LibreSource: Web Based platform for Supporting Collaborative Activities

Hala Skaf-Molli, Pascal Molli, Olivera Marjanovic, Claude Godart

LibreSource : Web Based Platform for Supporting Collaborative Activities

Hala Skaf-Molli[1] , Pascal Molli[1], Olivera Marjanovic[2], and Claude Godart[1]

[1]*LORIA-INRIA*
615, Rue du Jardin Botanique
Villers-lès-Nancy, 54600, FRANCE
{skaf, molli, godart}@loria.fr
[2]*University of Sydney,*
Sydney, NSW 2006, Australia
{o.marjanovici}@econ.usyd..edu.au

Abstract

The existing Collaborative Distributed Environments (CDE) have several important drawbacks. Consequently, users need to rely on very few of installation like SourceForge and accept all the associated risks. The main objectives of this paper are to describe collaborative technology called LibreSource and to illustrate how this technology can be used to support a collaborative software design process. LibreSource[1] is an open source software platform that provides a framework and the associated services designed to support different kind collaborative processes, in particular design projects.

1. Introduction

In recent years, many different Internet-based collaborative platforms have emerged. They can be classified into two large categories:

- **Community oriented platforms** that are mostly represented by content management systems like Zope/Plone[1] and PhpNuke[2]. The goal of these platforms is to manage information flow and communication among a large group of people.

- **Production oriented platforms** are best represented by software development environments. Sourceforge[3] and its descendants Gforge[4], Savannah are good representatives of these kind of platforms.

All these environments provide more or less the same kind of functionalities or services. Community oriented platforms provide communication services (i.e.forum, mail, notification) as well as content management services (i.e. publication, wiki, file management). Production oriented platforms provide file sharing services (configuration management, workspace management), communication services and coordination services (task management, trackers).

However, it is important to point out that the existing collaborative environments have reached their limits when dealing with large-scale collaboration structures, especially where several media are used at the same time by a community (or communities) of people. CSCW (Computer Supported Cooperative Work) and groupware platforms are supposed to support this type of collaboration but they suffer from complexity, high costs and rigidity (i.e. they do not adapt easily to different types of environments). Some companies propose commercial products that group several tools together to support such complex structures. Examples include BSCW [5, 6], SourceForge[3] and Lotus Notes [7]. However, they do not cover all facets of collaboration. In addition, they are proprietary systems, not so easy to deploy, and require solid programming skills. Recently, there have been some efforts to provide flexible and open collaboration platforms e.g. ZOPE.

Compared to the existing environments in the same category (such as G-Forge [4] and Savannah [8]), LibreSource offers a high level of integration.

This paper introduces LibreSource that has been specifically designed to support large-scale collaboration structures that are customisable to a wide range of needs and easy to use by non-specialist users.

The paper is structured as follows. The next section will introduce some organizational models for complex distributed software development teams . Section 3 will give a brief overview of LibreSource. Section 4 will present an example of a distributed software development process supported by a network of LibreSource servers. Section 5 will focus on technical implementation of that process. The last section offers the main conclusions and describes our future work in this area.

2. Supporting complex distributed structures

We see collaboration as a group activity of a large number of participants (i.e. a community), designed to achieve a particular purpose or goal. A *collaboration structure* refers to an IT-enabled solution that supports collaboration. Furthermore, a *complex* collaboration

[1] Libresource (www.libresource.org) is RNTL funded project. It is developed on the ObjectWeb Jonas application server.

structure consists of a combination of several tools for collaboration and communication being used simultaneously. In addition, we assume that the community of users is geographically distributed i.e. not co-located at the same site. For example, the main purpose of the open-source community is to develop free software. This community is distributed all over the world, and is hosted by SourceForge.

One major problem with this centralised management solution is service availability. Thus, service provision is not guaranteed. Actually, if SourceForge closes (or becomes unavailable), the community looses access to 80000 projects. This indeed happened to smaller-scale systems after corruption of their sites by hackers. Obviously, centralized management of user communities and their projects is a major drawback.

LibreSource aims to provide an alternative to this solution in order to avoid the same problem. Thus, instead of installing one single, complex community and project support site, we propose to support federation and pooling. If one node becomes unavailable for any reason, resources hosted on this site can be moved to other sites easily. With this vision, instead of building a complex site with 800000 users, we propose to aggregate a hundred of nodes of thousands of users. The infrastructure needed is much more simple and the visibility of the nodes is maintained.

2. 1 Community organization

In order to build a federation of LibreSource servers, it is necessary to organize the user community in some way. For this purpose, it is possible to use the organization models proposed in [9]. Here, the authors propose four models to coordinate the work of geographically distributed Research and Development (R&D) teams: functional area of expertise model, product structure model, process step model and finally customization model.

In this paper, we illustrate how to implement the first model by using the LibreSource approach.

2.2 Functional area of expertise :

With this organizational model, expertise for a specific area involved in development of the product is located at a single site. Let us consider development of a very large software product by an organization or a community. This organization co-locate people according to the functional areas of expertise [9]. Thus, expertise for a specific area involved in development of the product is located at a single site. Therefore, a collaborative software design process consists of development, test and release activities that correspond to these three area of expertise.

If the organization uses a software like SourceForge to host the development project, all the data produced by the different activities: development, test and release will be stored at the same site. If this site becomes unavailable, the work at the different expertise will stop. However, if the organization uses LibreSource to host the development activities, it is possible to install separate server for each area of expertise. Thereforethe development team will have its own private server that can be installed locally. The same applies totest and release teams. This approach results in a network of LibreSourceservers that enables data propagation from one server to another. This architecture offers two main advantages. On one hand, there is not one single server that could be a bottleneck. On the other hand, several servers allow to large-scale projects with more people.

In the reminder of the paper, we describe the LibreSource architecture and how it can be used it to implement the above collaborative scenario.

3. The LibreSource approach

For the purposes of better understanding, it is possible to compare the behaviour of a LibreSource server with an ordinary file system. Thus, a LibreSource server is a tree of instantiated components. A component can be a forum, a project or a bug tracker. As in the file system, each component of the LibreSource Tree declares its own security policy. In the LibreSource tree, users can create a component, rename a component, edit it, delete it and move it. Furthermore, LibreSource has an open architecture. Community members can develop their own components and plug them in the LibreSource framework.

To support cooperation, LibreSource provides several services such as awareness and data sharing. Awareness allows people involved in a project to be informed about the modifications on project files. The awareness service is implemented by events mechanisms. Sharing data is a crucial for cooperative work, in LibreSource data can be propagated from one server to another by using a *synchronization component* called *So6*. More precisely, *So6* is a generic file synchronizer as described by [10].

3.1 Events

Unlike file systems, each component can generate events that could be used for the awareness and triggering purposes. Creating a component, moving a component, interacting with a component generates persistent event. This allows a better awareness of the users who work together in the platform. For example, if a user updates the content of a file, people could be informed of this action with an email generated by the event. This awareness feature is very important to enable collaborative work in a distributed environment. LibreSource event contains a lot of interesting information such as: the action type (create, update, delete, ...), the associated component , the user who has executed the action, the date of the action, and a list of arguments (allows additional information in the event).

Furthermore, there are several generic event types: creation, update and deletion of a component. Users can subscribe to events generated by a subtree of components. He/she will be then notified by an email and/or jabber instant messaging when an event is generated on a resource of this subtree.

3.2 Data Sharing

In LibreSource data can be propagated from one server to another by using a *synchronization component* called *So6*. More precisely, *So6* is a generic file synchronizer that can be classified as a configuration management tool comparable to CVS [11]. It allows synchronization with more than one repository. This is a very important feature that allows implementation of synchronization networks. In turn, these networks allow representation of dataflow processes.

So6 aims to improve the merging process by unifying it. The same algorithm is used to merge all types of data: text files, XML files. Consequently, , So6 can be considered as a unified framework for building robust merge tools and verification of their safety. Anyone can extends So6 to merge other kind of data by writing a new set of transformation functions[10]. So6 also allows building of synchronization networks. It means that a workspace can be synchronized with more than one message queue. This is an important feature of LibreSource.

4. Collaborative Software Development Process

This section illustrates how we can support the scenario described in Section 2. The main objective is to support the collaborative software development process in an organization that uses the functional area of expertise model. The development process consists of three areas of expertise, namely development, test and release. The implementation of this organizational model by LibreSource results in the schema depicted by Figure 1.

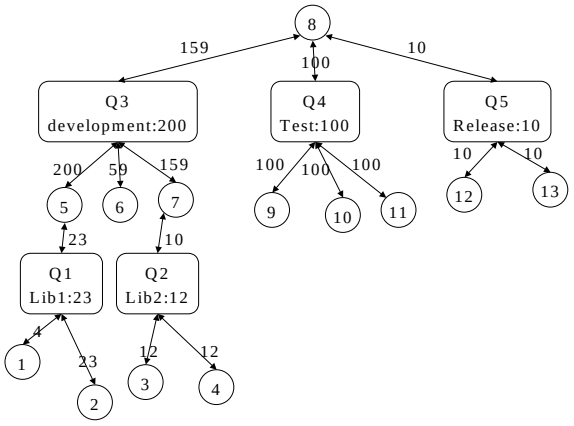

Figure 1: Software development process in LibreSource

This graphical notation uses two different LibreSource components: *Queues* (depicted by oblongs) and *Workspaces* (depicted as circles). Each Queue contains *n* operations (e.g. queue *Q3* contains *200* operations). Workspaces generate changes (called operations) and *commit* them to queues. They also *update* changes from the associated queues. A number of *commit/update* operations between a workspace and a particular queue are depicted as a label of the corresponding double arrow.

Note that a workspace can be connected to several queues. For example, *workspace 8* is connected to queues *Q3*, *Q4* and *Q5*. It has been updated by the first *159* changes coming from *Q3* and is up-to-date with *Q4* and *Q5*. *Workspaces 5*, *6* and *7* are also connected to Queue *Q3*. At the same time workspaces *5* and *7* are updated with the changes coming from *Q1* and *Q2*.

A workspace can *commit* operations to a queue only if it is up-to-date with this queue. This solution prevents the problem of lost updates. On the other hand, when a workspace is updated with changes from the associated queue, all incoming operations and local operations are merged using the operational transformation algorithm [10]. Furthermore, So6 ensures data convergence i.e. when all workspaces are up-to-date with all changes they will contain the same data. Therefore, when using *So6*, users of the workspaces commit the stream of changes that will be propagated to different queues.

For example, in a software development process, depicted by Figure 1, suppose that the user of *workspace 8* is responsible for data propagation between development, test and release. Therefore, *workspace 8* will be updated with the changes issued by the development queue and submit them to the test queue. If testers agree with the proposed solutions, they will commit operations to the Release queue. On the other hand, if testers report any bug, changes representing code fragments will be committed back to the developer queue. The above scenario, requires three LibreSource servers. Queues Q3, Q1 and Q2 are hosted on the same server, Q4 and Q5 are hosted on different servers. So, developers can have their own LibreSource server, testers another one, while the release queue can be hosted on a special server that can support the heavy load. Thus, each functional area of expertise can have its own LibreSource server and configure it to suit their own needs.

4. 5. Collaborative software development in action

This section illustrates how the network of the LibreSource servers depicted by Figure 1 will work. Suppose that the organization described in section 2 develops a software system called XYZ. Normally, this

system goes through several stages of development, beta testing and release activities. The following process is adopted by all participants to coordinate their efforts in the project: Suppose that there are 3 developers each one working on his/her own workspace developing the assigned components of XYZ software system (as depicted by Figure 2).

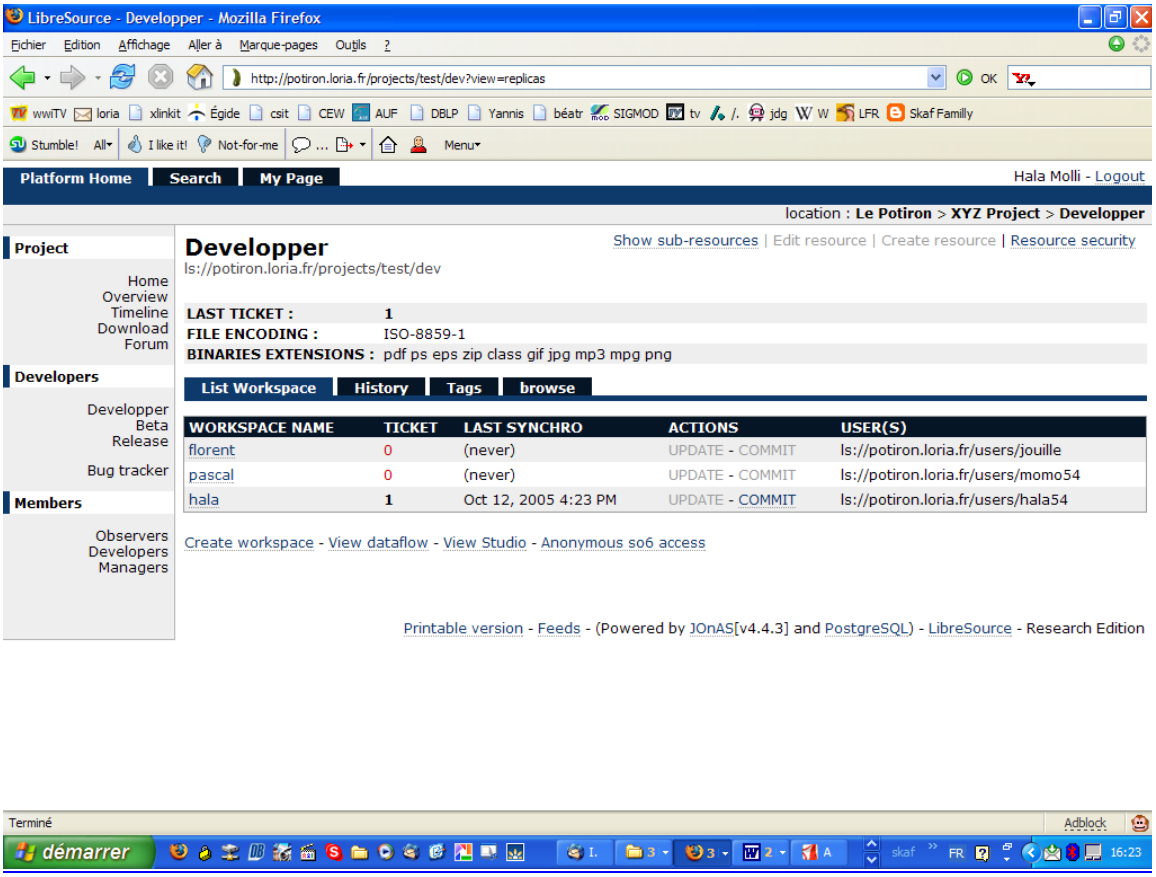

Figure 2: Developers' Workspace in LibreSource

- A queue Developer is used to propagate changes from one developer's version to another (Figure 3). It is hosted at the developer server.

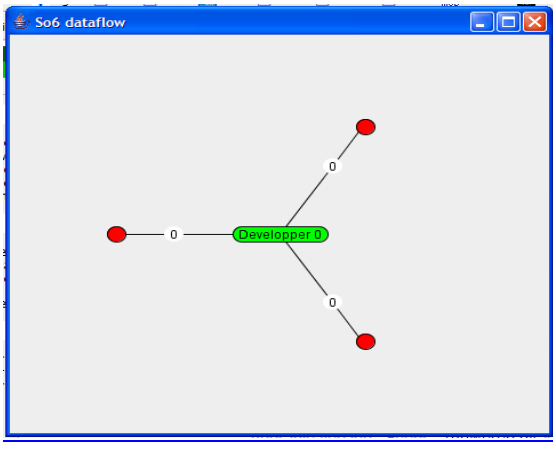

Figure 3: Propagation of changes from Developer queue

- When ready, the new version of XYZ is propagated for Beta testing (through operation *Commit*). We assume that this operation is restricted to few users only. Remember that the workspace 8 allows to connect the developer server and the tester server.
- Then, each user at a beta testing site can have access to the new XYZ's version using another queue resource *Beta* (through operation *Update*).
- As in the real-life software development scenario, there could be many iterations through which different beta versions are produced. Before the queue *Release* is used, only updates are propagated to the beta users.
- At some stage, the latest version of XYZ is released. They can obtain it by invoking operation *Update* on another queue resource called *Release*.

Developers communicate with each other using a bug tracker component. Changes and bugs are reported from users back to developers via the forum component.

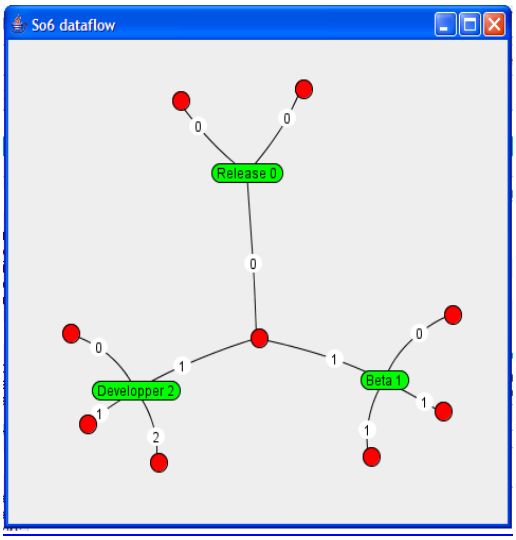

Figure 4: Synchronisation dataflow between Developer, Beta and Release queues

Figure 4 illustrates the synchronization dataflow between Developer, Beta and Release queues used by *So6*. The figure is generated from the real implementation of this example with LibreSource.

This software development process can be made even more complex by involving more participants in the project, setting different security policies, adding archiving mechanisms etc.

For example, each developer can manage more than one workspace if he/she is using a home computer, a work computer and a laptop for developments of XYZ. In this case, another queue resource and several workspaces are added to their original workspace. It is important to point out that participants are allowed to modify their resource tree to suit their needs (providing that they have the right permissions to do so). This is one of the features that make LibreSource suitable for large-scale collaboration projects.

6. Conclusion and future work

The paper describes a collaborative technology called LibreSource designed to support cooperative work in a complex collaborative structure. In this paper, we illustrate how it is possible to use LibreSource to support software development process in an organization that uses the 'functional area' model to coordinate geographically distributed development team. Similarly , it is possible to use LibreSource to support a software development process in organization that chose to adopt other organizational models described by[9].

LibreSource server can be accessed through a Web browser such as Internet Explorer or Mozilla at the address www.libresource.org. LibreSource can be easily customized to be use in different cooperative situations including software development and elearning [12].

As a future work, we will use LibreSource to build a LibreSource to LibreSource network such as P2P network and to support cross organizational work including collaborative business processes

7. References

[1] Zope/Plone [online] available from http://www.zope.org

[2] PhpNuke [online] available from http://phpnuke.org

[3] SourceForge [online] available from http://www.sf.net.

[4] G- Forge [online] available from http://gforge.org/

[5] BSCW [online] available from http://bscw.fit.fraunhofer.de/

[6] Bentley, R., Appelt, W., Busbach. U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J. and Woetzel, G., "Basic Support for Cooperative Work on the World Wide Web", in *International Journal of Human-Computer Studies 46(6)*: Special issue on Innovative Applications of the World Wide Web, p. 827-846, June 1997, ©Academic Press.

[7] Lotus [online] available from http://www-306.ibm.com/software/lotus/

[8] Savannah [online] available from http://savannah.gnu.org/

[9] Grinter, R.E., Herbsleb J.D. andPerry, D.E., "The geography of coordination: dealing with distance in R\&D work", *Proceedings of the International ACM SIGGROUP conference on Supporting Group Work: Group99*, Phoenix, Arizona, United States.

[10] Molli, P., Oster, G., Skaf-Molli, H. and Imine, A. (2003), "Using the Tnasformational Approach to Build a Safe and Generic Data Synchronizer", *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work: Group2003*, Sanibel Island, Florida, USA.

[11] B. Berliner. CVS II : Parallelizing software development. *Proceedings of USENIX*, 1990.

[12] Marjanovic, O., Skaf-Molli, H. , Molli, P. , Rabhi F. and Godart C. "Supporting Complex Collaborative Learning Activities: The LIBRESOURCE Approach". Proc. of the International Conference on Enterprise Information Systems, ICEIS2006, Paphos - Cyprus.

Acknowledgment

The authors would like to thank Olga Unzueta and Diana Escribano for their constructive feedback. Also many thanks to Florent Jouille for practical implementation of the software development example in the LibreSource environment..