Supporting Complex Collaborative Learning Activities: The LIBRESOURCE Approach

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

SUPPORTING COMPLEX COLLABORATIVE LEARNING ACTIVITIES – THE LIBRESOURCE APPROACH

Olivera Marjanovic[1], Hala Skaf-Molli[2] , Pascal Molli[2] , Fethi Rabhi[1] and Claude Godart[2]
[1]*University of New South Wales,*
Sydney, NSW 2052, Australia
{o.marjanovic, f.rabhi}@unsw.edu.au
[2]*LORIA-INRIA*
615, Rue du Jardin Botanique
Villers-lès-Nancy, 54600, FRANCE
{skaf, molli, godart}@loria.fr}

Keywords: Collaborative work, eLearning, Learning Designs, Collaborative software design.

Abstract: The main objective of this paper is to describe collaborative technology called LibreSource and how it is used to implement an innovative learning/teaching activity designed for software engineering students. From the educational perspective, this educational activity is based on the principles of problem-based learning and the latest Learning Design theory. The main objective of this activity to offer students a real-life experience in collaborative software development. Compared to the popular Learning Management Systems that only offer collaborative tools and support individual collaborative tasks, this technology enables design and implementation of complex collaborative processes.

1 INTRODUCTION

In order to respond to various powerful internal and external challenges, more and more universities embark on the same journey called by different names such as "e-learning", "web-based learning", "flexible learning", "on-line delivery" just to mention several popular terms. Many universities do it out of necessity to stay competitive, provide more flexibility, potentially reduce costs, provide alternative for mass lectures or out of interest or commitment to enhance student learning. Consequently, there is a great confusion in the educational literature and practice as to what this form of learning actually involves. Practical applications range from simply "putting lecture notes on the web" to very sophisticated web-based tools.

As a starting point for further discussion, this paper assumes that e-learning denotes learning experience in online environment where student's learning is enabled and supported by educational technology. This mode of learning provides any time/any place learning and as such, does not include face-to-face or on-campus learning activities. On the other hand, the blended (also called mixed or combined) mode of learning combines face-to-face teaching learning experience with online activities (e-learning). The concept of blended learning is very powerful as it enables the teachers to provide innovative learning activities to supplement what they are already doing in the classroom. Therefore, they can meaningfully extend the learning activities between face-to-face lectures and tutorials to re-enforce important points, give supplementary activities but also encourage learning community among students.

When carefully designed, blended learning can offer to students the best of both worlds, face-to-face and e-learning. However, this is not an easy task. Very often, eager to move from traditional "teacher-centered" learning, educators put too much emphasis on technology and as a result create technology-centered learning.

Consequently, technology is often used for delivery of the content (i.e. as delivery machines) rather than the tool that has a potential to enable new forms of learning never before possible. The main point is that technology itself does not result in learning or as (Ehramann, 1997) correctly observed the medium is not the message. Success or benefits of a certain technology can be largely attributed to the teaching methods used. At the same time, technology is not irrelevant. Any particular technology can be well or poorly suited to support the intended teaching and learning method. "There may indeed be a choice of technologies for carrying out a particular teaching task,

but it isn't necessarily a large choice. There are several tools that can be used to turn a screw, but most tools can't do it, and some that can are better for the job than the others."(Ehramann, 1997).

When designing activities for e-learning or blended learning, another equally important problem is learning methodology. There is the tendency to transfer traditional teaching and learning methods into new environments rather than invent the new ones. A large number of on-line learning environments are characterised by a narrowly defined educational model that emphasises delivery of materials and instructions rather than flexibility, intellectual engagement, participation or progress of individual learners.

The main objective of this paper is to describe an innovative teaching/learning activity designed for software engineering students and the actual collaborative technology used for its implementation. Although the main focus of this paper is on collaborative technology, to design it, we adopted the top-down approach and started from the appropriate learning methodology. Thus, from the educational perspective this activity is based on the principles of problem-based learning and is designed to offer students a real-life experience in collaborative software development. To enable and support this learning activity, we use the collaborative environment called LibreSource (see online reference). Compared to the currently available learning management systems (such as WebCT (see online reference) that only offer collaborative tools (e.g. forums and chat), this technology enables and supports collaborative processes.

The paper is structured as follows. The next section will introduce a motivating example of the intended teaching/learning activity and will place it within the appropriate framework of the related educational theories. Section 3 will give a brief overview of LibreSource and the remainder of the paper will focus on technical implementation of this educational activity in LibreSource.

2 MOTIVATING EXAMPLE

It has been widely recognised that one of the most important skills that students should acquire during their higher education today, is the ability to learn how to learn. To help students build these skills, it is necessary to engage them in carefully planned process-oriented learning activities rather than isolated learning tasks (Race, 1999). These activities should include a number of inter-related learning tasks that promote active learning through collaboration, critical thinking, problem solving and authentic interactions with the real-world problems.

Furthermore, by making students aware of the processes they are participating in, we are actually helping them to become more independent and self-regulated learners.

It is also important to recognise that the majority of students coming to universities these days are already computer literate. The so-called "Nintendo" generation is now in our classrooms. Obviously, they have different expectations about their learning/teaching experience and educational technologies used to support it. This is especially evident in the area of Information Systems/ Information technologies/ Computer Sciences education where students want to see their teachers practice "what they preach" both in the classroom and in on-line environment.

To design this particular learning activity we follow the approach proposed by the very recent theory of learning designs (Koper and Tattarsell, 2005). Design of this theory has been an international collaborative effort with the main objective to enable conceptual representation of learning/teaching scenarios, guided by pedagogy, so they could be supported by educational technology, but also shared among teachers. Therefore, design of a particular learning design (i.e. teaching/learning activity) starts from learning objectives and identification of all activities that different participants (students and their teacher(s)) need to do in order to achieve these objectives. Then for each activity, it is necessary to identify a set of learning resources as well as possible educational technology tools that could be used to support it. Adoption of the principles of learning design theory ensures that pedagogy guides all educational activities, rather than the available educational resources or technology.

Our example comes from the software engineering teaching discipline. From the educational perspective, the main objective of this learning design is to enable students to experience the process of collaborative software development. At the same time, it is also very interesting to observe that the underlying educational technology serves dual purpose: to support the intended learning activity but also to demonstrate an example of a complex technology and its use to software engineering students. So it could be used as an object of study for future learning activities.

Suppose that a software engineering teacher is interested in involving students in a software development project. This is a common learning activity, used in software engineering discipline, to give students hands-on experience and exposure to the real-life software development projects. It could be used as a part of formative or summative assessment. This teacher would also like to simulate a real-life software development experience in a team, so students are required to engage in collaborative software development activities.

This particular learning design involves the following activities. First of all, students are given a real-life software engineering problem and to solve it students are required to design and implement a software solution (system). Students are divided into teams and each team is

required to collaborate and come up with its own solution for the given problem. To facilitate their collaboration, within each team students assign different roles to different members.

Obviously, the main challenge of this project is design and development of a particular solution. In order to complete it, students need to engage in collaborative software development. Furthermore, after all teams have completed their projects, the teacher is interested in implementing the reflection phase. This requires each team to reflect upon their learning, rather than software development experience. Note that reflection-in-action is a very powerful learning activity, especially in the field of design. This particular activity is very useful for students to help them to become the so-called "reflective practitioners" (Schon, 1983; Schon, 1987) as required by their future profession.

This paper will focus on educational technology used to support the process of collaborative software development that is at the core of the previously described learning design. This process cannot be effectively supported by the existing educational technologies as they provide very simple tools (such as forums, chat tools, electronic whiteboards) to support individual tasks. Rather than simple tools, to support the intended learning activity, we need to provide a complex collaborative structure as described in the next section.

3 SUPPORTING COMPLEX COLLABORATIVE 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, we define a *complex* collaboration structure as consisting of a combination of several tools for collaboration and communication being used simultaneously.

The existing collaborative tools 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 these types 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 SourceForge, Lotus Notes (see corresponding online references). 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 at providing flexible and open collaboration platforms (e.g. ZOPE (see online reference).

This paper introduces LibreSource that has been specifically designed to support large-scale collaboration structures that are customizable to a wide range of needs and easy to use by non-specialist users. In this context, we use LibreSource as a platform designed to support collaborative software development process as required by the intended learning activity. Compared to the existing tools in the same category (such as G-Forge and Savannah (see online references), LibreSource offers a high level of integration.

To facilitate understanding, a LibreSource server behaves similarly to 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. However, unlike file systems, each component can generate events that could be used for the awareness and triggering purposes.

Data can be propagated from one server to another by using the *synchronization component* called *So6*. More precisely, *So6* is a generic file synchronizer (Molli et. al, 2003). It can be classified as a configuration management tool that 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* component can be easily applied to enable implementation of the classical software process that consists of development, test and release activities (as illustrated by Figure 1).

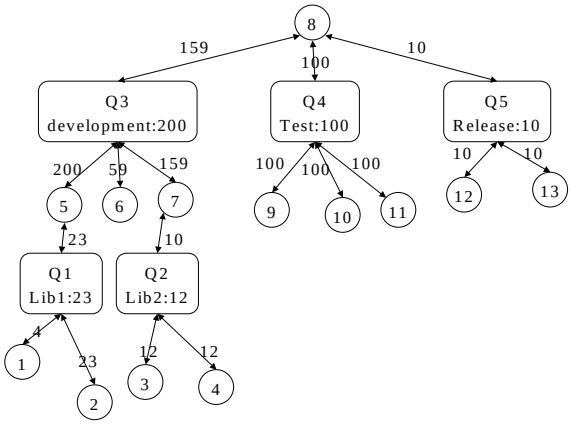

Figure 1: Software development process in LibreSource

The graphical notation uses two different LibreSource components: *Queues* (depicted by oblongs) and *Workspaces* (depicted as circles). Each Queue contains *n* operations i.e. 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 the 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 (Molli et.al., 2003). 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 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 bugs, changes representing code fragments will be committed back to the developer queue.

Each queue can be hosted on a different LibreSource server. 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 community can have its own LibreSource server and configure it for their own needs. Combined with forums, wiki and bugtrackers, LibreSource components can support complex collaboration structures.

4 IMPLEMENTATION OF COLLABORATIVE SOFTWARE DEVELOPMENT ACTIVITY

This section illustrates the implementation of a complex collaboration structure using LibreSource, based on the motivating example introduced in Section 2. We assume that each team of software engineering students is developing a software system called XYZ. Just like in a professional software development scenario, 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 working on his/her own workspace developing the assigned components of XYZ software system (as depicted by Figure 2).

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

- 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 (e.g. members of the same student team).

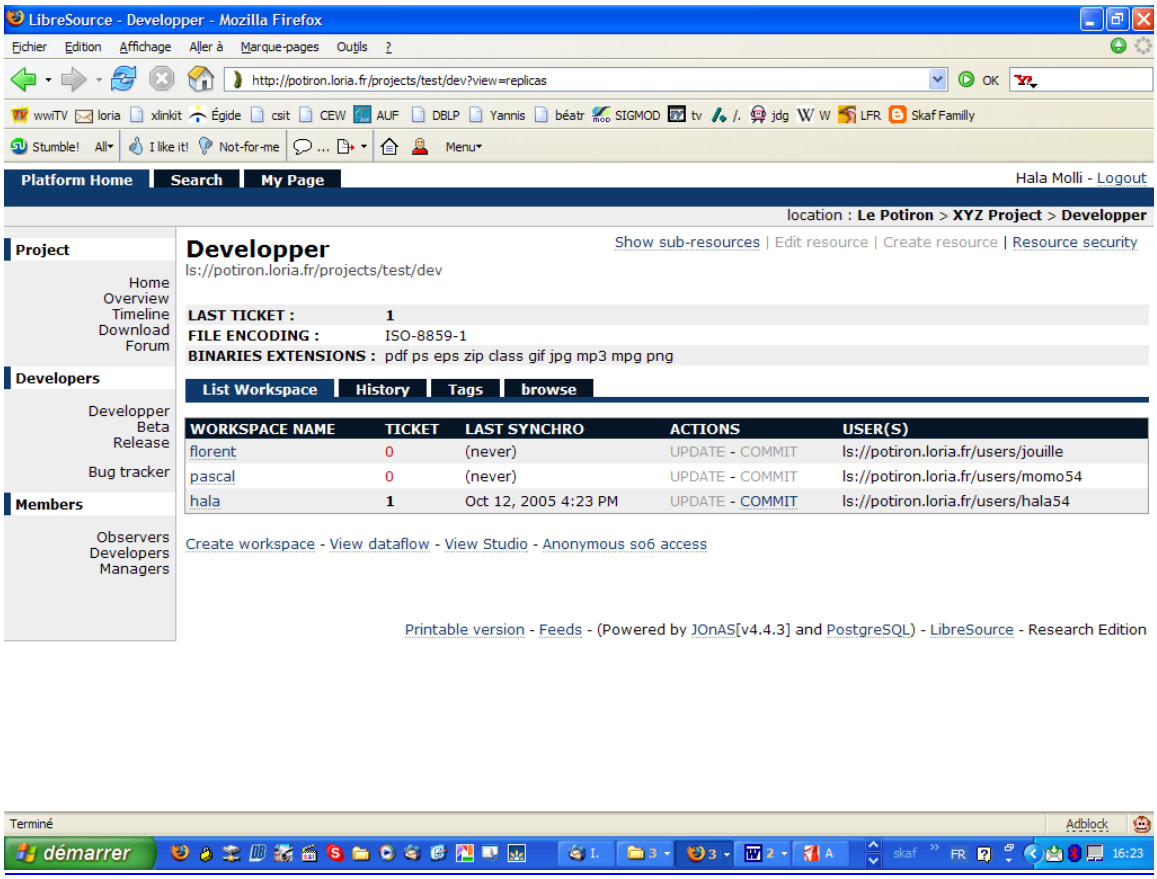

Figure 2: Developers' Workspaces in LibreSource

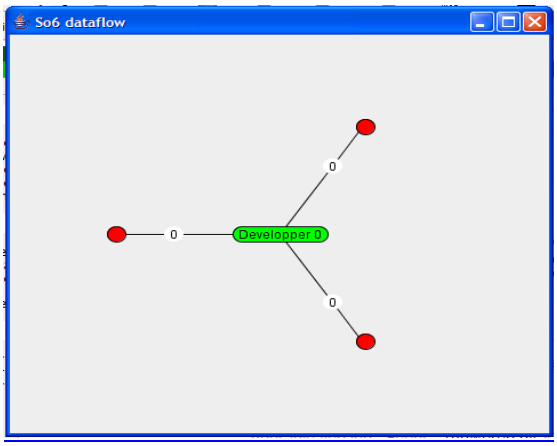

Figure 3: Propagation of changes from Developer queue

- 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 *Resource* is used, only updates are propagated to the beta users.
- At some stage, the latest version of XYZ is released to users (all members of the software development team). They can obtain it by invoking operation *Update* on another queue resource called *Release*.
- Developers communicate with each other using a bug tracker resource. Changes and bugs are reported from users back to developers via the forum resource.

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.

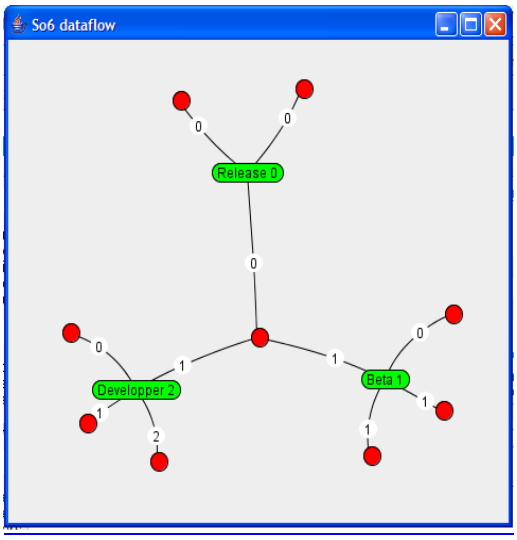

Figure 4: Synchronisation dataflow between Developer, Beta and Release queues

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 in place of 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.

Finally, from the educational perspective, this software development learning/teaching activity can be made more complex by using different collaborative scenarios. For example, it is possible to get students from one group to test solutions of another group and report back the problems. Furthermore, the same solution can be used in other teaching/learning scenarios to support for example collaborative writing of an assignment. However, code development and the associated design methods make collaborative software development much more complex than writing simple text. This is why this particular solution is tailor-made for software engineering students engaged in collaborative development of any type of software system.

5 A QUICK TOUR OF LIBRESOURCE SERVER

The LibreSource server can be accessed through a Web browser such as Internet Explorer or Mozilla. During navigation, LibreSource may prompt the user to download some Java programs. In this case, the user should choose the option “Open File” instead of “Save File”.

Figure 5 shows the home page of the server LibreSource.loria.fr. In this figure, we can see some generic information associated with every resource e.g.:

- The resource tree path is shown on the top left side of the window. Each resource is uniquely named and can be included as a reference in other resources (e.g. a Wiki page can be made to point to a workspace or another Wiki page).
- Login information on the top right, showing who is logged in or a login menu (if no one has logged in yet).
- The top line shows some generic operations that can be performed on the resource tree. At this stage, we can only list the children of a particular resource in the tree.
- The left and middle windows contain user-defined text, menus, links, that provide easy access to resources such as projects, information etc. located on this server.

After logging, users can navigate through any of the links provided. As most pages are Wiki-enabled, so users can also modify the content of these pages (providing they have the right permissions). The LibreSource evaluation package is available online.

6 CONCLUSION

The main objectives of this paper were twofold: (i) to illustrate an innovative teaching/learning activity designed for software engineering students and (ii) to describe how this activity can be supported by collaborative technology called LibreSource. Although the main focus of this paper is on collaborative technology, we argue that when technology is used in the education domain, it is necessary to adopt the top-down approach. Thus, it is necessary to start from the intended learning objectives and learning designs and then look for possible technical solutions. Design of this activity is based on the latest theory of learning designs and principles of problem-based learning.

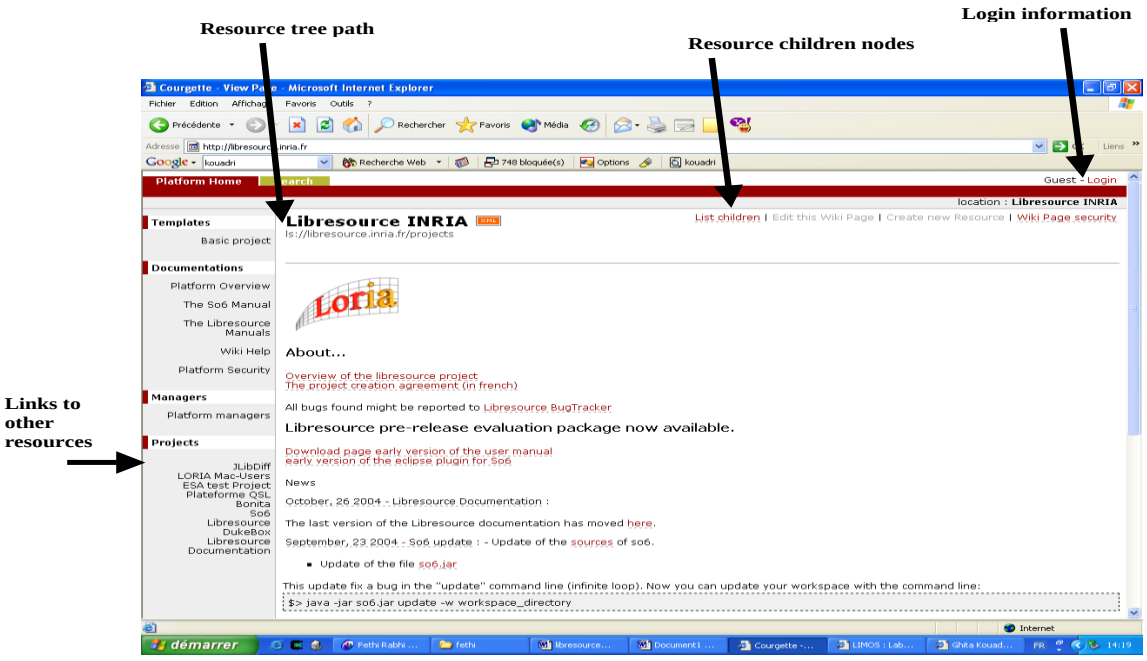

Figure 5: LibreSource home page

The same educational activity could be further extended to support more complex scenarios. For example, it could support peer review process where one group of students test solutions developed by other groups. Furthermore, the same collaborative technology can be used in other, non-technical, disciplines to support, for example, collaborative writing of an assignment. Our current and future work include further development of LibreSource technology and its application to collaborative activities in the domain of eLearning, business and software engineering.

REFERENCES

Louis, R., 1999. Software agents activities. In *ICEIS'99, 1st International Conference on Enterprise Information Systems*. ICEIS Press.

Smith, J., 1998. *The book*, The publishing company. London, 2nd edition.

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

Ehrmann, S.C. (1997), Asking the right questions: What does research tell us about technology and higher learning?, Annenberg/ CPB Learner Organisation, available from www.learner.org/edtech/rscheval/rightquestion.html.

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

Koper, R. and Tattersall (eds.) (2005), Learning Design: A Handbook on Modeling and Delivering Networked Education and Training, Springer, Berlin.

LibreSource [online] available from www.librasource.com

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

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.

Race P. (1999) Practical Pointers to Flexible Learning available from (http://www.lgu.ac.uk/delibrations/flex_learning/race_content.html

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

Schon, D.A. (1983), The Reflective Practitioner: How Professionals Think in Action, Basic Books Inc., USA.

Schon, D.A. (1987), Educating The Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions, Jossey-Bass Inc., USA.

SourceForge [online] available from http://sourceforge.net/

WebCT [online] available from www.webct.com

ZOPE [online] available from http://www.zope.org/