CO-Protégé: A Groupware Tool for Supporting Collaborative Ontology Design with Divergence

Alicia Díaz[1,2] and Guillermo Baldo[1]

[1]Lifia, Fac. Informática- UNLP, CC 11, 1900 La Plata, Argentina
[2]Loria, Campus Scientifique, B.P. 239, 54506 Vandœuvre-lès-Nancy cedex, France
{alicia.diaz, guillermo.baldo}@sol.info.unlp.edu.ar

1 CO-Protégé: Introduction

CO-Protégé is a set of plugins that extends Protégé-2000 in order to support a collaborative developing of an ontology. In Co-Protégé people do not make a direct edition of the shared ontology, but also they change it through the publication of ontological artifact. Co-Protégé users manage simultaneously two ontologies the private ontology and the shared one. Private ontology can be edited in the private workspace as users were working in a stand-alone fashion. There is also a shared workspace where the shared ontology is updated through the publication of ontological contributions, even divergent contribution. Divergent contributions make possible the coexistence of many conceptualizations simultaneously at the shared ontology. They are the resource to discuss the design of an ontology. Most of collaborative edition functionalities are provided through a special tab, the shared-private tab that overlaps both workspaces and allows user to manipulate simultaneously their private and the shared ontology. Figure 1 is a snap-shot of the interface of Co-Protégé when a user has already logged into the system.

2 Co-Protégé: Foundations

Co-Protégé supports a collaborative process to develop a domain ontology. This process is based on the knowledge-sharing process (ks-process) described in ([Diaz04], [Diaz05]). The ks-process is a spiral process where a knowledge repository representing the common understanding of the group is developed collaboratively. This process has as premise that people move between two knowledge contexts: the individual knowledge context and the community knowledge context. Individual knowledge context represents the individual's point of view of the domain, while shared knowledge context is the group's point of view of the domain; thus both points of view can be different.

The ks-process consists of 4 steps: externalization, publication, internalization and reaction. *Externalization* is an individual and private activity through tacit knowledge becomes explicit. *Publications* means to share some externalized knowledge with the community. This externalized knowledge is called the knowledge contribution and goes from the individual knowledge context to the community one. *Internalization* is a private act, where individual realize of new knowledge occurrence. Finally, *reaction* is the act of giving some response (a new contribution) to a previous contribution.

Reconsidering the ks-process in terms of ontologies, we can remark:

– People need to be able to manage two versions of the domain ontology: the *individual ontology* and the *shared ontology*, representing the individual knowledge context and the shared knowledge context respectively. Consequently, there is only one *shared ontology*}, but there are many *individual domain ontologie*s, one for each member.

- Both ontologies are developed following different modalities; while *individual ontologies* are developed through externalization, *shared ontology* is developed through by publishing ontological contribution. Externalization involves the edition of the *individual ontologies* by the direct manipulation of the ontological primitives. Publication involves integrating to the *shared ontology* an ontological conceptualization coming from an *individual ontology*. This conceptualization is an *ontological contribution*. An ontological contribution is a portion of the *individual ontology* that will be *integrated* to the *shared ontology*.
- Discussion about the conceptualization of the ontology should be guarantee. It is also necessary to give account with a mechanism to contribute with alternative conceptualizations to the current shared ontology. These contributions are called *divergent contributions*. At the ontology, divergent ontological contribution will be augmentative and compatible with the remainder of the shared ontology.

Integrating an ontological contribution to the *shared ontology* is not a direct action, because sometimes it may cause some kind of *incompatibility* because it may not result in a monotonic extension of the target ontology. Any contribution should result in an *augmentative* version of the ontology without introducing any incompatibility. Understanding an ontological contribution as ontology, the integration is reduced to merge both ontologies. The augmentative ontology development is achieved if it is possible to automatically make the alignment of both ontologies without any conceptual structure mismatch [Klein01]. In such case the ontological contribution becomes *augmentative contribution*.

Fails can occur due to: *name mismatches* and *local context mismatches*. Alignment considerations are detailed in [Diaz05]. Fail alignments are useful to give an explanation of the causes of fails. In Co-Protégé, failed-combination information is useful to assist users to the take decision to re-enunciate the ontological contribution or publish it as divergent contribution.

The last case represents the enunciation of a conflict at the shared ontology and is the way to discuss a conceptualization. Any divergent contribution is always attached to a portion of the shared ontology, that to be discussed or set in conflict. To establish a discussion involves two steps: first, identifying the conceptualization to be set in conflict, and then, linking it with a divergent conceptualization. Giving argumentations is also part of the discussion activity, but they only help to give comments to support or dissent with some conceptualization, both an objected and divergent one. A divergent contribution at the shared ontology involves enabling the coexistence of two or more incompatible conceptualizations. Because, these incompatibilities remain encapsulated in a *divergent conceptualization* artifact, and thus, ontological incompatibility disappear.

3 Working in Co-Protégé

Co-Protégé imposes its own manner of carrying out the collaborative developing of an ontology. Users edit the private ontology in the private workspace and then, they can publish ontological artifacts to the shared ontology. Edition at the private workspace is carried out as users were working in a stand-alone fashion in Protégé-2000. The shared ontology is manipulated in a shared workspace and its "edition" is carried out through publications.

Co-Protégé visualization conserves the philosophy of Protégé-2000 (see Figure 1). There are tabs for modeling the shared-private workspace, the conflict tab, the user tab and the difference tab.

Shared-Private tabs. Co-Protégé proposes tabs that "overlap" both workspaces in the same tab. This kind of tab joins both the private and the shared ontologies, easily achieving to a direct manipulation of the two ontologies (Figure 1). To alternate between both ontologies is easy, because they are visually overlapped. Only the private side of a shared-private tab has the same

functionality that the Protégé-2000 to edit a single ontology; the shared side cancels them because the shared ontology is updated by publications. There is one tab for each kind of frame (class, slot and instance) and each one shows the two ontology versions: the private and the shared. They are the classes-shared-private tab, the slots-shared-private tab and the instances-shared-private tab. There is a series of operations that allows making contributions from one side to the other. They are organized in two groups: publications from private to shared side and publications from shared to private side. Contributions from private side to the shared are always augmentative. The system makes incompatibility checking each time a publication is performed, and it is completed only if the publication is augmentative. Whatever is the checking result, Co-Protégé informs this result on the bottom the shared-private workspace tab. Besides, at the shared-private tab it is also possible to open conflicts (see conflict tab).

User tab. This is the tab to manipulate the user profile, to manage user interest. Users' interest can point to any kind of frame describe 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 in charge of changing user profile, by tracking user activity. Then, this may be used to adapt delivered awareness information. Co-Protégé provides a set of suggestions that helps users to complete their profile [Baldo03].

Conflict tab. A conflict is created in the shared-private tab by selecting the set of frames that will be put in conflict. After that, the frame are shown with "in conflict" icon. To facilitate the visualization of conflict, it was decided to separate the conflict management form the shared-private tab. The conflict tab defines a space where users can browse and develop a conflict. 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. It is the mechanism that enables to publish contributions that did not pass the compatibility checking.

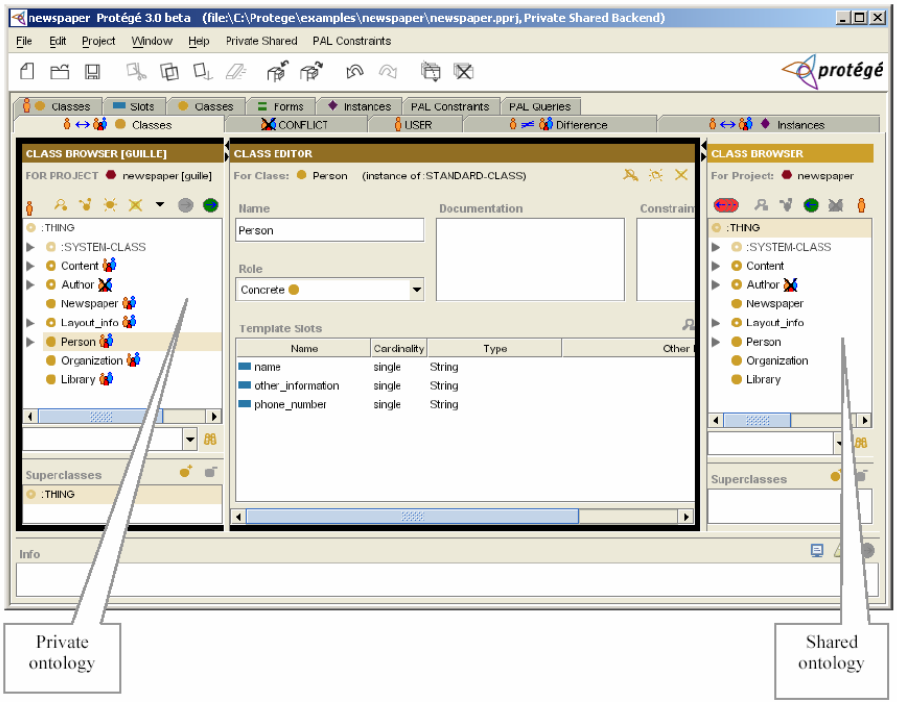

Fig. 1. A snapshot of Co-Protégé. Both private and shared ontologies can be appreciated simultaneously. The black rectangle remarks the associated property pane to the current ontology.

4 Conclusions

Co-Protégé extends Protégé through the definition of some plugins. The resulting architecture looks like any other Protégé extension because it follows the Protégé extension philosophy. In Co-Protégé a project is made up of the shared ontology plus every private ontology (one for each user). Both kinds of ontologies are stored in Standard Text File format. Co-Protégé is a client-server application, where a project is defined as a Protégé's metaproject. In this metaproject are defined every ontologies (the shared and each private) and the access permissions.

Co-Protégé uses the Protégé-knowledge model: classes, slots, facets and instances. However, it differs at the Protégé-2000 metamodel level. Co-Protégé uses two different metamodels to model both the private and the shared ontologies. Any private ontology is considered as a Protégé-2000 project. However Co-Protégé proposes its own metaclass architecture, which is an extension of the Protégé-2000, to model the primitive frames of a shared ontology (shared-class, shared-slot, etc.) and conflict primitives. Besides, Co-Protégé defines a set of generic ontologies to model the knowledge about the community and the collaborative activity.

The Co-Protégé prototype was developed only to manage the conceptual design of the shared ontology, still remains the treatment of instances.

References

[Baldo03] Baldo G., Lemus M., Diaz A. 2003. Acciones como Fuente del Conocimiento de una Comunidad. Clei'03, La Paz, Bolivia, 29 Septiembre - 3 Octubre.

[Diaz04] Diaz, A., Canals G. Divergence Occurrences in Knowledge Sharing Communities. LNCS, Springer Verlag -Proceedings of Criwg'04 17-24

[Diaz05] Diaz, A. 2005. Divergence Occurrences in Knowledge Sharing Communities. PhD Thesis, to be presented during 2005.

[Gennari02] Gennari J., M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, S. W. Tu. 2003. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies archive Volume 58, Issue 1 89-123 ISSN:1071-5819

[Klein01] Klein, M. 2001. Combining and relating ontologies: an analysis of problems and solutions. In Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., and Uschold, M., editors, Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA.