

# COLLABORATIVE LEARNING TOOLS AS PART OF AN OPEN ARCHITECTURE

Marcelo Cohen, Daniela Remião de Macedo, Patrícia Augustin Jaques and Michael da Costa Mora<sup>1</sup>

**Abstract** - In order to build a virtual university, one of the purposes of the Campus Global project at PUCRS is to study and develop tools for distance learning environments. This paper presents the development of two of these tools, namely the whiteboard and chat tools. They are designed to fit the specifications of the open architecture proposed in the project, developed as independent modules. The entire system was developed in the Java language, thus providing excellent portability.

## The Virtual University Model and Architecture

The Campus Global project is a partnership between Faculdade de Informática at Catholic University of Rio Grande do Sul (PUCRS) and IBM. The project aims at developing models, methodologies and tools that enable the construction of Virtual Universities (VU) supported by the new information technology under development nowadays.

The VU model defined by the project assumes that the physical structure that provides resource to implement, maintain and access a Virtual Campus, which is the "place" where the Virtual University activities occur. Notice that the Virtual University that is defined here is complementary to the existing Physical Universities, not as a replacement. The academic community has, therefore, new possibilities as to how perform their activities.

The VU model enables the participants to engage in activities that were not possible due to limitations of time or the necessity of being physically present in the same location. Also, it allows aggregating new resources and methods to the activities performed in the traditional way, improving its quality. Notice that, with respect to these qualitative aspects, our VU model emphasizes the need to provide a high degree of interactivity among the participants, and between the participants and the environment.

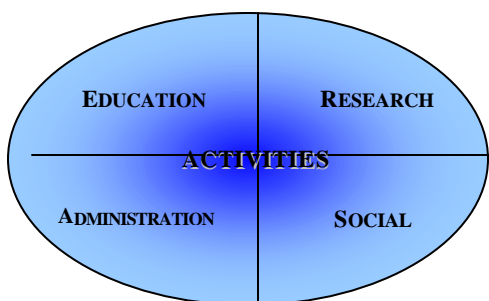


Figure 1: The Four Dimensions of Academic Activities.

Taking all these aspects into account, it is our view that every activity that occurs in this academic environment should be inserted in, at least, one of the four dimensions (and, very often, in more than one) described in figure 1.

An activity is a set of actions with a specific goal, where those four dimensions may be (and, in general, are) present. Naturally, each activity, depending on its nature, is more predominant in one of those dimensions. Each activity is developed through cooperative interaction among the participants at an appropriate environment. This environment is characterized by a set of resources that are necessary to perform that activity. These resources may be participants or actors (human resources), software and hardware tools (computational resources), information produced and consumed during the activity (information resource) and methodological resources.

Therefore, in our UV model the notion of environment is more covering than simple computational infrastructure, and may be applied to both the virtual and the more traditional physical scenarios. Notice, nevertheless, that the purpose of this model is to be a basis for virtual environments. Figure 2 shows the VU model characterized by the interaction of those different activities, and performed in virtual environments formed by human, methodological, information and computational resources.

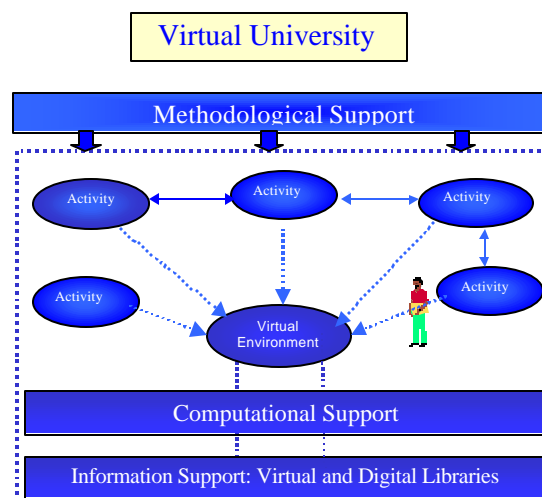


Figure 2: The Virtual University Model

The collaborative learning tools described in this paper are part of the computational resources. They are developed

<sup>1</sup> Campus Global - Faculty of Informatics, PUCRS – Brazil

following the Open Architecture Model [3], where there is not a monolithic environment, but a set of tools following common development policies, and with a common network interface.

### Collaborative Learning Tools

For a distance course to be considered well structured, it must supply a teaching and learning environment [6]. That environment should fill the teacher's needs, aiding in the creation of the course, as well to make easier the students' participation and motivating them to do cooperation. According to [1], cooperative learning is a teaching strategy where small groups, each one with students of different skill levels, use a variety of learning activities to improve the understanding of a subject. Each student of the group is responsible not only for learning what it is being taught, but also for helping his/her colleague.

When this collaboration happens in a distance teaching environment, supported by the Internet technology, the GroupWare notion can be used well, which makes to appear a new subdivision in the area: Computer Supported Collaborative Learning (CSCL).

[6] points out that for a GroupWare system to be used in distance education then it should supply some **tools** that allow the students' participation and collaboration among the colleagues of the course.

Moreover, the construction of our tool had the contribution of teachers of virtual classes at Pontificia Universidade Católica do Rio Grande do Sul (PUCRS). The distance education classes at PUCRS follow a methodology developed at the Campus Global project, originally applied by [4]. According to this methodology, the student is the central personage of the learning and the teacher has the mediator role, as assisting the student in his/her learning as well as stimulating and monitoring the interactions between the students.

At PUCRS, a student when registering in one class (if this class can happen virtually), can choose between traditional classes - as it happens normally - or virtual classes. In the virtual modality, the first class occurs in a traditional way, in order for the teacher to present the methodology and/or resources to be used (software, hardware, etc). All remaining classes are virtual and happen in two weekly meetings by chat (synchronous) - in the normal schedule of the traditional class.

In the design part, the research group has watched some virtual classes, which allowed to understand the working of the classes, how they were presented, which tools and resources were used. Moreover, some meetings with the educators had been carried with the intention to know which tools and technologies the teachers needed in their classes. This feedback happened during all the system's development, such as in the part of design, supplying the indications of essential features, as well as in the phase of development, in returning opinions, remarks and

improvements that could be made in the implemented prototype.

In this way, the Campus Global project proposes an integrated system of whiteboard and chat tools to help teachers and students in the execution of collaborative classes and activities. This system is based on a client-server architecture, where all the functions of the application can be divided in two programs: the client program that is executed in the user's computer and the server program that it is executed in the host. Thus the client program provides an interaction with the user through data input and presentation of the results. It is able to communicate with the server program, sending requests and receiving responses.

Four modules compose the proposed system: control, chat, whiteboard and server. The control, chat and whiteboard modules are the client programs of the application, while the server module is the server program.

In the next sections each module will be explained in full detail.

### Control and Server Modules

The Control Module manages the messages that are received and sent by the users in the Whiteboard and Chat. So the control module classifies the messages in three types:

- Messages intended for the Chat application
- Messages intended for the Whiteboard application
- Control Messages

The messages destined to Chat and Whiteboard are stored in a buffer, waiting for the corresponding application to read them. The control messages are interpreted internally inside the control module and shown in a message window. Typical control messages include logon notification, user connection, disconnection, errors, etc.

The control module has an interface that allows the user to connect to the server for a session (Figure 3). In this same interface the user can see who else is connected, disconnect and initialize the Whiteboard and Chat programs.

All users' messages are transmitted to the server that will then broadcast them (Figure 4). For communication between the client software and the server were used primitives of the communication protocol Internet Relay Chat [5], in the case of Chat messages. In addition, primitives were created for the Whiteboard messages.

The communication of the control module with the server is achieved through sockets. The control module will make a connection request with the server in the port number supplied by the user. All the messages are sent and received through this connection port.

In addition, the server stores a log of all the messages of a session. In this way, the user that connects later can

visualize the messages and pictures changed previously by the others.



Figure 3: Control Module Interface

to the chat, as well as messages informing when a user enters or leaves)

- Bottommost area: user message (where the user types the message to be sent to other users connected to the Chat tool).

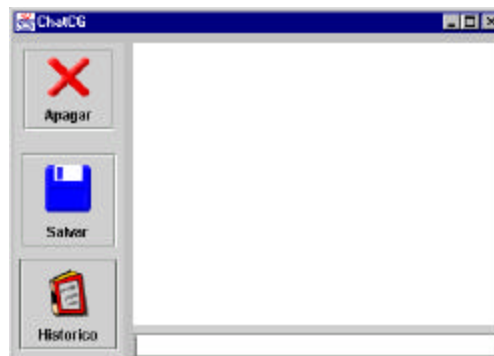


Figure 5: The Chat Tool interface

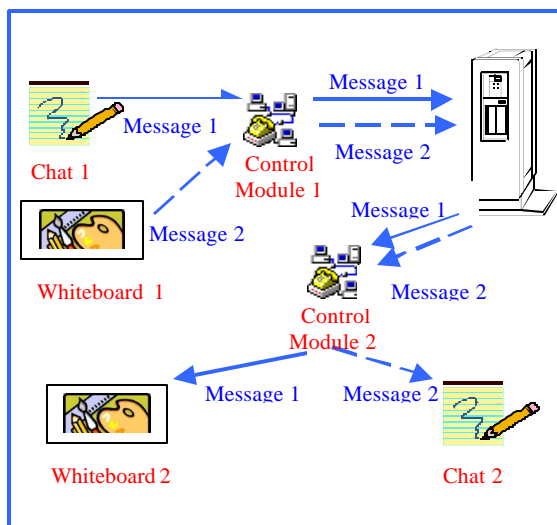


Figure 4: Diagram of connections between Server and Control Modules.

## The Chat Tool

As all the participants in a virtual class are physically apart from each other, the main form of communication is through written text, by the use of chat tools. The system provides a Java implementation of a chat tool, with additional features and following the open architecture concept.

Figure 5 shows the chat tool interface. Following the areas of this screen are described.

- Leftmost area: operations (erase, save, history)
- Center area: message visualization area (where are displayed all the messages sent by the users connected

An important operation available at the chat tool is the **history** of the chat session. It can be useful when a student have arrived late to the virtual class. Choosing this operation, the student receives the history of all messages the users had changed while he/she was not connected to the chat tool.

Also is possible to save a chat session to the disk. This allows the students to have an additional reference material about the classes.

The Chat tool allows the user to send both public messages (sent to all users connected the system) and private messages (sent just to some specific users). To send private messages the user needs to choose at the Users List on Control Module, the names of the users he/she wants to send the message.

### Chat Tool implementation

The client exchanges information with the server following a simple protocol, presented as a text line. There are three types of messages:

- Log request: *sendlogchat*  
This message requests the server to send the client a log of all messages changes by the users connected to the chat.
- Users connected request: *rpl\_namreply*  
This message is send by the server to all clients every time a client connects or disconnected to the server.
- Message sending: *primsg receiver, {receiver}: message*  
Where *receiver* can be just one user name or a list of user names or a channel, and *message* is the message to be sent. The client sends this message to the server in order to deliver it to other users in the chat. The server

sends it to one or more clients to inform that a user has entered or left the system.

### The Whiteboard Tool

Today the use of chat tools is common: it's pretty simple to use such a tool, but it's not easy to present a class with only written text. The whiteboard tool allows a more advanced interaction than just plain text: a shared drawing area. This area is controlled by a client application, used by everyone.

There are many ways to picture a whiteboard: one can imagine it simply as a screen where the teacher presents some information in a controlled fashion. In this context, the whiteboard is just a slide projector and the students don't have the right to change what it's projecting.

But one can also assume it as a truly shared space, where everyone can work together. In practice, when we have a larger collaborative group of people - a virtual class for instance - these tools generate heavy network traffic, as all students will want to use it simultaneously but no one will be able to.

This problem leads to the need of a control mechanism; in order to make sure that only selected people will have the rights to change the whiteboard's contents. This can be solved by many ways: we can provide the teacher with the ability to grant and take control to one or to a group of students. In a virtual class this seems to be the safest way, but in a different context can be fairly obtrusive. An example is where we have a smaller class, or when we like to work on a collaborative operation, where the entire group should have access to the whiteboard.

The present implementation provides this control in a simplified manner, by the use of two passwords: one for the teacher and other for all students. When these passwords are used, only the teacher can change the whiteboard and give/take control to a student if desired. If the passwords are not used then everyone will have full access.

### Whiteboard concepts

The whiteboard was written with the concept that it should be easy to create and present information for the students. Thus, our main concern was how that information would be presented. We needed an easy way to represent not only textual data but images as well.

As almost all teachers in our faculty make extensive use of HTML pages to provide educational information for their students, this appeared as a natural choice. The idea is to previously prepare all needed background information, to be presented as HTML pages (although the system only supports a reduced set of all available HTML tags<sup>2</sup>).

In order for this system to work, all pages must be available at a web server (figure 6). All whiteboard clients fetch page data from that server. The server module

exchanges only object and control information with the clients, thus minimizing the traffic. In this way, all pages are loaded asynchronously, reducing the server load.

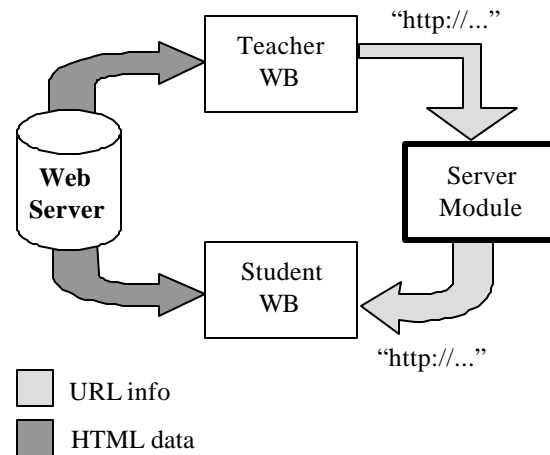


Figure 6: Data flow between Web Server, clients and Server Module.

A typical class would happen as follows: the teacher tells his/her students that everyone should connect to the system at a given time. At the time of the class, he/she connects and tells the server module that a new page should become available (by the page URL, which obviously he/she knows). At the same time, all students that are also connected receive a message informing that a new page is available and that page is then loaded by the whiteboard. Then the teacher can draw something over the page, make notes, etc, while also using the chat to explain the subject and ask/answer questions to the students. If the teacher wants, he/she can also grant permission for a student to use the whiteboard and contribute for the class. It is possible to add new pages at any time, just by sending the page URL.

In such a scenario, our goals of simplicity, easiness and interactivity would be definitely satisfied.

Figure 7 shows the whiteboard interface, explained as follows.

- Topmost area: list of already visited pages. This list grows when the teacher informs each page URL.
- Center area: current page, along with an object drawn over the HTML view (a square). This object is currently selected by the user (denoted by the square handles).
- Leftmost area: drawing tools (selection, lines, rectangles, ellipses, text)
- Bottommost area: object attributes (line width and style, fill type, color). When the text tool is in use, this area also shows the text attributes, (font, size, bold/italic, etc).

<sup>2</sup> Due to implementation constraints in the JDK

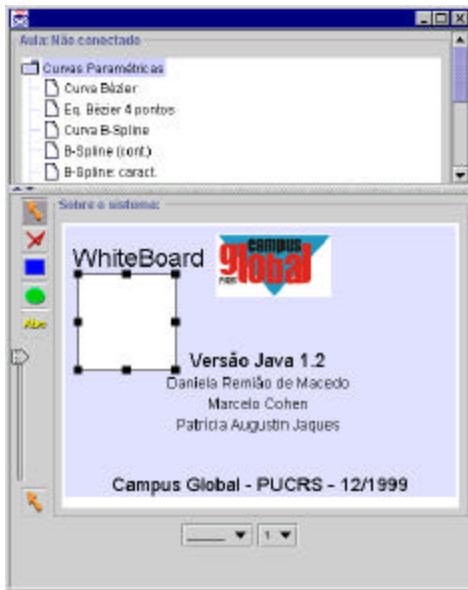


Figure 7 – The whiteboard interface

The list of visited pages (topmost area) is of particular importance. During the class a teacher may want to present the information one page at a time. This way he/she can ensure that the students will be following him/her closely. As stated early (whiteboard concepts), students will be able to view a page **only after the teacher sends the page URL to the server.**

### Whiteboard implementation

The whiteboard class was derived from *JEditorPane*, Sun's own class to provide HTML browsing and editing. We override the *paint()* method, let the superclass (*JEditorPane*) paint the canvas and then we paint the whiteboard objects over the page image. Transparency and fill effects were accomplished by use of Java2D features.

In order to prevent the superclass from selecting text when dragging with the mouse (which is a normal behavior), in the mouse move event we reset the selection to none.

As stated earlier, the control module handles all low-level communication. The whiteboard just have to call a method to send data to it.

The client exchanges information with the server following a simple protocol, presented as a text string. Two categories of messages exist, described as follows:

- Content: messages that allow changes in the content of the page, such as: inclusion of objects, attribute change, inclusion of a new page, etc.
- Control: messages responsible for log requests, blocking and unblocking of objects, etc.

The blocking process is very important. When the whiteboard is being used without passwords, everyone has full access to all objects. Therefore, there must be some control to prevent more than one person to change the same object at a time.

This control was implemented in the form of a blocked status: when a client wants to do something with an object, it first requests access to that object, sending a blocking message. If someone else isn't using the object, the server acknowledges the request and blocks the object. Then the client can change the object freely. After the update, the client must send an unblocking message. To prevent deadlocks if the client dies unexpectedly or disconnects, the server unlocks the currently locked object by it.

### Conclusion and Future Work

This paper presents an implementation of chat and whiteboard tools as an example of an open architecture to support distance learning through Internet.

This openness allows the easy development and integration of new modules according to the users' needs.

The system presented will be used in virtual classes of the Faculty of Informatics at PUCRS, where all meetings are currently realized by the use of common chat tools. The whiteboard and chat will be first validated as support learning tools during the classes. We believe these are practical tools that will greatly enhance class interaction.

Some non-implemented features remain, mainly at the control module. These include more robust user management, allowing the teacher to selectively grant and take rights from any student. Also new modules are planned supporting additional resources, such as intelligent agents to supervise interaction, video, audio and 3d visualization.

### References

- [1]. Balkcom, S. *Cooperative Learning: What Is It?* <http://www.ilt.columbia.edu/k12/livetext/docs/cooplern.html>. September 1998.
- [2]. Budny, D.D., Budny, D.D., "Counselor Tutorial Program (A Cooperative Learning Program for the High Risk Freshmen Engineering Courses)", *Journal of the Freshmen Year Experience*, 1994, Vol. 6, No. 1, pp. 29 - 52.
- [3]. Campus Global Team. Towards the Virtual University (in Portuguese). Technical Report. <http://www.cglobal.pucrs.br/bibdig/>
- [4]. Ferreira, Simone Nunes; Campos, Marcia Borba. CBP 2001: Uma experiência prática de sala de aula virtual nos cursos de graduação da PUCRS. URL: <http://www.cglobal.pucrs.br/bibdigital/>. Presented in Latin-Iberoamerican Congress On Operations Research, 9, 1998, Buenos Aires.
- [5]. Oikarinen, J.; Reed, D. *Request for Comments: 1459. Internet Relay Chat Protocol*. [http://www.irc.org.au/irc\\_help/irchelp/rfc/rfc1459.txt](http://www.irc.org.au/irc_help/irchelp/rfc/rfc1459.txt). May 1999.
- [6]. Palme, J. *GroupWare tools to support distance education*. <http://www.dsv.su.se/~jpalme/distance-education/mmm-tools.htm>. June 1997.