# OFLOSSC, AN ONTOLOGY FOR SUPPORTING OPEN SOURCE DEVELOPMENT COMMUNITIES

Isabelle Mirbel
*INRIA Sophia Antipolis, 2004 route des lucioles - BP 93, FR-06902 Sophia Antipolis, Cedex France*
*Laboratoire I3S, Route des Lucioles, BP 121, FR-06903 Sophia Antipolis, Cedex France*

Keywords: Ontology, open-source development, Communities of practice.

Abstract: Open source development is a particular case of distributed software development having a volatile project structure, without clearly-defined organization, where activity coordination is mostly based on the use of Web technologies. The dynamic and free nature of this kind of project raises new challenges about knowledge sharing. In this context, we propose a semantic Web approach to enhance coordination and knowledge sharing inside this kind of community.
The purpose of this paper is to present OFLOSSC, the ontology we propose as the backbone of our approach. It is dedicated to the annotation of the community members and resources to support knowledge management services. While building OFLOSSC, our aim was twofold. On one hand, we wanted to reuse the ontologies on open source provided in the literature. On the other hand, we adopted a community of practice point of view to acquire the pertinent concepts for annotating resources of the open source development community. This standpoint emphasizes the sharing dimensions in knowledge management services.

## 1 INTRODUCTION

According to (Hesse, 2005) and (Happel and Seedorf, 2006), applications of ontologies in software engineering are manifold. They cover all phases of the software development process, from requirement elicitation to software maintenance through implementation and deployment steps. As far as we understood them, applications of ontologies in software engineering aim at supporting specific technical tasks of the software development process. Indeed, existing applications of ontologies in software engineering focus on a better exploitation of explicit knowledge during the development process.

However, according to (Ntioudis and al., 2006), the main kind of knowledge exchanged by actors all along the software development process is tacit and based on direct communication between colleagues. Exchanges are possible when developments are made by small team but become difficult when the team size increases or when the team members are geographically dispersed. In these contexts, specific supports (as for instance wikis or instant messaging) are used to share and exchange non structured knowledge. This is particularly true in open source development contexts, where team members work in geographically distinct places, rarely meet and coordinate their activities mostly by using Web technologies.

In the context of open source development communities, the objective of our work is to enhance coordination and knowledge sharing through the development of dedicated knowledge management services. Knowledge management services aim at offering efficient and effective management of the community knowledge resources, so as to improve access, sharing and reuse of this knowledge, which can be tacit or explicit, individual or collective. A knowledge resource can be a document (bug report, post in a forum, user manual, etc.) materializing knowledge acquired and shared through cooperation between the community members, a service useful to the community members or a person holding tacit knowledge. To achieve efficient coordination and knowledge sharing through the development of knowledge management services, we rely on an ontology and on semantic annotations of the community knowledge resources with regard to this ontology. Such semantic annotations (for example, on the profile, role and competencies of a community member or on the semantic content of a document) can then be used by knowledge management services such as knowledge search services, knowledge visualization services and therefore support the coordination and sharing processes in the open source development process.

The purpose of this paper is to describe OFLOSSC (Ontology about Free/Libre Open source Software Communities), the ontology we propose as the backbone of our approach. In the next section, we discus the way we consider open source development community. Then, the following section describes the content of OFLOSSC and our conclusions.

## 2 OPEN SOURCE DEVELOPMENT COMMUNITIES

A FLOSS (Free/Libre Open Source systems)[1] is a software or a computer language which license allows everybody to use, study, modify, duplicate, give and sell it. According to (Ankolekar, 2005), communities developing FLOSSs are usually created from a software or computer language developed by an individual or an organization which source code, called seed code, is then transferred to the open source domain. The activities of the individuals belonging to the community built around the FLOSS consist in maintaining and supporting evolution of the seed code. Linux, Mozilla, Apache, OpenOffice.org ou MySQL are well-known examples of FLOSS.

Members of FLOSS development communities work in geographically distinct places, rarely meet and coordinate their activities mostly by using Web technologies (mails, forums, discussion lists, collaborative work platform). According to (Ntioudis and al., 2006), FLOSS development may be seen as a particular case of distributed development having a volatile project structure, without clearly-defined organization and assigned tasks for all of its members, requiring a long term commitment and a common vision of the participants. The dynamic and free nature of this kind of project raises new challenges about knowledge sharing.

Proposals have already been made to exploit semantic Web techniques to improve knowledge sharing in FLOSS development communities. A. Ankolekar proposes a tool, Dhruv (Ankolekar, 2005), which exploits semantic Web models and techniques to support bug resolution in FLOSS development communities. G. Simmons (Simmons and Dillon, 2006) proposes an ontology to support the development of semantic portals dedicated to FLOSS development community. The proposed ontology, OSDO (Open source Development Ontology), mainly focuses on concepts describing tasks and tools dedicated to software development activity. It provides a detailed classification of the different kinds of tools and tasks encountered in FLOSS development communities, which let us think that it is devoted to large FLOSS development communities.

[1]http://en.wikipedia.org/wiki/FLOSS

From our point of view, FLOSS development communities may be considered as Communities of Practice (CoP). According to (Wenger et al., 2002), CoPs are "groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis". The members of a CoP cooperate and exchange knowledge to create a collective value useful to everyone. They share common resources (know-how, experiences, documents) and collaborate in a collective learning process.

Web technologies encourage the emergence of virtual CoP. The two main specificities of a virtual CoP are to exist outside of any particular organization and, because of this independence and the geographical dispersion of its members, to be based on Web technologies (Zarb, 2006). FLOSS development communities belong to this category of CoP.

In this context, we consider FLOSS development communities as virtual CoPs in order to emphasize the collective and collaborative learning dimensions of such communities, so as to support the enhancement of coordination and knowledge sharing.

To model the FLOSS development concepts from a CoP point of view, we started from O'CoP, an ontology dedicated to CoPs which has been developed inside the framework of the PALETTE european project[2]. The aim of O'CoP is to provide a full set of concepts to describe any CoP (its actors, their competencies, their resources, their activities, etc.) in order to allow the semantic annotation of the CoP resources with regard to this ontology. Indeed, three ontological levels are provided in O'CoP. The high level ontology provides models in order to build the other layers of the ontology. The middle layer provides concepts common to all CoPs and the specific layer provides concepts specific to each CoP. The high level ontology and the middle layer(Tifous et al., 2007) have been the starting point of our work. Our proposal can be seen as part of the specific layer of the O'CoP ontology.

We also started our work from the OSDO ontology (Simmons and Dillon, 2006) and from the ontologies provided in Druhv (Ankolekar, 2005). From Dhruv, we reuse the vocabulary suggested to describe bug related resources (bug reports, discussions, posts, etc.) and code related resources (files, packages, variables, etc.). We reuse few interaction classes. In-

[2]http://palette.ercim.org

deed, to model interactions, we prefer to rely on a broader ontology proposed in the SIOC project [3] aiming at providing methods for interconnecting discussion means such as blogs, forums and mailing lists to each other. As we will discuss it in details in section 3.3, we specialized the vocabulary provided in the SIOC project with concepts from the Dhruv interaction ontology describing bug-related messages. We do not reuse Dhruv community vocabulary. Indeed, this vocabulary focuses on bugs only and we want to provide a broader vocabulary. From the OSDO ontology, we reuse a large set of classes about roles, activities and tools of the community.

Indeed, the ontologies provided in Dhruv are dedicated to FLOSS development community members in charge of bug tracking and solving. On the other side, the OSDO ontology provides concepts dedicated to FLOSS development process management, therefore describing in details roles, activities and available tools. There is no intersection between the two approaches and we propose to conciliate the OSDO and Dhruv ontologies by looking at FLOSS development communities as CoPs. In OFLOSSC, we put our efforts on providing a vocabulary to describe aspects not covered neither by OSDO nor by Dhruv (mainly implicit roles, implicit activities and decision-making), by specializing the vocabulary provided in the O'COP middle layer. We also provide relationships to link concepts from the different ontologies together.

From the methodological point of view, we adopted a top-down approach to build OFLOSSC. We started from the main concepts provided in O'CoP and we specialize them to meet the specificities of the semantic annotations suitable for FLOSS development community resources. We also respected the principles suggested by (Bachimont, 2000) and (Kassel et al., 2000) and tried to reuse existing ontologies about FLOSS development. OFLOSSC is formalized in OWL DL. Because of space limitation, we will not present its whole content in this paper. We focus on the main concepts of actors and resources.

## 3 OFLOSSC

In this section we first give an overview of OFLOSSC. Then we focus on concepts proposed to describe actors and resources.

### 3.1 Overview of OFLOSSC

A CoP is defined through the activities performed by its members as well as its practices and resources which are exchanged and shared by the community members. Resources and actors will be described in details in the next sections. Figure 1 summarizes the main concepts taken from O'CoP and describing an open source development community.

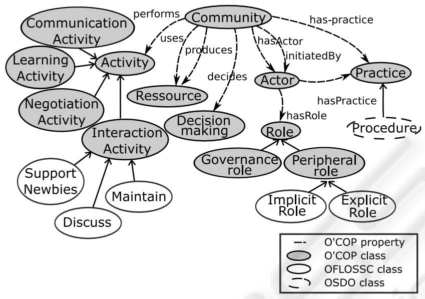

Figure 1: Overview of OFLOSSC.

Concerning the activities, we rely on the generic models proposed in the O'CoP high level ontology, where four kinds of activities are distinguished: (i) communication activities dedicated to information transmission, (ii) interaction activities dedicated to knowledge sharing and exchange, collaborative design, co-production, (iii) negotiation activities dedicated to interactions to agree on ideas or make consensus and (iv) learning activities dedicated to new knowledge acquisition. Among the interactions activities, we distinguish support to newbies, maintenance and discussion activities, which are the main activities of FLOSS development communities members (Barcellini, 2005). Therefore we specialize the class *InteractionActivity* from O'CoP into *SupportNewbies*, *Maintain* and *Discuss*.

With regard to practices, we specialize the O'COP generic class *Practice* following the OSDO ontology through the class *Procedure*, which represents any established and well defined behaviour for the accomplishment on some activity (Simmons and Dillon, 2006).

Concerning the role notion and following the O'CoP middle layer, we distinguish between governance role (*GovernanceRole*), aiming at supporting community members through their interactions and knowledge sharing, and peripheral roles (*PeripheralRole*), played by actors building and exploiting the knowledge of the community. Facilitators or coordinators are examples of governance roles. Peripheral roles doesn't mean secondary role. These roles cover most of the roles played by the members of the community. They are strongly related to the activity domain and therefore they are not further detailed in the O'CoP middle layer. We refine the class *PeripheralRole* into roles dedicated to FLOSS development

[3]http://sioc-project.org

community. We distinguish implicit roles (*Implicit-Role*) from explicit ones (*ExplicitRole*). Explicit roles are roles assigned to members of the community. Examples of explicit peripheral roles in FLOSS development community are developers, administrators or projet manager. Implicit roles reflect the implication of actors in the community life, as for instance initiation or participation to discussion threads.

By modeling explicit roles, our aim is to support materialization of knowledge held by community members and in particular their profile. As we are also concerned with tacit knowledge, we also provide concepts to materialize implicit roles in the community. By looking at the authors of posts in discussion threads, it is possible to understand when community members play the role of discussion initiator or discussion animator for instance. This knowledge may be useful to better understand who is doing what in the community. It contributes to support the collective and collaborative dimensions of FLOSS development communities.

We will now discuss in details actor and resource related concepts.

## 3.2 FLOSS Actors

In the O'CoP middle layer, different actors (*Actor*) are distinguished: community members (*Member*), of course, but also individuals participating in some of the community activities (*Individual*) or contributing to the community life (*Contributor*) without being members. Actors may also be legal entities (*LegalEntity*) behaving as project partners. Among legal entities O'CoP distinguishes between professional organizations (*ProfessionalOrganization*), partners (*Partner*) and institutions (*Institution*). The class *Member* is also specialized in order to distinguish former members (*FormerMember*) from current members (*CurrentMember*). This last class is again specialized into the class *NewMember*.

In addition to concepts selected in the O'CoP middle layer, we introduce, among actors of FLOSS development communities, three specific actors presented in the following.

- Newbies (Barcellini, 2005) are new users of the FLOSS provided by the community. *Newby* is introduced as subclass of *NewMember* because it is distinguished from *NewMember* (generic to all CoPs) by the fact that, in FLOSS development communities, new members can only play the role of FLOSS users.
- The individual at the root of the FLOSS development (or his successor) is a particular member of the community, not considered as former or current member. His surnames are "benevolent dictator for life" (Barcellini, 2005) or "visionary" (Rahtz, 2005). We introduced *Visionary* as subclass of *Member*. We choose to represent *Visionary* as a subclass of *Member* because the visionary is the only one to play the role of project manager.
- Some partners only contribute to the FLOSS development community as distributor (*Distributor*), supporting the FLOSS dissemination. We introduced *Distributor* as a subclass of *Partner* because in FLOSS development communities, distributors are distinguished by the fact that they only play the role of FLOSS disseminator.

By refining the O'COP middle layer vocabulary with concepts describing FLOSS development community actors specificities, our aim is to support the improvement of knowledge sharing concerning who is who and who does what in the community, especially for newbies.

Figure 2 summarizes the vocabulary dedicated to FLOSS actor specification. The relationships between actors and roles are formalized through the *has-role* property whose domain is *Actor* and whose range is *Role*. For subclasses of *Actor*, the range of *has-role* is restricted to sublasses of *Role*. Classes, properties and restrictions are summarized in figure 3.

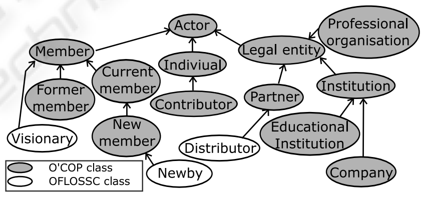

Figure 2: OFLOSSC Actors.

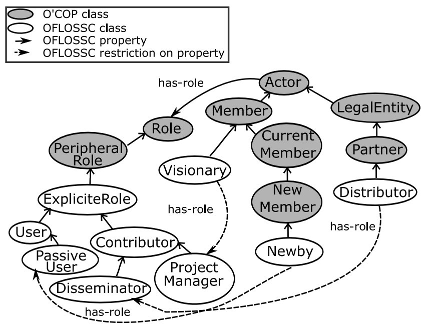

Figure 3: Roles on Actors in OFLOSSC.

## 3.3 Resource Related Concepts

Concerning the resources of the community, we started from concepts provided by the O'CoP middle layer in which tools are distinguished from interactions and documents (the class *Resource* is specialized into *Tool*, *Document* and *InteractionResource*). And we add to the classification proposed in O'CoP a fourth kind of resource dedicated to the knowledge of the community we are dealing with, that is the code of the FLOSS. We rely on Dhruv ontology for code related artifacts. The connection is supported by *SoftwareObject* which is a subclass of *Resource*.

Regarding tools, we reuse the classification proposed in OSDO. The OSDO ontology distinguishes configuration management systems, content management systems, defect management systems, asynchronous and synchronous communication tools, backup systems and test framework. We introduce these concepts as subclasses of *Tool*.

For documents, we also partially reuse the OSDO ontology. We distinguish between help documents (FAQs, how-tos and tutorials) and release documents (administrator manuals, API documentations, defect lists, developer manuals, release notes and user manuals). Therefore the class *HelpDocument* is specialized into *FAQ*, *Howto* and *Tutorial*; and the class *ReleaseDocument* is specialized into *AdministratorManual*, *APIDocumentation*, *DefectList*, *DeveloperManual*, *ReleaseNote* and *UserManual*.

With regard to interaction resources, that is to say resources dedicated to knowledge sharing and knowledge exchange, as blog or forum posts for instance, we rely on the ontology provided in the SIOC project. More precisely, we reuse classes *Post* and *Item* supporting annotation of blogs, forums and mailing lists. *Post* is a specialization of *Item* and *Item* is connected to our ontology as subclass of *InteractionResource*.

We also reuse the distinction between messages initiating discussion threads (*OpenMessage*) and messages animating the discussion (*CommentMessage*) from Dhruv ontologies. These concepts are also connected to our ontology as subclasses of *Post*.

The relationships between roles and messages are formalized through the *write* property whose domain is *ImplicitRole* and whose range is *Post*. For subclasses of *ImplicitRole*, the range of *Post* is restricted to sublasses of *Post*. Figure 4 summarizes the different classes, properties and restrictions required to model posts, together with their associated roles.

Among the discussion activities (introduced in section 3.1), we distinguish between discussions about changes, bugs and improvements. Therefore, the class *Discuss* is specialized into *DiscussChange*, *DiscussBug* and *DiscussSolution*. We also specialize the class *Post* with regard to the kinds of discussion specific to FLOSS development communities. This specialization matches the one we provide to model the interaction activities of the community. The class *Post* is specialized into posts dedicated to FLOSS evolution (*ChangeMessage*), posts dedicated to bug resolution (*BugMessage*) and posts dedicated to code improvement (*SolutionMessage*).

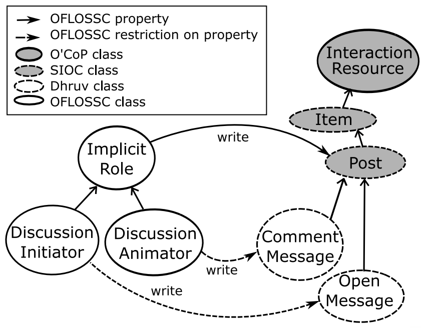

Figure 4: OFLOSSC Resources and Associated Roles.

We also reuse the Dhruv class *Commit* to refer to messages about commits which can be present in any kind of post (solution, bug or change). *Commit* is therefore connected to OFLOSSC as a direct specialization of *Post*.

The relationships between discussion activities and posts are formalized through the *dealWith* property whose domain is *Discuss* and whose range is *Post*. For subclasses of *Discuss* (*DiscussSolution*, *DiscussChange* and *DiscussBug*), the range of *Post* is restricted to sublasses of *Post* (respectively *SolutionMessage*, *ChangeMessage* and *BugMessage*). Figure 5 summarizes the different classes, properties and restrictions required to model discussion-related posts.

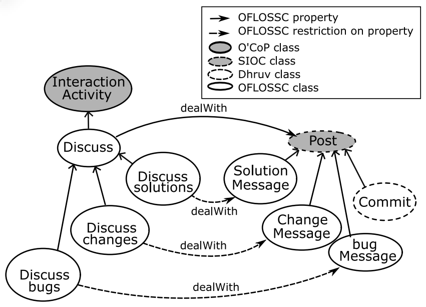

Figure 5: OFLOSSC Discussion Related Resources.

In OFLOSSC, we currently propose 46 core classes and 8 core properties. These artifacts focus on CoP related aspects (implicit roles, implicit activities and decision making). We also provide binding with Dhruv, OSDO, SIOC and O'COP ontologies. 10 classes participate to bindings with Dhruv, 38 with OSDO and 46 with O'COP. 2 properties have been defined with Dhruv classes and 4 with O'COP classes. OFLOSSC current core concepts as well as bindings with O'COP, Dhruv, OSDO and SIOC ontologies are available at http://ns.inria.fr/oflossc/.

## 4 CONCLUSIONS

Our work aims at facilitating the exchange and sharing of tacit and explicit knowledge inside FLOSS development communities. This particular case of distributed software development having a volatile project structure, without clearly-defined organization and assigned tasks for all of its members, requiring a long term commitment as well as a common vision of the participants, where activity coordination is mostly based on the use of Web technologies, raises new challenges in terms of knowledge management.

We choose a semantic Web approach relying on OFLOSSC which allows the annotation of the community resources in order to enhance the exploitation of these resources through dedicated knowledge management services.

This paper focused on OFLOSSC which is the backbone of our approach. While building this ontology, our aim was twofold. On one hand, we reused the ontologies about FLOSS provided in the literature (Dhruv and OSDO) as well as the ontology provided in the SIOC project. On the other hand, we tried to represent pertinent concepts of FLOSS development community resources from the community of practice point of view and we therefore started from the O'CoP generic ontology provided inside the framework of the PALETTE European project.

We plan to work on annotation mechanisms. We will start from SEMFAQ (Makni et al., 2008), which aim is to build a FAQ by extracting knowledge from emails in an automated way. We intend to reuse this approach to annotate FLOSS development community messages (which look similar to emails) and to extract tacit knowledge about implicit roles of community members from email headers.

We also plan to enlarge the scope of OFLOSSC to also support the annotation of resources not belonging to the community (external resources), in order to allow FLOSS development community members to take advantage of resources and services provided over the Web in an integrated manner.

## ACKNOWLEDGEMENTS

The author would like to thanks Catherine Faron-Zucker and Fabien Gandon for discussions and comments on this work.

## REFERENCES

Ankolekar, A. (2005). *Towards a semantic web of community, content and interactions*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Bachimont, B. (2000). *Ingnierie des connaissances, volutions rcentes et nouveaux dfis*, chapter Engagement smantique et engagement ontologique : conception et ralisation d'ontologies en Ingnierie des connaissances. Eyrolles.

Barcellini, F. (2005). Les discussions en ligne en conception de logiciels libres: analyse des traces d'un processus asynchrone de conception distance. Master's thesis, Conservatoire National des Arts et Mtiers.

Happel, H. and Seedorf, S. (2006). Applications of ontologies in software engineering. In *International Workshop on Semantic Web Enabled Software Engineering (SWESE'06)*, Athens, USA.

Hesse, W. (2005). Ontologies in the software engineering process. In *Workshop on Enterprise Application Integration*.

Kassel, G., Abel, M.-H., Barry, C., Boulitreau, P., Irastorza, C., and Perpette, S. (2000). Construction et exploitation d'une ontologie pour la gestion des connaissances d'une quipe de recherche. In *Journes francophones d'Ingnierie des connaissances*.

Makni, B., Khelif, K., Cherfi, H., and R., D.-K. (2008). Utilisation du web smantique pour la gestion d'une liste de diffusion d'une cop. In *8ime journe francophone Extraction et Gestion des Connaissances*.

Ntioudis, S. and al. (2006). Report describing state-of-the-art km in software engineering. Technical Report D1, TEAM IST Project 35111.

Rahtz, S. (2005). What is an open source software community? http://www.oss-watch.ac.uk/events/2005-07-04/index.pdf.

Simmons, G. and Dillon, T. (2006). Towards an ontology for open source software development. In *IFIP Working Group 2.13 Foundation on Open Source Software*, pages 65–75.

Tifous, A., El Ghali, A., Dieng-Kuntz, R., Giboin, A., Evangelou, C., and Vidou, G. (2007). An ontology for supporting communities of practice. In *International Conference On Knowledge Capture, K-CAP'07*, pages 39–46.

Wenger, E., McDermott, R., and Snyder, W. (2002). *Cultivating communities of practice*. Harvard Business School Press.

Zarb, M. (2006). Modelling participation in virtual communities of practice. Master's thesis, Information Systems Department at the London School of Economics, London, UK.