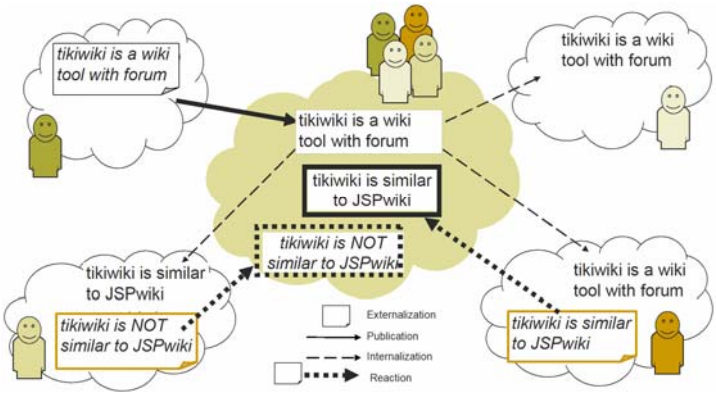

Figure 1. A schematic view of the knowledge sharing process.

3. THE COLLABORATIVE KNOWLEDGE SHARING FRAMEWORK

The collaborative knowledge sharing framework is a conceptual framework that joins fundamental components of a groupware application which supports the knowledge sharing process through which a knowledge repository is developed in collaboration. These components are: the *knowledge-sharing workspace*, the *divergence management component* and the *awareness component.*

Before describing the three main components of the framework, we will take a moment to describe knowledge artifacts as the minimal unit of exchangeable knowledge. Depending on the level of formalization of the knowledge representation system, a knowledge artifact can be: informal, where knowledge is strong hard-coded (i.e. wikis), semi-formal where informal

knowledge representation is mixed with formal representation, for example documents are classified by domain ontologies (i.e. semantic wikis) or typed messages; or formal where knowledge is represented by a formal knowledge representation system (i.e. by means of a domain ontology, as it will be shown in section 5). However, considering the knowledge artifact as a unit which encapsulates knowledge, it is possible to conceive the knowledge sharing framework without presupposing any knowledge representation paradigm. Thus, the knowledge repository can be understood as a collection of knowledge artifacts arranged according to the formalization system and the discussion thread.

3.1 The Knowledge-Sharing Workspace

The knowledge-sharing workspace is a process-driven shared workspace that supports the collaborative development of a knowledge repository. It is based on the knowledge sharing process which was described in section 2.

As externalization is a private activity and publication affects the public context, we conceive the workspace made up of a public workspace and many private workspaces. The *public workspace* is a shared workspace that is unique, accessible to everyone and contains the shared knowledge repository (whose edition is only achieved by publishing knowledge artifacts). On the other hand, the *private workspace* is a non-shared workspace (only accessible by its owner) and hosts a private knowledge repository which represents the private view of the shared one. Private knowledge repositories can differ from the shared one, but they can have overlapped parts.

According to the structure of the workspace, the execution of public actions is perceived by any member, but the execution of private actions is hidden to the other members. The main private actions are those to externalize a knowledge artifact, and thus, the private repository is developed by the direct edition of knowledge artifacts. Private actions are dependent on the knowledge representation system. On the other hand, the main public action is the publishing action. Publishing means making a contribution of a knowledge artifact from a private repository to shared one. They involve changes at the shared repository.

Management of private and shared knowledge repositories has a direct consequence: each member manages two different knowledge versions that coexist: private and shared versions.

Any contribution involves “merging” the contributed knowledge artifact with the shared knowledge repository. The resulting merge should provoke an *augmentative* version of the knowledge repository. A contribution is augmentative if it can be *integrated* to the shared repository without introducing any divergence (contradiction). Let us come back to the example, the Rick’s contribution can be integrated to the shared knowledge repository without introducing any conflict, because it complements existing knowledge. On the other hand, Diego’s contribution will introduce a contradiction in the shared knowledge repository. This last contribution means a divergent perspective.

In our approach, we suggest that both *augmentative and divergent contributions* coexist in the knowledge repository. Therefore, mechanisms to check the integration viability are required. Each time a publishing action takes place, it is necessary to “check” whether it involves an augmentative contribution. This checking is strongly dependent on the knowledge representation system proposed (further details in section 5). Contributions that pass this checking can be merged to the shared repository without any inconvenience. On the contrary, non-augmentative contributions should be rejected or should be explicitly contributed as a

divergent contribution. The divergent management component is in charge of dealing with this last case allowing the occurrence of divergences.

3.2 Divergence Management Component

Due to the shape of the workspace, divergences can occur in two senses: it can be a private or a public divergence. *Private divergences* are those which remain private in the private knowledge repository. They are the simplest and easiest ones to support because both repositories are in two differentiated workspaces. On the other hand, a *public divergence* is a divergence in the shared repository. This means having alternative knowledge artifacts in an augmentative fashion. To achieve this, the underlined model must provide suitable primitives.

In our approach, non-augmentative contribution is published explicitly as a divergent contribution. Divergent contributions are attached to a special kind of knowledge artifact, the *discussion artifact*, which encapsulates a divergent knowledge artifact. Discussion artifacts are the resources by means of which divergent versions can coexist in the shared repository.

The second goal of the divergence management component is to put divergent contributions in the context of a discussion. The resource to manage this is the *discussion thread*. The discussion thread model provides a simple yet formal structure for the discussion and exploration of shared knowledge. Discussion thread is in charge of linking the discussion artifacts and identifying the role of the contributed artifact in the context of the discussion (initial artifact, augmentative artifact, divergent artifact and argumentations). The discussion thread is an aggregation of augmentative and/or divergent contributions. The figure 2 shows the discussion thread developed in the example.

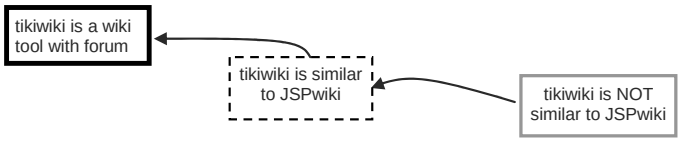

Figure 2. The discussion thread of the scenario. The black rectangle represents the initial contribution, the dashed one represents the next augmentative contribution and the grey one is the divergent contribution.

This discussion activity establishes a set of new knowledge sharing actions: the discussion actions. Discussion actions refine reaction step in order to allow the development of the discussion thread. They are comprised by two groups: the *opening discussion action* and the *discussion actions*. To trigger a discussion thread, it is necessary to make the identification of the initial discussion artifact. Then, the linking of the discussion contributions will take place. The identification of the initial contribution involves the identification of the knowledge artifact to be set "in discussion". This identified knowledge artifact becomes the initial discussion artifact and the action is the opening discussion action. Discussion contribution linking means attaching the new discussion contribution to an initial contribution.

3.3 Awareness Services

The third component of the knowledge sharing frameworks is in charge of the awareness services. Awareness is a relevant component of any groupware application; it keeps users up-

to-date about the collaborative activity [Schmidt 2002]. Then, it will be useful to keep people aware of the knowledge changes and discussion evolution. In this approach we have identified and defined the specific requirements of awareness services for CKS systems. These awareness services should be seen as an engine of the knowledge sharing activity and as a facilitator of the internalization (which promotes the reaction) because they have to be in charge of noticing about new contribution occurrences and of identifying highly-active concepts. But also, awareness should be useful to complement the support of divergences, it should make the divergence occurrence evident– to help people keep the discussion context.

Therefore, we have defined the awareness of knowledge sharing activity as the needed awareness information to keep a knowledge-sharing community up-to-date about the knowledge evolution. This awareness plays a crucial role because it is a means to internalize and externalize knowledge. Indirectly, pushing internalization is a way of pushing also the knowledge sharing activity, because internalization becomes the seed of reaction occurrence; and thus, more knowledge is provided, either augmentative or conflictive. Beside, awareness will take into account knowledge discussion occurrences and thus, it is in charge of making divergences acceptable; it reinforces the occurrence of interaction among people. It comprises, together with the conflict occurrence, the means to improve the knowledge sharing activity.

In order to track the collaborative activity, people need information about the historical context of the activity. To design the awareness service, we have made an analysis to discover which information is necessary to be tracked and captured when knowledge sharing activity occurs and how this information may be useful to the user (we have taken an analogous approach to the one in [Tam 2004]). This information is organized in low-level and high-level information. *Low-level information* serves to answer questions like: W*ho has contributed with this knowledge artifact? Which knowledge artifact has been contributed by a user?* On the other hand, *high-level information* could be deduced by mining the low-level information (Who has been contributed with this person? How this knowledge artifact has evolved? What are the more active topics?). High-level information is also useful to update de knowledge repository by adding activity knowledge (by adding high-level actions) and rendering the member profile (by adding new discovered interest). The member profile defined the interest of a user on knowledge domain, other users and discussions.

Each time a new action takes place; the awareness mechanism has to capture information about the performed action and stores it at the knowledge base. Then, the awareness mechanism delivers this information by means of *notifications* (*notifier* component). A notification is related to the action and attached to the users. People need to be notified according to the member profile and their activity.

Awareness services are also useful to aid people to externalize knowledge at their own individual knowledge repository. This means that a new contribution to the shared knowledge repository can be automatically incorporated to the individual one. Indirect externalization is complemented by the "*notifier*" component. But awareness services also have to help people become aware of differences between the private and the shared knowledge repositories. Because, any change at the shared repository may leave the private one out-of-date; or because any change at the private repository may only mean a private divergence. This service helps users to locate changes and divergences in her private repository.

4. KNOWLEDGE SHARING ACTIVITY IS PART OF THE KNOWLEDGE REPOSITORY

As it was mentioned in previous section, it is mandatory to capture information about the knowledge sharing activity. While the collaborative activity is carried out, knowledge about this activity has to be captured in order to maintain the history of the activity and to improve the collaborative activity. People also need to share information about who has contributed with what or how was discussing some topic, independently of the awareness services. Consequently, we conceived the knowledge repository not only as a domain knowledge repository, but also as a repository of knowledge about the members and their activities.

On the top of the workspace, the activity knowledge is captured in term of: performed actions, domain knowledge, people and the relationships between them. This knowledge is ontologically represented in generic ontologies [Corcho 2003] (Figure 3).

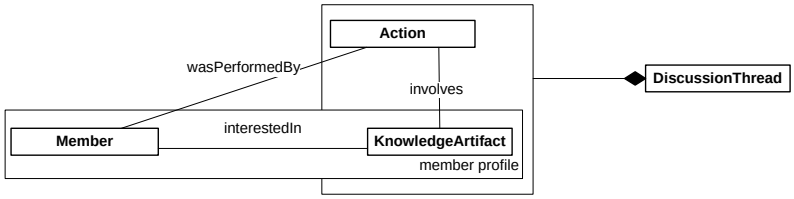

Figure 3. Schematic relationships among the domain knowledge, members and their activity.

In order to capture knowledge about the performed actions it is necessary to track each event that occurs in the workspace. This knowledge will be part of the activity ontology. *Activity ontology* represents knowledge about the performed action (*action*), who performed this action (*member*) and which *artifact* it involves. It also covers knowledge about the discussion activity. In order to complete the knowledge about the activity, the system also captures knowledge about members by means of the members' profile ontology and knowledge about conflict solving process through the discussion thread ontology. In the *user's profile*, people indicate what knowledge they are interested in (i.e. actions, knowledge artifacts, users and other users' activity).

Besides, previous knowledge is complemented by incorporating knowledge about the discussion activity. The discussion activity knowledge is captured by the specialization of the *knowledge artifact* in the different types of discussion artifacts and by the *discussion thread* concept. The discussion thread concept is defined as the aggregation of discussion artifacts. Finally, the action ontology is specialized in order to model the discussion contributions (Figure 4).

5. WHEN ONTOLOGIES ARE THE KNOWLEDGE REPRESENTATION FORMALISM

Choosing knowledge representation formalism is the first decision to make before facing the instantiation of the knowledge sharing framework. Each knowledge representation system proposes its own conceptual model, which defines the primitives to express the knowledge.

These primitives describe how the knowledge representation has to be understood and influence in the way of updating the knowledge repository.

Although there are different knowledge representation systems, we have chosen a formal approach by using *ontologies* [Corcho 2003, Staab 2004], because firstly, they allow developing a shared and common understanding of the domain of interest; secondly, they capture and formalize knowledge by connecting human understanding of symbols with their machine processability; and thirdly, because they ends up the knowledge sharing activity to a collaborative design activity --the conceptualization of the domain. This approach is in the address of other tools that allow the collaborative design of an ontology [Corcho 2003, Faquhar 1996] like WebOnto [Domingue 1998] and Apecks [Tennison 2002]. However, it takes into account the asynchronous development of the ontology and its discussion and it is more focused on the development of a shared ontology than in the development of personal ones. Besides, in most of these systems, the occurrence of divergences is avoided or the management of divergences and their negotiation mostly kept out of the shared ontology; and thus the shared ontology only captures the last update.

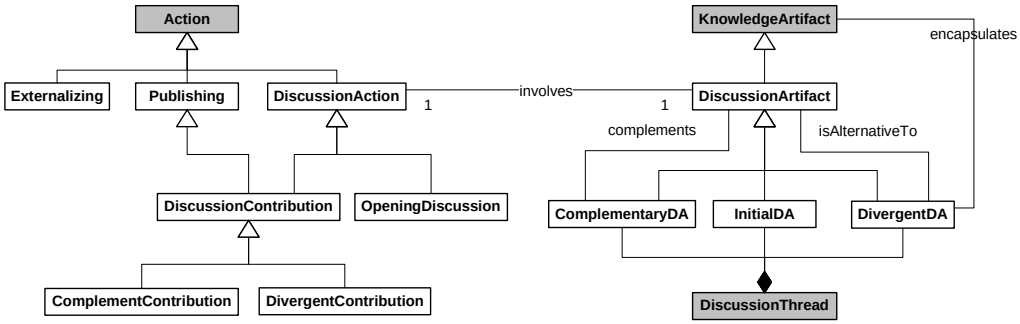

Figure 4. The Activity ontology and the Knowledge Artifact ontology and the Discussion Thread ontology.

Then, an ontological knowledge repository becomes a set of interrelated ontologies: domain ontology, activity ontology, discussion thread ontology, user's profile ontology. Particularly, the domain ontology will be developed collaboratively, being the subject of the knowledge sharing and discussion activity; while the other ontologies are pre-established and instantiated by the underlying system.

Now, when ontologies are used to represent knowledge, a knowledge artifact is seen as a formal conceptualization of a knowledge item in terms of ontological primitives, and it is called an *ontological artifact*. Then, an ontological artifact identifies the knowledge artifact of an ontological contribution.

According to the knowledge sharing workspace presented in section 3.2, people need to be able to manage both versions of the domain ontology: the individual domain ontology and the shared domain ontology, representing the individual and the shared knowledge repository respectively. Both kinds of domain ontology respect the same structure, but represent different knowledge spaces and they are developed following different modalities; while individual domain ontologies are developed by externalizing ontological knowledge artifact, shared domain ontology is developed by publishing ontological knowledge artifacts. Now, externalizing knowledge involves building a conceptualization of a knowledge artifact ---the direct edition of ontological artifact by ontological primitives.

On the other hand, publication involves "integrating" to the shared ontology an ontological artifact resulting in an augmentative version of the shared ontology without introducing any description mismatches.

An ontological contribution is augmentative if its publication must conserve the monotonic principle enunciated section 3.2.1. That is, it should avoid the occurrence of any ontological mismatch [Klein 2001]. Understanding an ontological artifact as an ontology, the problem of publishing an ontological artifact is reduced to combine both ontologies, the ontological artifact and the shared domain ontology (Figure 5 (2)). This combination can be done by *integrating* both ontologies, which means that they are merged into one "new version" of the shared domain ontology [Pinto 1999].

In short, if O_p denotes the private ontology and *oa* denotes the ontological artifact that exists at the O_p, the merging or integration of an ontological artifact *oa*, involves updating O_s (the shared ontology), by adding *oa*, where ontologies O_p and O_s may have overlapping parts. This overlapping between both ontologies can be innocuous or it can eventually make conceptual description mismatches arise. In the last case, alignment it is not possible. Therefore, to merge both ontologies it is necessary: first, to check the viability of the integration and secondly, to align the *oa* to the O_s. Aligning two ontologies involves updating the O_s by adding *oa*. As a result, there is a new version of O_s.

Dealing with ontologies, a divergent contribution means the introduction of an inconsistence at the shared ontology. Therefore, it is necessary to propose some approach that allows maintaining the coexistence of ontological divergence and simultaneously avoids the eventual occurrence of inconsistencies at the shared ontology. In order to tackle this situation, we suggest extending the ontological model with discussion thread primitives. These new primitives will be considered as a concrete resource to make explicit the divergence and become first order ontological primitives. The main advantage of having an ontological representation of the discussion thread is to encapsulate inconsistencies, since they remain encapsulated in an alternative ontological artifact.

Particularly, an ontological discussion thread identifies the initial ontological artifact and the divergent ontological artifacts. An initial ontological artifact encloses an ontological artifact that will be set "in conflict"; while an alternative ontological artifact encapsulates a divergent conceptualization of a particular objected ontological artifact. Figure 4 shows the discussion thread ontology.

To open a discussion, users are forced to identify the objected ontological artifact and then, they may express divergent position as alternative ontological artifact. Figure 5 shows the sequence of ontological contribution that describes a discussion thread.

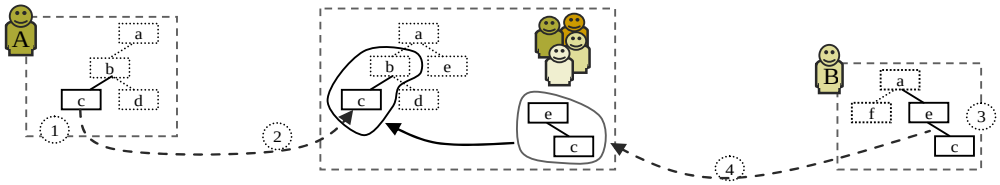

Figure 5. A discussion thread where with the black rounded area is represented the initial ontological artifact, and with the grey area is represented the divergent ontological artifact.

6. CO-PROTÉGÉ SYSTEM

Co-Protégé [Díaz 2006] in an extension of Protégé-2000 [Gennari 2003] which is a process-oriented groupware application based on the process that was described in previous section to edit ontologies and knowledge bases in a collaborative fashion. It is made up of: the workspace, which supports the necessary functionalities for externalizing and publishing; the divergence management component, which is in charge of making contribution by reactions (divergence occurrences and discussion threads) explicit and lastly the awareness component which facilitates internalization. Its visualization is in terms of tabs like in Protégé-2000 (Figure 6). There are tabs for modeling the shared-private workspace, the conflict tab, the user tab and the difference tab.

Shared-Private Workspace Tabs. Co-Protégé proposes tabs that "overlap" both workspaces in the same tab in order to make easily achieving to a direct manipulation of the two ontologies. Only the private side (on the left) has the same functionality as the Protégé-2000 to edit the private ontology; the shared side on the right) cancels them (the shared ontology is only updated by publications). There is one tab of this kind for each kind of frame (class, slot and instance). A conflict is created in the shared-private tab by selecting the set of frames that will be put in conflict. After that, the frames are shown "in conflict".

The system makes incompatibility checking each time a publication is performed in order to ensure an augmentative contribution. Whatever the checking result may be, Co-Protégé informs it at the bottom of this tab.

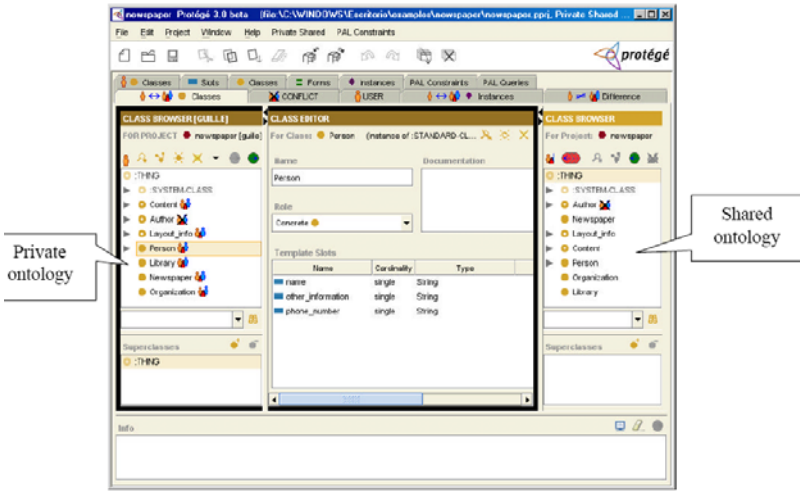

Figure 6. A snapshot of Co-Protégé. Both private and shared ontologies can be appreciated simultaneously.

User Tab is to manipulate the user profile. Users' interest can point to any kind of frame described by the metamodel of Co-Protégé, that is, elements of the shared ontology, other users, conflicts and conflict components. There are some cases where the system is able of updating the user's profile or making suggestions.

Conflict Tab defines a space where users can browse through the conflict and make it evolve. Once a conflict was created, it becomes part of the conflict list, where all currently open conflicts are enumerated. Users can add alternatives and argumentations. Alternative are created with frames from the private ontology. This is the mechanism that allows for the publication of divergent contributions.

Co-Protégé supports two visualizations of notifications. One indicates the degree of similarity/difference that ontological artifacts in both ontologies maintain in order to provide awareness of private divergence. This is shown over the private ontological elements. This visualization is rendered each time any change occurs at the private or shared ontology. The other visualization is more general and shows all the notifications in a chronological order. At the user's tab, users can specify different filters to show notifications.

6.1 Co-Protégé Implementation Features

Co-Protégé extends Protégé through the definition of some plugins by following Protégé extension philosophy. In Co-Protégé a project is made up of the shared ontology plus all private ontologies (one for each user). Co-Protégé is a client-server application, where a project is defined as a Protégé's metaproject. In this metaproject every ontology is defined (the shared and each private) together with the access permissions.

Co-Protégé uses the Protégé-knowledge model; however, it extends Protégé-2000 metamodel in order to provide primitives to model special primitives for modeling the shared ontology and conflicts. Besides, Co-Protégé defines a set of generic ontologies to model knowledge about the activity and user. Figure 7 shows the new classes which were added to the Protégé-2000 metamodel.

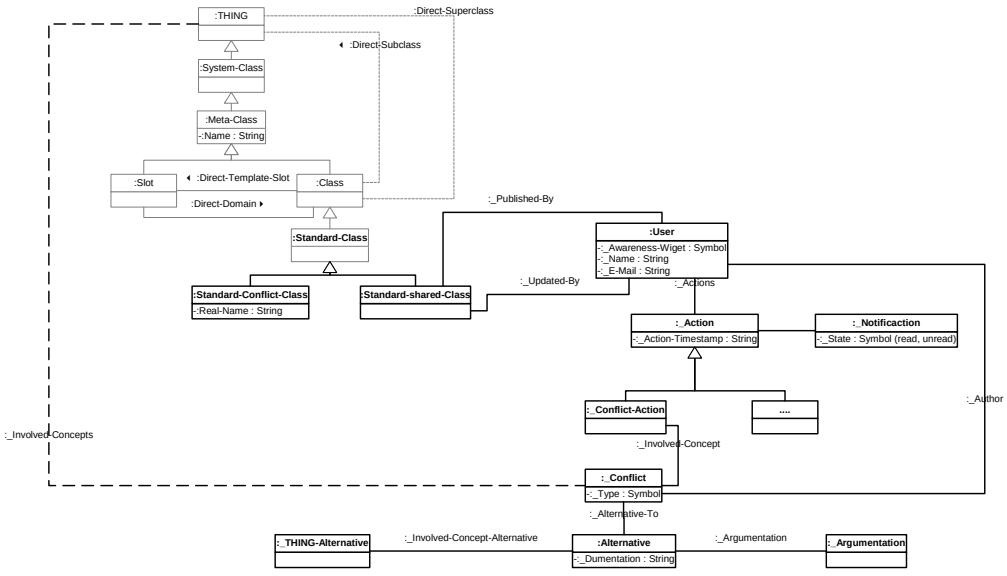

Figure 7. Co-Protégés' metamodel, model and generic ontologies. Original Protégé primitives are in grey.

Then Co-Protégé ontologies use the same types of Protégé's primitives (frames): classes, slots, facets and instances. However, Co-Protégé uses two different metamodels to model both ontologies: the private and the shared. Any private ontology is considered as a Protégé-2000 project; therefore private ontologies respond to the regular Protégé-2000 metamodel; for example, if a user creates a class DomainClass1 at his/her private ontology, class DomainClass1 is an instance of *:Standard-Class*.

However, *:Standard-Class* is not enough to model an ontology artifact that is at the shared ontology, because of they also need to model other relationships that manifest features of being a shared artifact in a collaborative process (i.e. creator or modifier (users) relationships or it is needed to know when it was created). In order to solve this, Co-Protégé has its own metaclass architecture that is an extension of the Protégé-2000 metaclass architecture. It is done through the addition of a set of new metaclasses. These new classes are shared-classes, shared-slots and shared-instances. This specialization adds the additional relationships that shared primitives need to be a frame of a shared ontology. These metaclasses makes a difference between the ontological artifact of the private ontology and those of the shared one. Every ontological artifact of the shared ontology is model by *:_Standard-Shared-Class*, *:_Standard-Shared-Slot* and *:_Standard-Shared-Instace*, however the remainder of concept are model directly with the metamodel architecture of the Protégé-2000.

Taking into account that the shared knowledge it not only knowledge about the domain of interest, but it is also knowledge about the collaborative activity, Co-Protégé incorporates primitives that model these particular kind of knowledge. Because of knowledge about the activity is independent of the specific domain of the discussion, the scheme of this knowledge should be applied to any environment that supports a knowledge sharing activity, whatever the domain of the interest of the community may be. Therefore, it is possible to design a generic ontology to model the activity. Concepts like users, actions, conflict, argument, alternatives, and others are modeled by this set of generic ontologies. These generic ontologies are strongly related to the metaclass architecture of the shared ontologies (i.e *:_Published-By*, *:_Involved-Concepts*). Knowledge about the activity in some cases is automatically (implicitly) captured according to users actions at the workspace or in other cases it is provided explicitly by users.

7. CONCLUSIONS

The knowledge sharing framework is a conceptual framework that describes the fundamental components of a CKS system. This framework is based on an explicit process model that governs and coordinates users' actions. The process model includes operations for externalizing new knowledge and making public new or divergent contributions. This process model is enacted on top of a workspace that includes a private part where users can externalize their personal knowledge and a public part where shared contributions, arguments and discussions threads are published. We have remarked that divergence can be accepted if the participants can build a clear understanding of the shared knowledge evolution. In that way, a special attention has been paid to the design of a dedicated awareness mechanism.

The concepts of this framework have been implemented in a prototype called Co-Protégé [Díaz 2006], where knowledge sharing activity is considered as the collaborative design of a shared ontology.

Although this approach seems very promising, it still needs to be evaluated. In this way, our immediate objective is to get more feedback about the use of the tool by real users. To do so, we are actually working on an enhanced version of the prototype that will support OWL.

Another future work is to adapt this approach to more general propose like collaborative design activities. While the participants are designing collaboratively, they are involved in a negotiation activity through which they share design artifacts, but they also share different points of view, alternative/divergent designs and knowledge about the design activity. When the design subject is based on a well-defined model, it is possible to extend this model by adding new primitives which model the activity for negotiating designs (alternative and divergent) and the knowledge about this activity. Therefore it is possible to state a CSCW approach to support a collaborative design activity which involves the negotiation of the design. This approach would be based on a well-defined conceptual model, the management of shared and private workspaces and the occurrence and the coexistence of divergent and alternative designs.

REFERENCES

Corcho O. et al. 2003. Methodologies, tools and languages for building ontologies: Where is their meeting point? *Data Knowledge Eng.* 46(1): 41-64.

Diaz, A. and Canals G. 2004. Divergence Occurrences in Knowledge Sharing Communities. *LNCS, Springer Verlag -Proceedings of Criwg'04* pp17-24

Diaz A. et al. 2006. Co-Protégé: Collaborative Ontology Building with Divergences. *Seventh International Dexa Workshop on Theory and Applications of Knowledge Management(TAKMA).*156-160

Domingue J., Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web, *in: Proc. 11th Knowledge Acquisition Workshop (KAW98),* 1998.

Farquhar A. et al. 1996. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *In: Proc. 10th Knowledge Acquisition Workshop(KAW96).*

Gennari J., M.A. Musen, R. Fergerson. 2003. The Evolution of Protégé: An environment for knowledge-based systems development. *IJHCS*, vol 58 (1): 89-123.

Klein, M. 2001. Combining and relating ontologies: an analysis of problems and solutions. *In proc.of the Workshop on Ontologies and Information Sharing, IJCAI'01,* Seattle, USA.

Nonaka, I. and Takeuchi, H.1995. *The Knowledge-Creating Company*, Oxford University Press.

Pinto S. et al. 1999. Some issues on ontology integration. *In Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99*, Stockholm, Sweden, August.

Stahl G. 2005. Group Cognition in Computer Assisted Learning. Journal of Computer Assisted Learning (JCAL).

Schmidt K. 2002. The Problem with Awareness Introductory Remarks on Awareness in *CSCW Journal* 11(3-4), 285_298. Kluwer Academic Publishers.

Staab S. and R. Studer (eds.). 2004. *Handbook on Ontologies*. Int. Handbooks on Information Systems, Springer Verlag,

Tam, J. and Greenberg, S. 2004, A Framework for Asynchronous Change Awareness in Collaboratively-Constructed Documents. LNCS, Springer Verlag -Proceedings of Criwg'04.

Tennison J. et al. 2002. APECKS: using and evaluating a tool for ontology construction with internal and external KA support. IJHCS. 56(4): 375-422.