A Peer-to-Peer Collaborative Framework Based on Perceptive Reasoning

Olivier Eymard, Eric Sanchis, Jean-Louis Selves

A Peer-to-Peer Collaborative Framework Based on Perceptive Reasoning

Olivier Eymard, Eric Sanchis, Jean-Louis Selves
Laboratoire Gestion et Cognition
Université Paul Sabatier
115 route de Narbonne, 31077 Toulouse, France
{eymard, sanchis}@iut-rodez.fr, jean-louis.selves@iut-tlse3.fr

Abstract: Collaborative Learning Systems are often developed to enhance the communication between the actors. These systems suggest a lot of interactivity to collaborate and a great efficiency to look for and to find solutions to a great number of problems. But in certain educational contexts, numerous communication tools distract the users from their main collaboration activity. RetroMob is a peer-to-peer (P2P) lightweight framework which focuses on the collaborative activity dedicated to the emergence of individual solutions built from the parallel perception of solutions suggested by the other members of the group. For that, the mechanisms of interaction between the users are restricted to the discovering and to the transparent and anonymous transfer of the solutions suggested by the peers.

1. Introduction

By integrating tools as various as e-mail and contact list management, threaded discussions, instant messaging, calendar or whiteboard, Collaborative Learning Systems like BSCW [6] aim at increasing significantly the quantity and quality of the communication tools used by peers in order to improve the learning conditions of the group they belong to. On the other hand, some systems like EDUTELLA use a P2P architecture to make a large number of learning resources available at a low cost. A characteristic common to both classes of systems is that they favour collective learning practises [9].

Contrary to the preceding tools, the RetroMob framework focuses on the collaborative activity dedicated to the emergence of individual solutions. In this case, each actor uses the work of the other participants independently of their availability. The procedure includes two stages. First, each member of the group works alone and carries out the solution according to his own knowledge. Then people collaborate and benefit from the work of the other participants to find the best solution. The platform's aim is to trigger a specific cognitive process – the perceptive reasoning – for each member of the group. The result of this process can quickly be described in the following way: a member of the group having a personal solution (true, approximate or false) to a target problem can improve the quality of his own solution by comparing it with the solutions suggested by the other members of the group. For that, RetroMob provides to each participant, in a transparent and anonymous manner, the other suggested solutions. In other words, contrary to existing solutions like Wikipedia [12] where the finality is to define a shared solution patiently built by the community, RetroMob is a framework designed to increase the emergence of the best individual solutions in a collaborative context. All these solutions arise from unconscious treatments triggered by the visual perception and by the instinctive evaluation of other suggestions.

This paper is structured as follows. The next section introduces the working assumptions, the main concepts and the design choices of the RetroMob framework. Section 3 presents an actual application in the educational area built upon RetroMob and used in our Institute of Technology.

2. A framework to improve individual solutions

2.1. Working assumptions

In order to have a lightweight framework, we made some assumptions. They arise from the focusing on the specific domain of individual solutions rising from a collaborative activity.

Firstly, collaborative work is a cognitive process. It emerges from the interaction between actors, technological artifacts and processes. The material traces resulting from these interactions are at the beginning of the best individual solutions.

Secondly, the initial information induces the production of complementary information. For example, in a convergent engineering situation, initial information streams circulate in a rough state until the activities to complete have been defined. Propositions for imperfect but constructive solutions are at the base of the emergence of a final solution. In another area, the educational field [1], processes where students work together toward a commonly known goal by discussing and questioning each other are very effective.

Thirdly, the cognitive process is fed by perception and parallel processing of information. Reasoning and problem solving are considered as the result of a set of parallel processes which is largely unconscious and related to the perceptive activity of the subject. Some researchers as Holyoak [4] suggest that the traditional approach is not adapted to problem solving, especially when the problem is badly defined. According to him, reasoning and problem solving can be considered as a more global process with parallel and largely unconscious treatments related to the perceptive activity of the subject. We suppose that this perceptive reasoning plays an important part in learning and that it must be taken into account by the software infrastructure.

At last, to facilitate the emergence of a solution and the capitalization of knowledge, information must be multiple and *perceptively* available. We think that it is necessary to memorize this information in order to have elements of traceability. Then, this information can be exploited only using the corresponding means of research and consultation information. From these assumptions, we defined the minimal list of services for the RetroMob framework.

2.2. Requested services

To carry out our framework based on the collaboratively assisted work, a limited number of services has been defined to communicate. According to the previous assumptions, we have identified four essential services: the *get service*, *change service*, *propagation service* and *retropropagation service*.

The *get service* allows transferring information between users across a network. The information support used to communicate across this network is a *document* file. A history of the exchanges is filled in by this service to know the participation of each one.

The *change service* authorizes the creation or the modification of a document exchanged between users.

The *propagation service* is dedicated to a user with special grants. It enables him to browse a tree of diffusion and to retrieve information from peers. This service provides a vertical / hierarchical treatment of the data between a specific peer and the others.

The *retropropagation service* authorizes a horizontal peer to peer treatment of the information. It makes possible for each peer to reach data from the others in order to improve their own work.

Model of interchanged documents

These services involve exchanging information between the group members. To this end, we have adapted the concept of *Intermediate Design Objects* (IDO) introduced by Boujut and Blanco [2]. IDO entities represent all the concrete and abstract objects which are produced or used during the action of design and which connect tools, procedures and actors. The great applicability of this concept allowed us to define a minimal model of an IDO, perfectly operational within the framework of a collaborative activity.

The model of IDO we have chosen breaks up into two parts: a main element, called *document* which is the visible part to the users of the IDO, and a composite part called *history*, which is partially or totally masked to users according to the application.

The *document* is the principal aspect of the IDO for the actors. It is the element which triggers users' actions (consultation, modification, complementary contributions of information, etc.).

The *history* synthesizes all the enrichments brought to the document by actors (possible copies, versions, displacements, etc). It summarizes in a total way the state of development of the IDO at a given moment.

2.3. General architecture

Figure 1 shows the general architecture of RetroMob. It corresponds to the first two layers (*Communication* and *Mobile Agent* layers).
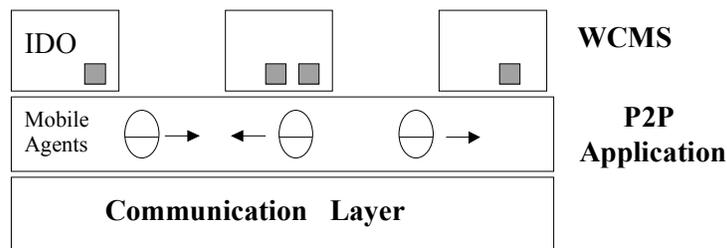

Figure 1. RetroMob Architecture

A user application based on RetroMob is ideally developed with a *Web Content Management System* (WCMS) which provides all the tools necessary to the creation, edition, modification and storage of contents intended for the publication on the web. Moreover, this kind of systems allows the management of the user interface along with the permissions of the users. The interface presented by the RetroMob platform to the different WCMS is constituted by the four services ***get***, ***change***, ***propagation*** and ***retropropagation*** described in section 2.2.

Peer-to-Peer Application

The designer of collaborative applications has mainly two models to articulate the interactions between the users: the *Client/Server* model and the *Peer-to-Peer* (P2P) model.

In the *Client/Server* model, the server is the only provider of content and service, and represents the central entity for clients. Tools such as BSCW [6] or Wikipedia [12] are built according to this model.

On the contrary in the P2P model, peers are able to act as client and server at the same time. Furthermore, peers are accessible by other peers directly without using intermediary entities. For example, the EDUTELLA project [8] is built upon a P2P architecture. RetroMob also uses this model because the *retropropagation* service places each user on the same level of sharing. More precisely, RetroMob is an unstructured P2P framework with no central server, even if in a collaborative activity, the initial work is started by a user who plays the role of coordinator and represents a central entity.

P2P systems generally offer two main services: a *file search* service and a *file transfer* service. The *file search* service is one of the most delicate parts to design and implement because it determines the global performances of the application. But in the RetroMob framework, the history part of each IDO allows *mobile agents* to easily retrieve information.

Mobile agents

To communicate between peers across the communication layer, we chose the mobile agents technology. In this way, the local code sends an autonomous code to the remote site which executes it, and during its execution the received code has the possibility to transfer itself on another site.

Three independent properties were integrated into an agent: *mobility*, *replication* and *host perception*. *Mobility* allows an agent to migrate from a peer to another. This is the most important property of our agents. *Replication* allows an agent to create a clone locally. And finally, the *host perception* property is necessary for the agents moving.

The mobility property was particularly studied and implemented since the middle of the nineties [5] [10] due to the inadequacy of the client/server model used until then. Indeed, this kind of systems used a great part of the bandwidth to decrease the number of exchanged messages and to improve performances and the dynamic of the applications. White [10] showed that it was preferable that the client's software agent interacts locally with the server. Then, when the treatment is finished it returns on its original site. Only the code of the agent is transmitted and not data. The mobility property also brings a solution to mask the heterogeneity of the platforms and improve fault tolerance according to a mechanism of agent replication. Thus, the execution of a process can carry on in spite of errors, which are accurately managed.

In the RetroMob framework, a software agent possesses an architecture composed of two sub-systems. The first sub-system implements what concerns the achievement of the task assigned to the agent: the discovery, the analysis and the transfer of the IDO. The other implements the three properties (mobility, replication, host perception) mentioned above.

In order to illustrate the RetroMob framework used, we designed an application dedicated to learning language. The next section presents this application after a brief description of the educational context.

3. A collaborative application in an educational context

3.1. Collaborative learning applications

In a recent study [1], J. Biström analyzed the possibilities of the P2P environments in the educational area and reported that the problem in educational tools is not the architecture but the functionality for the educational methods and needs. Most of the P2P environments like Groove [11] or Helpmate [3] offer a lot of functionalities and some of them like information sharing, instant messaging, telephony or whiteboard are not particularly well suited to small groups in collaborative learning context. Other researchers, Melin and Cronholm **Erreur ! Source du renvoi introuvable.** have studied the learning and examination of project oriented student work. They report the following four categories of problems:

- Coordination: different desires about working times and geographical distance between project members.
- Heterogeneity: different previous knowledge, techniques of study and ways of thinking.
- Motivation: different levels of ambition and commitment to task and goal.
- Social: different personal chemistry and abilities.

Moreover, with a large number of tools, group members spend more time to choose the good functionality and to learn how it runs than to cooperate actively with peers. Indeed, we think that the interactivity between the users must be controlled. This is particularly true when the main objective of the application is to allow a user to lead a personal work suitably while benefiting from the work of the other participants.

3.2. A collaborative translation application

According to the RetroMob framework described above, we designed a Collaborative Translation Application (CTA) with specific characteristics. Compared to the tools mentioned in the previous section, our

application provides the means necessary to the control of the services of communication between the users with a lightweight infrastructure. This application works as follows:

First stage: a language teacher publishes a text to be translated intended for an open class. It means that people can be geographically scattered and users outside the class of the teacher can participate in the collective activity of translation. Moreover, the anonymity of the students is kept. In accordance with the IDO concept, the text for translation is registered in a main file (the original text file) and its history part is stored in another associated file.

Second stage: the students can download the text to be translated and publish a proposition of translation. When a student wants to download the original text file to be translated, a get request is sent to the host known by the student who keeps this file. Therefore, this action is memorized into the history of the downloaded file. For each get request, the user's IP address is recorded. In the scenario presented in *Figure 2*, student #1 and student #2 download on their host the original text file to be translated. Consequently, the history file kept on the teacher's host contains the information: "GET HOST #1" and "GET HOST #2".
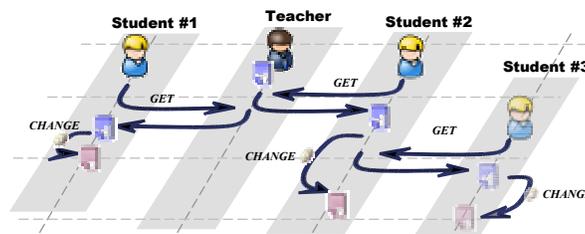

Figure 2. A simple scenario of requests

A document can be downloaded from another host than the teacher's. For example, the duplicated document on host #2 can interest student #3. In this case, a get request is sent by student #3 to the corresponding host. Thereby, the history file present on host #2 contains a new line: "GET HOST #3".

Third stage: the teacher can consult all the translations made by the students. This is the propagation service. The main consequence of the independent duplications of the original text between the students is that the teacher cannot know the location of all the copies of his document. To be able to build a map of its diffusion and to retrieve the suggested translations, the system uses mobile software agents. Figure 3 illustrates the work of mobile agents when the language teacher uses the propagation service.
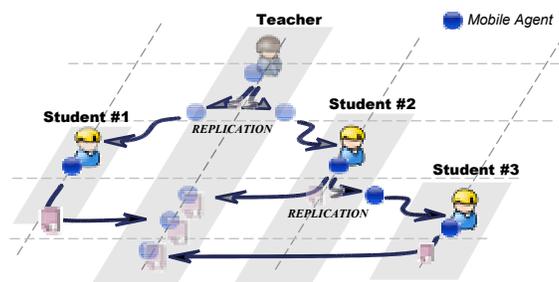

Figure 3. Propagation service called by the language teacher

With the propagation service, the teacher launches a mobile software agent to analyse the local history of the document to be translated. Its history file contains the set of users having requested a copy of the original document. In our example described on Figure 3 student #1 and student #2 downloaded the document from teacher's host. As a result, when the software agent analyses the history file of the teacher, it duplicates itself as necessary and each clone goes towards the different implied hosts. In the same way, when a clone arrives on the host of student #2, it analyses its local history file and creates a new clone which goes to the host of student #3. The work of each student who suggested a translation of the original text is transmitted to the teacher. With the different history files,

the language teacher is able to build a map of the original text file propagation and to retrieve the translations scattered in the network. This process is completely asynchronous.

Fourth stage: each student has the possibility to improve his own translation from submissions of the other group members. This functionality refers to the retropropagation service. Indeed, the emergence of a better translation produced by a student can occur by merging the different translations produced by other participants. The retropropagation has a functioning similar to the propagation, the main difference being situated at the level of its activation place. Indeed, all the users except the teacher can activate this function. The aim is the same, i.e. retrieving the documents from the machines participating in the application but there are a few differences between the executions of retropropagation service and propagation service.

For example, in Figure 4 when student #1 starts the retropropagation service for a document, the mobile agent analyses its local history file and navigates according to the contents of the referred remote history files which have been read. Students are just supposed to be interested in the translations and not in the history parts of the IDO. Thus in our example, only the translations student #2 and student #3 are downloaded.
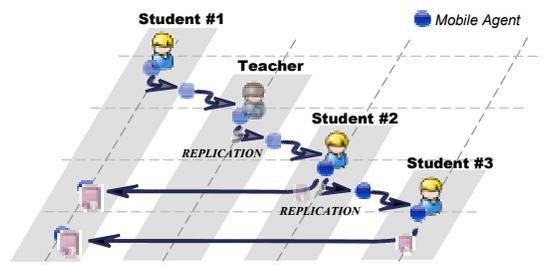

Figure 4. Retropropagation service used by student #1

With the retropropagation process, we make the assumption that the development of a translation may be considered as a process of search of a solution through interactions among several actors.

3.3. IDO implementation

The application makes the distinction between three sorts of documents: *user documents*, *propagated documents* and *history files*.

A *user document* is to be understood as a document downloaded from a machine or modified by the user on a machine. The download is explicitly done by the user. At the start of the application, a document is created by the teacher. Once registered, this document becomes accessible to all the students who can download it on their own machine. When it is downloaded on a machine, an associated history file is then created. It is the user document which circulates when (retro) propagation takes place between the different machines.

A *propagated document* is a document which is brought by a mobile agent from a remote machine. These documents are stored in a special directory. The name of a propagated document contains the IP address of the machine it comes from or the name of the user it belongs to.

A *history file* is associated to a user document. It contains the events that occurred to the user document. Three kinds of events can be found in a history file: PARENT, MODIFIED and GET. The first one indicates when a document is downloaded from a remote machine. The next event is triggered with the change service when a document is updated. Finally, the GET event is activated when a remote machine downloaded a document. In all these cases the IP address of the actors is registered as argument of the event. Figure 5 shows the history file of student #2 in the scenario previously defined on *Figure 2*.

```
PARENT 192.168.2.30
MODIFIED 192.168.2.32
GET 192.168.2.29
MODIFIED 192.168.2.32
```

Figure 5. History file

Student #2 is working on the machine which IP address is 192.168.2.32. The file containing the text to be translated has been downloaded from machine 192.168.2.30 (1st line). The file has been modified by him (2nd line), then the original document (and not student #2's translation) has been downloaded by student #3 on machine 192.168.2.29. Finally, the last line of the history file indicates that student #2 has changed his translation.

3.4. CTA user interface

The user interface designed for the Collaborative Translation Application is as simple as possible in order to be clear and intuitive even for an occasional user. Working with the CTA software is reduced to a simple local navigation to make the application accessible to most users.

Figure 6. CTA teacher's homepage

Figure 6 and Figure 7 present the teacher and Student homepages of the CTA software. We chose Zope [13] as CTA user interface. Zope is an open source WCMS which provides many tools for content management and information publishing on the web: it integrates its own database compatible with the SQL language and its own programming language. Applications designed with Zope have an Internet oriented interface, i.e. all the possible actions are accessible via a simple Internet browser.
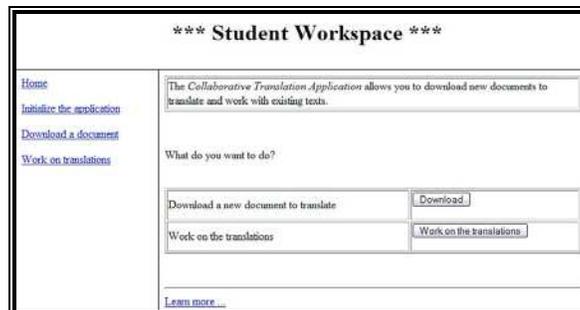

Figure 7. CTA student's homepage

4. Conclusion

The experience collected during the use of the CAT application by various classes of students showed that the principles of design of the RetroMob platform were judicious. Indeed, while concentrating on a single cognitive process and by proposing a set of very few but specialized services, the CAT/RetroMob couple gained in effectiveness and facility of use what it had lost in generality of use.

Nevertheless, this framework seems to be usable in the collaborative areas where the activities of training are based on the emergence of new ideas or solutions built from individual and collective works.

5. References

[1] J. Biström, "Peer-to-peer Networks as Collaborative Learning Environments", *HUT T110.551 Seminar on Internetworking*, Helsinki University of Technology, 2005.

[2] J.F. Boujut, and E. Blanco, "Intermediary Objects as a Means to Foster Co-operation in Engineering Design", *Journal of Computer Supported Collaborative Work,* Vol. 12, Issue 2, 2003, pp. 205-219.

[3] K. Curran, "Peer-to-Peer Networking Collaboration Within Education", *Journal of Educational Multimedia and Hypermedia,* Vol. 11, No 1, 2002, pp. 21-30.

[4] K. Holyoak, "Problem Solving", In D.N. Osherson and E.E. Smith, *Thinking: An Invitation to Cognitive Science,* Vol. 3, The MIT Press, Cambridge, Massachusetts, 1990, pp. 267-296.

[5] D. Johansen, R. van Renesse, and F.B. Schneider, "Operating System Support for Mobile Agents", *Readings in Agents*, Morgan Kaufmann, San Francisco, USA, 1997, pp. 263-266.

[6] K. Klöckner, "BSCW - Educational Servers and Services on the WWW", in *Proceedings of the International C4-ICDE Conf. on Distance Education and Open Learning "Competition, Collaboration, Continuity, Change"*, Adelaide, September 9-14, 2000.

[7] U. Melin, and S. Cronholm, "Project Oriented Student Work – Learning and Examination", *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, Leeds, UK, June 28-30, 2004.

[8] W. Nejdl, B. Wolf, C. Qu, S. Decker,bM. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, "EDUTELLA: A P2P Networking Infrastructure Based on RDF", in *Proceedings of WWW2002*, Honolulu, Hawaii, May 7-11, 2002.

[9] W.M. Waite, M.H. Jackson, A. Diwan, and P. Leonardi, "Student Culture vs Group Work in Computer Science", *In 35rd ACM Technical Symposium on Computer Science Education (SIGCSE)*, Norfolk, VA, March 2004.

[10] J.E. White, "Telescript Technology: Mobile Agents", *Software Agents*, J. Bradshaw Ed., MIT Press, 1996.

[11] http://www.groove.net

[12] http://www.wikipedia.org

[13] http://www.zope.org