International Conference on Engineering Education and Research "Progress Through Partnership"

Collaborative Compilation: An Experience on Using Groupware Tools Applied to Computer Programming Learning

Manuel GARRIDO, Jesús C. PANCORBO, Justo N. HIDALGO
Department of Computer Science and Engineering, University Antonio de Nebrija, c/Pirineos, 55 MADRID, mgarrido@nebrija.es, jpancorb@nebrija.es, jhidalgo@nebrija.es

KEYWORDS: *groupware, collaborative, programming lectures, CSCL*

ABSTRACT: *The aim of every teacher is to create a learning environment based on communication, on opinions and exchange of knowledge, and on the participation of every student, so that an effective and ideal learning process can be achieved. Therefore, the use of adequate mechanisms, methodologies and group dynamics which support this goal, is becoming a necessity.*

This paper deals with the design and development of a groupware application in a synchronous programming learning environment. An innovative way for teaching programming is proposed, by using a groupware system which creates an ideal atmosphere of interaction among teacher and students, following the social constructivism theories to build a CSCL environment. The idea lies in using a teacher-managed groupware collaborative learning application at lectures on computer programming, which allows every participant to see the coding of a computer program, how the compilation is made and the results that the compiler returns, as well as offering participants the chance to contribute on the development process, learning by the mistakes and successes of everybody; in this way, the lecture mixed the theoretical concepts with how they are applied in a real case, turning the classroom into an open learning space of communication and collaboration among the class members. Moreover, this system allows the teacher to manage the different programming stylistic ways and how the students must go on with a problem.

To carry out this idea, a distributed tool called COCO (COlaborative COmpilation) has been developed.

1 INTRODUCTION

Nowadays, a typical lecture on programming methodology consists of different algorithms and syntax explanations through slides and blackboard notes, along with smalls laboratories and programming assignments that students work on outside of the classroom by their own, in which they put the theoretical concepts into practice. Usually, most of them face up to similar doubts and problems which must be solved one by one by the teacher with an increasing waste of time, being extremely hardly to teach the students to get the correct way of the reasoning process when they confront with an algorithm resolution.

In some other occasions, teacher-led labs are put into practice, where the instructor must solve students' inquiries one by one, which usually means leaving many questions unanswered, and, also, wasting everybody's time, so there can be a delay on some students' learning process with respect to the most advanced ones. This problem is increased when some students have no previous programming skills.

This paper proposes the use of collaborative software to programming lectures; the use of these systems will allow for everybody to learn from everybody, with exchange of view points and knowledge among students and instructors, and to see and understand different ways of programming and facing different challenges, even appreciating the different mistakes everybody can commit, so not to produce them again.

To achieve all of these goals, a software application has been developed, which permits students to see all class members from their own computers, how practice comes from theoretical concepts, how programming skills are learned, how a program is compiled, what types of errors are produced, and how they are solved.

The remainder of this paper is organized as follows: section 2 underlines the theoretical framework under this paper is mantained. Section 3 explains how the developed application works and the functionalities offered to users as well as the benefits this method applies to the programming lectures. Section 4 shows the design of the application and the tools used to carry out the idea. Finally section 5 lists some of the research lines that can be taken from the idea it is presented and the improvements the application could have as well as final conclusions about the paper.

2 BACKGROUND

What is an activity? An Activity is performed by a Human being which is motivated by a Goal which allows them to solve a particular Problem, by using different Tools, collaborating with other Groups and following a set of determined steps (Work Distribution). Figure 1 shows the architecture of these concepts, as defined by Engerstom in 1987.

Achieving an Activity is subject to cultural factors and conventions (Standards) inside of a particular context.

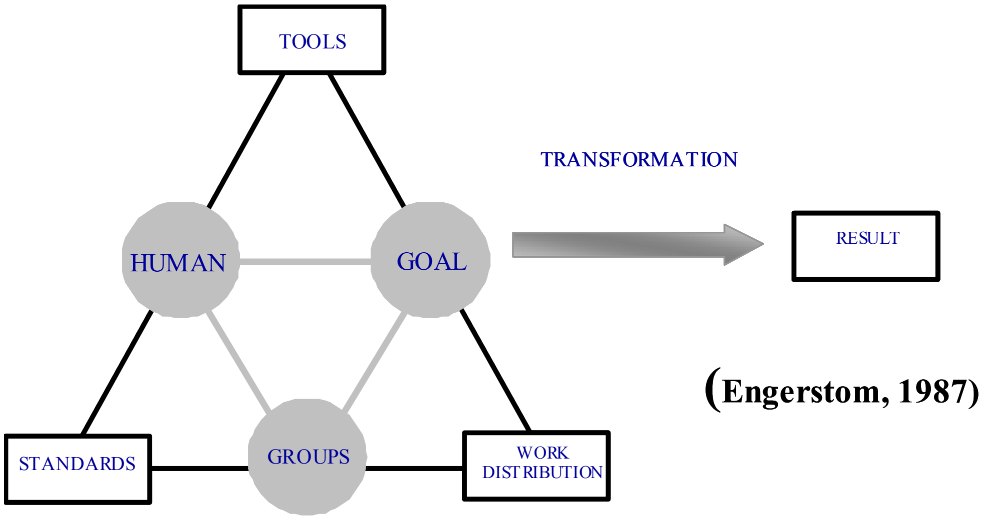

Figure 1: Structure of an activity

2.1 CSCL: Computer-Supported Collaborative Learning

Previous paradigms differ in their theories, but share a psychological view –either conductist or cognitive- of learning, like an intrapersonal phenomenon which happens in every individual's mind, and which can be studied with Psychology's classic experimentation methods. Thus, current educational software is established as an individual, independent system for each single student. However, the new CSCL paradigm –a new learning trend- is based in a socio-cultural view of cognition. Learning processes must be esentially social, so new methods of study and experimentation must be used, such as those coming from antropology, sociology, communication and pragmatics. From this point of view, technology is interesting as to how much potential it can offer to create, favor or enrich different learning interpersonal contexts.

2.2 Social Contructivism

Knowledge is something which must be built, but must be seen as something social, and not only invidivual: the result of a spiral of causalities, starting from a specific level of individual development, this theory allows for the human beings to participate in social interactions which produce new individual states, and so on.

The key element to intellectual development is defined in terms of conflict and coordination. A "socio-cognitive" conflict which comes out from the subjects' different perspectives of a same experience, might act as a cathalysis to unchain a "viewpoint coordination".

2.3 The soviet historic and cultural theory

Usually known as “cultural theory”. For cultural psychologists, the environment where intelligent human activity is created and produced, includes different artifacts, technologies and rituals which have been acquired and developed in a social way, alongside a historical process. Any conceptual framework about cognition and personal learning must take into account that it exists in a socially-arranged environment, therefore other element must be modeled: the inner relationship between thought and the different means which the cultural environment offers.

Cognitive functions are experienced in an inter-mental level before they exist in an intra-mental one. In other words, our mental reflections come from experience which firstly appear through a social interaction.

2.4 Situated cognition

Knowledge is not conceived as a catalog of mental representations, but as something which is circunstantially produced when interacting with the real world. This theory’s focus is the study of how cognitive agents act in a particular environment. Knowledge comes out from those interactions, and learning, even initially situated, can be possibly transfered through a socio-cultural intervention. Therefore, learning is conceptualized as a process of culture acquisition from a community of practicioners. For example, students’ daily activities must be related to professional communities so that they can own their necessary concepts.

3 CREATING A NEW PROGRAMMING LEARNING ENVIRONMENT

Based on the collaboration learning theories we want to improve the way that programming methodology is taught, creating an innovate learning environment in the programming lectures assisted by the use of a groupware application.

The intention is make use of this kind of applications in order to exploit the knowledge exchange between the lecture members and give the opportunity to the teacher of manage and advise how the students must confront a programming problem.

3.1 The Solution: An Overview of COCO

COCO is a groupware application which allows for the class members to see, in real time, the coding and compilation of a program, in any programming language, and let them participate on the development process. COCO is entirely managed by the teacher, who can drive the lecture combining their explanations with practical examples and encourage the participation of the students to contribute with their point of view, enriching the learning process.

3.2 How COCO Works

COCO is a tool which adds a new way to carry fordward a computer programming lectures. In the following sections, this paper will show how a lecture is developed by using COCO as a helper tool.

When explaining theoretical concepts, or when studying different methods or algorithm when solving programming problems, teachers can choose using COCO so show their students how to achieve it. COCO is a turn-managed application, and this management is accomplished by the instructor, who owns the turn at the beginning of the session. From that particular moment the teacher can choose to yield their turn and hand it over some other class student so that some problem is solved, an algorithm is developed, or an error is found. The teacher can recover the turn in any moment during the execution of the session.

“Turnless” students have most of the application’s functionalities disabled, so their role will be to pay attention to the instructor’s explanations, and be ready to intervene and participate when the student assigns them the turn.

The class member owning the turn can open any file from their computer, create a new one, editing a file –that is, coding a program-, text search inside the file, and, above all, compile the active file by using any language compiler in their equipment. Every action this student realizes, will be seen by all of the class participants in real time, so that they can follow in an attentive way how the student in control program, how stylish the program is and the type of reasoning in using the program; this way, they can express their ideas about this behaviour whenever the teacher wants it. An open debate can be very convenient once in a while, with the collaboration of all the class members, and where the teacher can make everybody see the correct reasoning behing algorithm solving.

Besides, COCO is also able to parse the possible errors returned by the compiler. Nowadays, only errors returned by the bcc32 compiler are parsed.

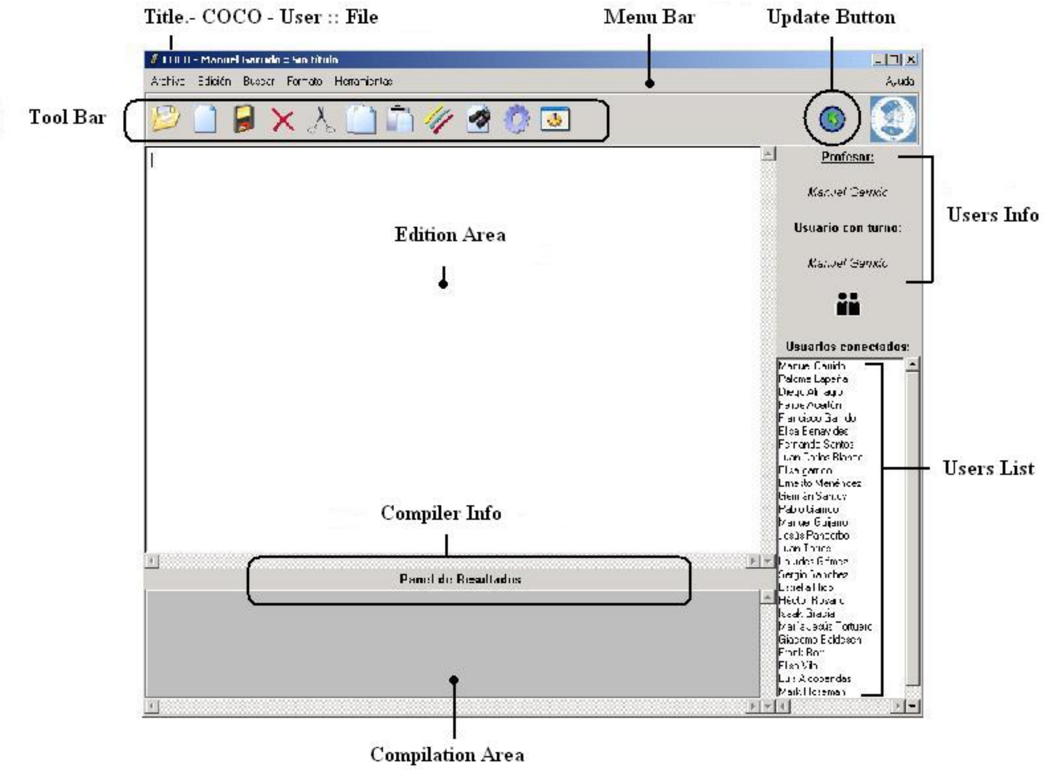

Figure 2: Snapshot of COCO's main window

This tool is currently in use in the Antonio de Nebrija University (Madrid, Spain), in the computer programming methodology lectures, and is being tried for other different subjects such as Distributed Systems and Internet Applications Development.

3.3 COCO: A Real Case

Place: *Antonio de Nebrija University.*
Date: *03/11/2004*
Lecture:*Computer Programming Methodology I (C++)*
Course: *First year of Computer Science and Engineering.*
Concept:*Pointer reference.*

Students have learnt, by cooperating among them, the important concept of Pointer Referente in C++, never explained before. The idea was to set up an Activity as a "problem to solve", that is, to code, and guide the class members so that they can solver by using the concepts learnt previously (that is, Pointers).

This activity allowed for the students to discuss, make mistakes, and understand that their solution, the only one built step-by-step, did not actually work. At the end of the Activity the Reference Pointer theoretical concepts were explained; this way, the solution is perfectly understood.

The potencial of using COCO comes from the fact that students have built together a solution just by themselves, and have solved the different problems and challenges as a team, because in every moment every one of them have contributed with their own ideas, being them visible in real time through the cooperative compiler –COCO-.

This tool and its use strengthens the Social Constructivism theory explained above.

Step 1:

Problem introduction: a C++ function must be coded, which create an integer variable using dynamic memory, and in a way such that the main program can access it.

Step 2:

The class members have thought about which functions would be hended, and one of the students has coded the following skeleton by using COCO:

```
void dynamic(const int &x)
{
}

int main()
{
}
```

Step 3:

A discussion is promoted about how the function must crear "dynamic memory", and how a parameter is passed as a pointer. Another student is given control to code the following:

```
void dynamica(const int &x, int *p)
{
 p = new int(x);
}

int main()
{
}
```

Step 4:

Other students finishes the following code. Everybody agrees:

```
void dynamic(const int &x, int *p)
{
 p = new int(x);
}

int main()
{
 int *k;
 dynamic(k);
 cout << *k << endl;
}
```

Step 5:

The code is executed, and a memory exception is thrown. What is happening?

The whole class works together and returns different erroneous reasonings, until one students guesses right: if the result is to be returned inside of a variable, its address will have to be returned; as the variable is currently a pointer, the address of the pointer will have to be declared!

The coding is the following:

```
void dynamic(const int &x, int *&p)
{
  p = new int(x);
}

int main()
{
  int *k;
  dynamic(k);
  cout << *k << endl;
}
```

And all works perfectly.

Figure 3 shows the structure of this particular activity.

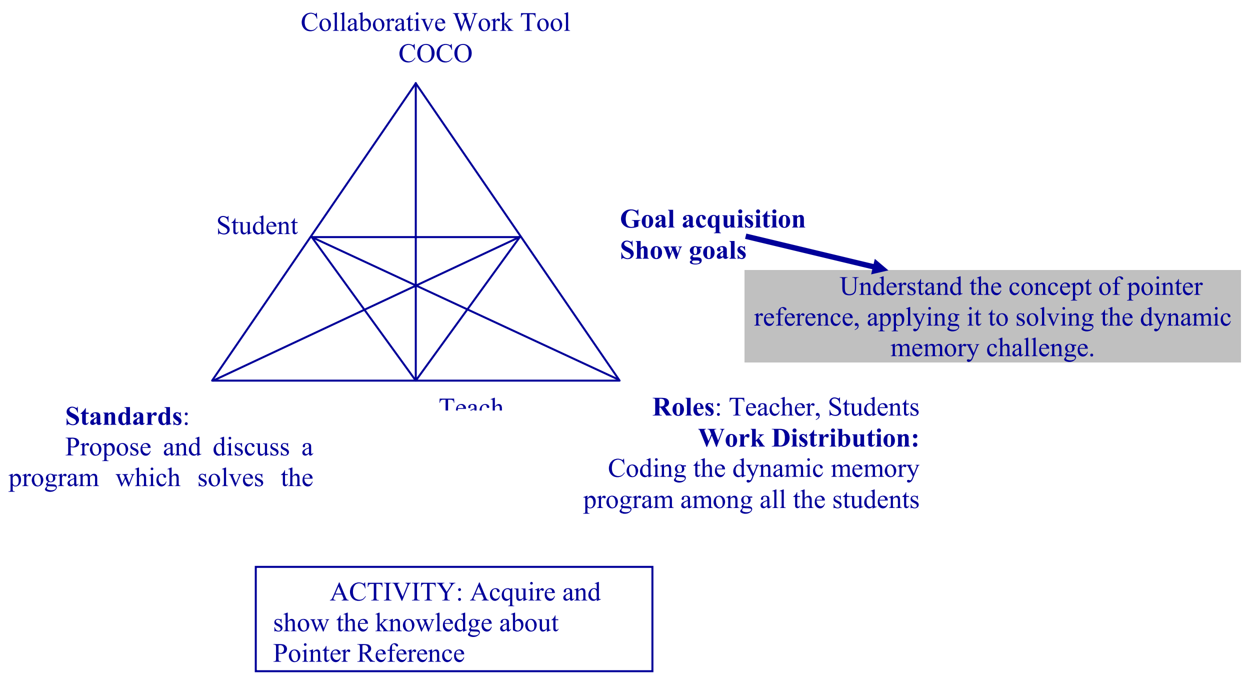

Figure 3: Structure of the Dynamic Memory Activity

4 DESIGN AND IMPLEMENTATIONS DETAILS

After a meticulous study of the different toolkits to develop groupware applications, such as JSDT (Java Shared Data Toolkit) and COAST among others, the use of GroupKit was chosen because of its simplicity and the high range of possibilities that the tcl/tk library offers on the groupware aplicattions development.

This section will define how Groupkit Works, and how it affects to the proponed system's design.

4.1 Groupkit

COCO has been developed using the tcl/tk library for building groupware applications called Groupkit, which has brought the collaborative functionalities added to COCO. Groupkit was created by Mark Roseman in 1992 at the University of Calgary and continued development till 1998, leaving a popular and robust platform. On march 2003 Groupkit's development was moved to SourceForge and Mark Roseman and Chad Thatcher retook the project.

Groupkit has three types of processes: a registrar, a session manager and conference applications. The "Registrar" is a centralized process and is the one who maintains a registry of all the conferences and the users of each one. The "Session Manager" is a replicated process which provide a user interface from which the users can see the opened conferences, create them or join to them. Groupkit provides an example of this process called "Open Registration" which we have reused, making small modifications, for the COCO use. And the "Conference Applications" which are the groupware applications, like a chat, a whiteboard or a brainstorming application, in our case COCO.

Therefore, the "Registrar", the modified versión of "Open Registration" and COCO, are the components of the complete distribution of the developed tool.

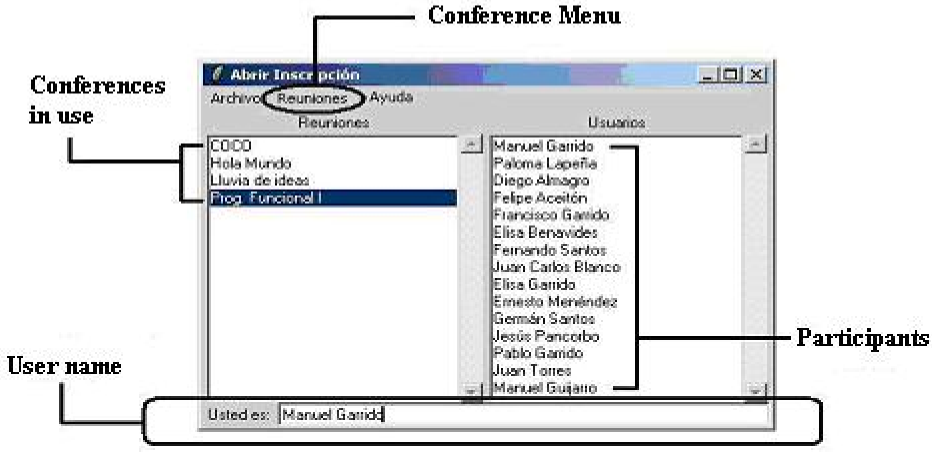

Figure 4: Starting a COCO conference

4.2 Starting a COCO conference

Two different distributions of the tool have been developed, a teacher and a student version. The teacher version corresponds to the full distribution of our groupkit's version and COCO, and the student's version is a retricted version where the Registrar process ans the options to create any conference have been deleted in order to provide total control of the lecture to the teacher –see figure 4-.

Once the teacher executes the Registrar process, the students can invoke the Open Registration application in order to connect to the session. On this application every participant can see the others and the Conferences that are in use on that moment. If a user wants to participate on one of the Conference Applications they must double-click on the name of the application listed on the Conferences List.

For example, when the teacher can open a new COCO conference renamed like "Functional Programming" and appears on their display, the main window of the COCO conference created of which they will be the manager and a new entry with the name of the conference is listed on the Conference pane of every user.

Then the students can join the COCO conference and they will be to disposition of the teacher to take at some moment the control from the application. The basic flow chart can be seen in figure 5.

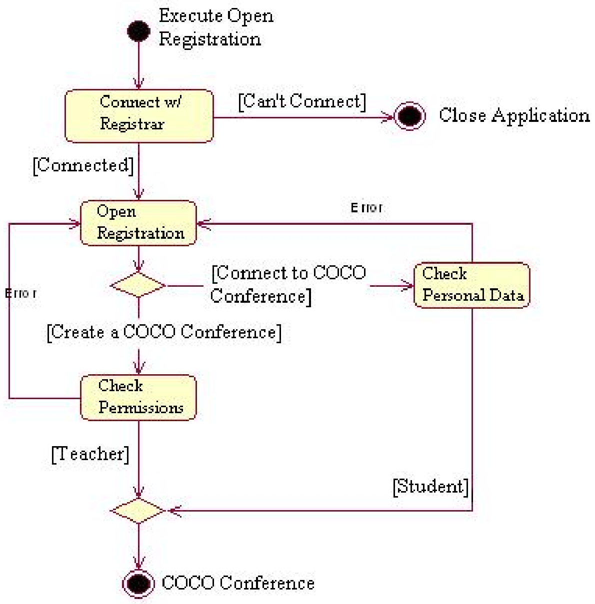

Figure 5: Open Registration flow chat

4.3 COCO Architecture

COCO is a synchronous groupware application created to be used at first on the same physical place since it hasn't got the neccesary tools, as video or audio characteristics, to support a distance learning classroom. For thus, COCO has been designed to run on a Local Area Network for a group of users working on the same classroom.

The communications among the processes are accomplished through sockets. The process "Registrar" is used to communicate the different "Open Registration" processes of each user, so to know the applications created, and which one is participating. Besides, each "Open Registration" process owns direct communication with each application managed by each user. The "Conferences Applications" processes have got a direct connection with each one of the "Conference" processes for each session user. Figure 6 shows how each COCO user process is connected to the rest of the members.

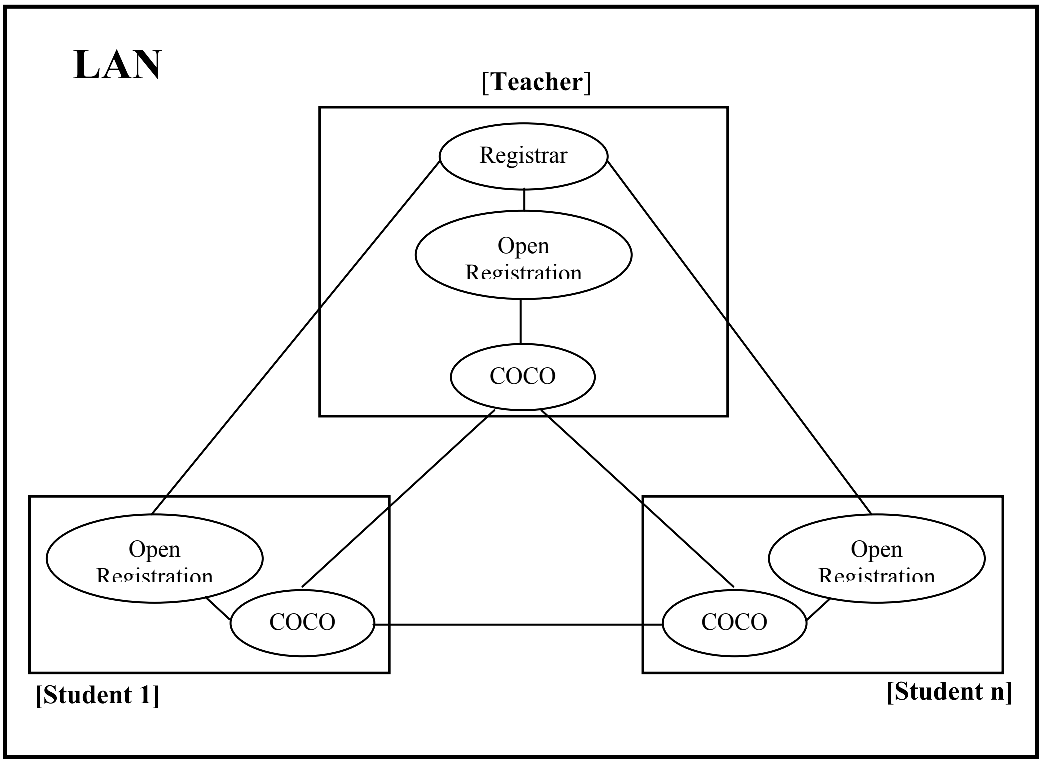

Figure 6: COCO network hierarchy

5 CONCLUSIONS AND FUTURE WORK

COCO is an ever-evolving project created as an apropriate implementation of a CSCL tool which allows for a class group to apply techniques such as group cooperation, group agreement or mistake-making. The instructor has more freedom to apply different concepts in different ways, so that students do not only "go to class", but actually participate.

This project can evolve in different and heterogeneous ways, being a base framework for potentially useful application. Some ideas for future work are the following:

- Internacionalization.
- Development of a more powerful editor, so that an appropriate project management can be achieved. All of the software project's components could reside in a server computer, from where all the users could have access, and an editor with a hierarchical browser could show all the project files and directories.
- Multimedia applications could be plugged in, such as videoconference, for geographically-disperse courses.
- A very simple improvement could be the ability of sending e-mails from COCO.
- Integrating COCO with other GroupKit applications, such as Text Chat.
- Adding new patterns for debugging . Including an error debugging pattern through the GNU GDB could be a great possibility.
- Syntax highlighting.
- Local or remote store of profile and preferentes information.

REFERENCES

ROSEMAN, M., GREENBERG, S. 1997. *Building Groupware with GroupKit.* M. Harrison (Ed.) Tcl/Tk Tools*,* p535-564, O'Reilly Press.

ROSEMAN, M., GREENBERG, S. 1996. *Building Real Time Groupware with GroupKit, A Groupware Toolkit.* ACM Transactions on Computer Human Interaction, 3(1), p66-106, ACM Press.

SCHUCKMANN, C., KIRCHNER, L., SCHMMER, J., HAAKE, J. M. 1996. *Designing object-oriented synchronous groupware with COAST*. In Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work (CSCW'96), ACM Press, pp. 30--38, 1996.

KUUTTI, K. 1996. *Activity Theory as a Potential Framework for Human-Computer Interaction Research.* Context and Consciousness. Activity Theory and Human-Computer Interaction, Nardi, B.A. (Editor) MIT Press, pp 17-44.

BELLANY, R. K. E. 1996. *Designing Educational Technology: Computer-Mediated Change.* Context and Consciousness. Activity Theory and Human-Computer Interaction, Nardi, B.A. (Editor) MIT Press, pp 123-146.

BARROS, B., VERDEJO, M. F. 2001. *Aprendizaje colaborativo. Marco teórico y tecnológico*. UNED OpenCoast. Information available from web: <URL: www.opencoast.org>

GroupKit. Information available from web: <URL: www.groupkit.org and www.cpsc.ucalgary.ca/grouplab/projects/Groupkit.html>

Java Shared Data Toolkit. Information available from web: <URL: java.sun.com/products/java-media/jsdt>