Dealing with the Students' Attention Problem in Computer Supported Face-to-Face Lecturing

Nelson Baloian[1], José A. Pino[1], H. Ulrich Hoppe[2]
[1]Department of Computer Science, University of Chile, Chile // nbaloian@dcc.uchile.cl // jpino@dcc.uchile.cl
[2]Dept. of Computer Science and Applied Cognitive Science, University of Duisburg-Essen, Germany // hoppe@collide.info

ABSTRACT

This paper addresses issues of using computer technology to support in-classroom teaching and learning regarding one of the most frequent problems in such scenarios: the students' attention. First, it presents the distraction and focus of attention problems that arise while using computer technology with off-the-shelf software for supporting different activities which take place in usual lectures. Examples of these activities are presentations of learning material, discussion, individual and collaborative problem solving, and difficulties of switching from one to another. A solution to this problem is an integrated approach with a particular implementation under the notion of "CiC". The implemented system attempts to reduce the number of interactions needed to switch from one activity to another as well as to reduce the cognitive load for both teacher and students. The usage of both this system and standard software are compared and research results are presented. Results show the CiC is useful to solve the problem.

Keywords

Improving classroom teaching; teaching/learning strategies; architectures for educational technology system; collaborative learning; face-to-face learning

Introduction

Traditionally, most research efforts in Computer Supported Cooperative Learning (CSCL) have been oriented towards developing distance learning scenarios (for a good covering of this subject see Berge, 1995). However, most learning activities still take place in a face-to-face learning scenario. It is still an open question if future learning will take place in the more de-personalized way of "virtual learning" or continue in the traditional style. Accepting that learning in classrooms under the responsibility of a teacher is still dominant in many places, we can ask what could be gained by using new information and communication technologies inside classrooms. Several systems and methodologies supporting in-classroom learning scenarios have been developed taking advantage of innovative hardware and software (Marjanovic, 1999).

In distance learning scenarios, technology is mainly used to restore the missing communication channels among people. Now, in both distance and in-classroom scenarios, there is a challenge to advantageously use rich multimedia material for collaborative learning (Mulder, 1997). Many authors claim this kind of material is an important resource that can significantly improve the learning experience (Sparks, 1999; Retalis, 1997). There is common agreement that computers should be used to enhance learning processes and practices. Of course, the mere use of computers in the classroom does not necessarily and automatically lead us to better learning as compared to traditional teaching/learning environments without computers. Students may be distracted by browsing through the Internet, or by sending messages to classmates, etc. (Scardamalia, 1996). Furthermore, teachers must make an effort to prepare courseware material to be used in the classroom (Santoro, 2005), either by getting and adapting material from various sources or developing it by themselves. Our initial trial in this respect turned out to be disappointing, as presented in the subsequent section.

Nevertheless, there is a shared belief that the computer technology, when appropriately used, should provide better results than unsupported teaching/learning. The issues boil down to how to "use computer technology properly" and how to measure research results. The purpose of this work is to deal with the first problem. Rather simple problems such as using suitable hardware are less important than problems presented by the use of adequate software. As described in Section "Characterizing the attention problem", we began using off-the-shelf software, but this turned out to be unsatisfactory. Consequently, we developed a comprehensive system with our own software.

The rest of the paper is organized as follows. Next section reviews the relevant literature in the subject. Section "Characterizing the attention problem" introduces the problem of using computer technology in the classroom and specifically, the distraction problems caused by switching from one computer-supported teaching/learning activity to another one. Section "An Integrated Software approach to implement CiC" presents our own experience in solving this problem in an academic setting. Section "Technical Evaluation of the Proposal" compares this solution with a simple off-the-shelf software approach. Section "Experiments with CiC" describes an experiment to empirically test the solution. The last section discusses both options mentioned above and concludes our argument.

Previous work

In computer-supported distance education, students have access to many different learning resources. They can get much information from the Internet, in addition to the learning material provided by their instructors. In order to facilitate the knowledge acquisition, they can also do practical work, like solving problems. On the other hand, in face-to-face settings, students have the benefit of easily talking to the instructor. This is a great advantage when compared to distance learning.

In higher education the most common situations are lecture-type classes. Thereafter, laboratory or homework sessions may take place as complementary activities. Let us take a college computer science introductory course as an example. The teacher, in a traditional methodology, will introduce students to a certain algorithm presenting the basic idea using a projector and a screen. The teacher may show a program implementing the algorithm and display a picture illustrating its basic structure and function. The teacher then turns to analyze the efficiency of the algorithm and/or other algorithms solving the same problem. The students must exercise after the lecture to internalize the concepts and apprehend the practical aspects of the algorithm.

This is not a typical way of teaching/learning in high school. Students must understand the concepts immediately after the idea is presented to them. They learn the basic ideas *while* solving a problem. The computer should support several educational activities within the class: lecture presentation, animations, movies, drill activities, tests, simulations, etc. All these activities should be done with the same computer. Some of the unexpected activities may arise on the fly; e.g., a concept which is not being well understood needs to be presented with some examples, additional discussion, etc. This methodology could also be applied in higher education in a natural way.

The Computer-integrated-Classroom (CiC) is a concept proposed more than a decade ago as a new way to properly use computer technology inside the classroom (Hoppe, 1993). The CiC was a pedagogically motivated answer to the push to incorporate technology within the classroom. The available technology at the time included computers for both teacher and students, in addition to electronic blackboards, electronic pointers, early mobile devices, sensors, cameras, etc. All this hardware could be connected by a local area network and to the Internet. The CiC proposes solutions to problems typically surrounding technology supported teaching/learning in a face-to-face situation.

In an updated CiC setting, the teacher presents learning material (multimedia documents) on an electronic blackboard and every student has a notebook or PDA to work on this material. A wireless LAN connects students' computers with the electronic blackboard, and there is a Web server which provides and stores learning material. A CiC should also have server functions which help teacher and students to retrieve, manage and share learning material, so that the face-to-face teaching situation becomes alive.

The COSOFT project (Baloian, 1995; Baloian, 2000) was an implementation of the CiC concept. The computer-integrated classroom combines positive aspects of the classical chalkboard approach, particularly its flexibility in the spontaneous elaboration of ideas, with the potential of modern networked multimedia. The value added lies on the avoidance of discontinuities in representations ("media breaks"), e.g., when the solution that a student has individually elaborated on is copied again by hand to the chalkboard. In the CiC, re-use and synchronous and asynchronous exchange of material are easy. The full spectrum of basic representations, ranging from freehand input to sophisticated simulation and animation is available. Another implementation of the CiC idea has been put into practice in the European NIMIS project (Hoppe, 2000; Lingnau 2002; Tewissen, 2001). Target users for this version were young schoolchildren (4-8 years old) and it was aimed to develop children's reading/writing skills based on the "reading through writing" methodology (Reichen, 1991). In the NIMIS project a classroom was created using interactive whiteboard and pen-based tablets embedded in the pupils' desks in a networked environment with

educationally motivated groupware functions. Other approaches to computer-equipped face-to-face classrooms have been developed in the Hypercourse project (Norman, 1994) and in Taiwan secondary schools (Wu, 2002).

The CiC approach differs from the traditional Leaning Management Systems (LMS) also known as courseware tools. Examples of LMS are: WebCT, developed originally at the University of British Columbia (2007), now a commercial product; Scholar360 (2007); Angel LMS (2007); TopClass (WBTSystems, 2007); Luvit (Lund University, 2007); and ICESEE (Ochoa, 2003). They focus primarily on the creation and/or management of the learning material (courseware) and the relation between this material and the different courses. Most of them also offer functionalities to manage the relations among students and courses by giving them access to the corresponding learning material. It is also very common to find that they offer discussion panels which can work synchronously (like a chat party) or asynchronously (like a news board) (Ochoa, 2002). Most of them are not designed to be used in the face-to-face classroom situation but to support a distributed, asynchronous learning scenario. A CiC is not oriented to support the creation and management of the learning content but to support its usage in a face-to-face teaching/learning situation trying to overcome the problems that arise in this kind of scenarios. Our experience has shown us one of the major problems in using computer technology in the classroom is the transition between sequential stages of a lecture (e.g. presentation, individual work, collaborative work, etc.) and how to use the material generated in one stage of the lecture in the following ones. Next chapter will describe this problem in detail.

Characterizing the attention problem

One of the authors has taught an introductory Computer Science course about computer programming to university freshmen for several years. The course has about 80 students and there is a team of Teaching Assistants for the laboratory sessions and for grading assignments and tests. The teacher uses the following teaching style: he presents a problem to the students, he develops a solution using the programming resources the students have learnt up to that moment, and then, he introduces new programming elements which improve the solution. During the second semester of 2001, he decided to experiment using the CiC approach. Other parallel sections of the same course were taught by other instructors using traditional technology (lectures with projector and screen).

Since this was an experiment, the teacher tried to start in a parsimonious way. He just used his notebook and a smart board. Instead of writing the programs by himself and discussing methods with the students, he would write the program with contributions from the students, he would debug the program on line and he would show execution of the program with sample data. He will also show animations, he will search the Internet for some data, etc. He expected the students would appreciate the additional time he spent to prepare such classes and they should have learnt more on the subject.

The results, however, were disappointing. The students did not like the course. In the class evaluation, they rated the instructor just *under* the approval minimum (3.7 points in a 1.0 to 7.0 scale, with 4.0 being the minimum approval grade). This was in sharp contrast with the rates the same instructor obtained in another course he taught (6.1) in the same semester, and with the rates he obtained for the same course in previous semesters (over 6.0).

Wanting to know why, we analyzed students' comments in the survey. The standard survey applied by the university is anonymous and it includes both ratings and free-format comments. For the second semester of 2001, 56 students answered the survey, and from those, 42 included comments. A study of the comments showed that 26 of them referred negatively to the way the learning material was presented in class. Representative comments are the following ones: "The teacher simply spends too much time trying to show us things", "We get easily distracted in class. I started to take some books with me and began to read them there. Finally I quit attending classes", and "Why does it take so much time to move from one program to another?"

Their message was clear. The instructor was using off-the-shelf software to teach the class: Microsoft PowerPoint to make presentations, Mozilla Firefox browser to navigate the Web, a Java compiler and a text editor. Each time the instructor needed to switch to a new program, he had to do several mouse movements, wait for the activation of the program, etc. He obviously thought these times were negligible, but they were not: the students could not keep concentrated while the teacher was doing the software transitions in his notebook although each program was well suited to the specific purpose activity. Spending too much time manipulating programs, like typing long commands or queries, or searching for files, breaks the dynamic flow of the lecture and distracts the audience. It has already

been identified that the most frequent teachers' mistake is to turn their backs to the audience while working with chalkboards and overhead projection (Grosvenor, 1999).

The attention issue in general has already been studied. Perhaps the most accepted theory about attention in general is Treisman's Feature Integration Theory (Treisman, 1980). It states that attentive selection is guided by both input and cognition. The issue of the user's attention on graphical computer interfaces has been studied by Vertegaal (Vertegaal, 2002). He states that the rationale for the design of windowing systems is to allow users to focus on the task with the highest priority in the context of other tasks with lower priority. In a typical graphical user interface, windows relevant to the present task should occupy the display space on which the user is currently fixating. Low priority tasks should occupy peripheral vision. Although both authors refer their findings to the attention on the individual person, their findings can be applied to multiple users trying to concentrate their attention on a single large display: they affirm the fact that the attention will be influenced by the external information a person receives and as a consequence, this information should focus on the subject the teacher wants the students to concentrate. Research on attention in the collaborative context is very recent, but the preliminary experimental results indicate that the group performance is influenced by attentional phenomena (Ferreira, 2007).

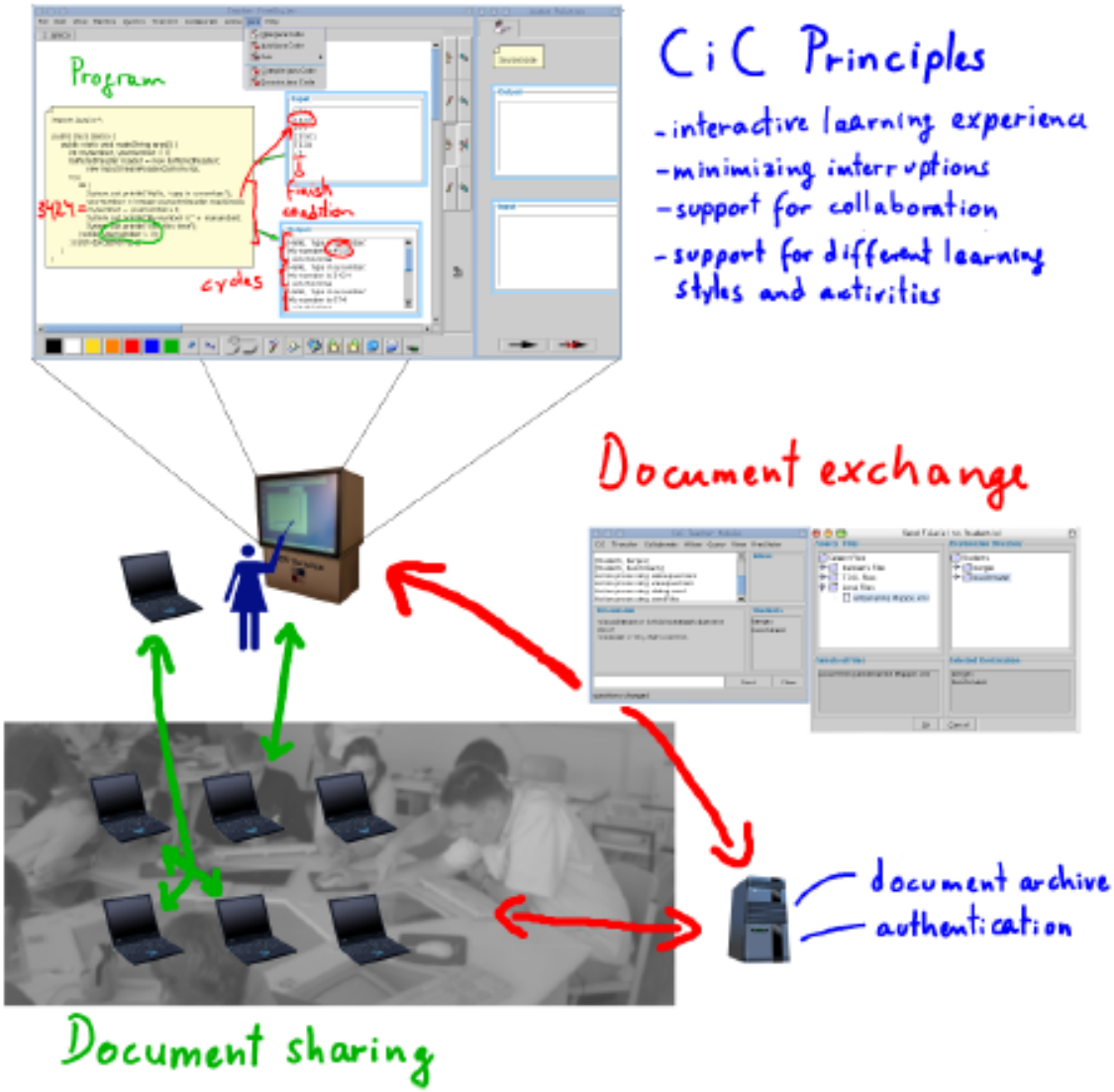

Figure 1: Computer-integrated Classroom (CiC)

New human-computer interaction problems arise in the electronic classroom situation due to the new physical human-computer arrangement. These problems should be addressed with a new approach. The first non-standard interaction is the teacher-blackboard one. It seems the traditional location of menus and messages does not fit well this situation: menu bars at the top of the blackboard may not be suitable for every teacher and error messages should be displayed in a very discreet way. Furthermore, having multiple windows opened on the blackboard may not only confuse the teacher but also the students who should also be regarded as blackboard users. These problems have been previously identified under the notions of "students' disorientation" (Colazzo, 1995) or of the "focus of attention" problem (Hoppe, 1993).

Other lecturers using the electronic classroom complained they had to run back and forth between the e-board, keyboard and mouse. The students' attention was distracted because of the teacher's attitude. This may be solved by enabling all the necessary interaction on the interactive whiteboard as the only input and output device. Various kinds of applications, e.g. to show sketches or renderings, perform live applications and simulations, or to collect and manage student input information, should be integrated within a consistent and not distracting software environment.

An Integrated Software approach to implement CiC

As we can derive from the above discussion, the most important issue in our experiment was to keep the focus of attention of a computer supported lecture audience. For this purpose, using the CiC should require less time to complete the sequence of activities, especially during the transition between various types of activities. The CiC implementation described in this work is called CiCv2, which consists of three tightly coupled modules. One of them is the central repository, available from inside and outside the classroom sessions containing the learning material for the lecture, as well as the information necessary for the management of the system. The other modules are the ones used by teacher and students, respectively. Figure 1 shows a diagram explaining the principles behind the CiC and the functionalities provided to achieve them.

Repository

As mentioned above, it is important that the teachers have to spend little time performing secondary activities, such as searching for an appropriate place to store and retrieve learning material, or choosing a correct file name in order to ease its retrieval. We decided to provide a central repository with a management service which will choose the correct location and sometimes the appropriate name for the files teacher and students need to store and retrieve. The idea is that the system defines the place and file name using the information of its usage context, e.g., the course, the purpose (assignment, homework, presentation material), author and session in which it was created. Thus, the teacher does not need to search for a certain directory or think and type filenames in many cases. Also, students benefit from the repository and the mechanism it uses for storing and retrieving files. Therefore, the repository must store the learning material the teacher has previously prepared for the lecture, the learning material the students have for themselves, as well as their homework and assignments. It also stores the administrative information about teachers, lectures and students. It keeps information about the "sessions" of current lectures. A session is initiated by the lecturer and students join it afterwards. All this information plays a key role in making swift transitions between the different lecture phases. Each user can access the files available on the repository at any time, without the need for a session to exist.

Student and Teacher Applications

These applications implement the functionalities for retrieving material from the repository and the necessary communication between teacher and students including learning material transfers. They share many elements, but the teacher's interface has some additional functionality, mainly related to allow the sharing of material among students.

Teacher and students have to start their applications on their computers and register to initiate a working session. A small application interface eases the usage of other programs simultaneously during the lecture. As additional functionality is required, new parts of the interface are shown and hidden. This graphical user interface (GUI)

includes several elements (Figure 2). The menu bar allows the user to take most of the actions for manipulating and sharing learning content. The inbox element, which is not always visible, indicates events that are important or need an action. Examples of these events are when a student receives material from the teacher or when a collaborative session has started.

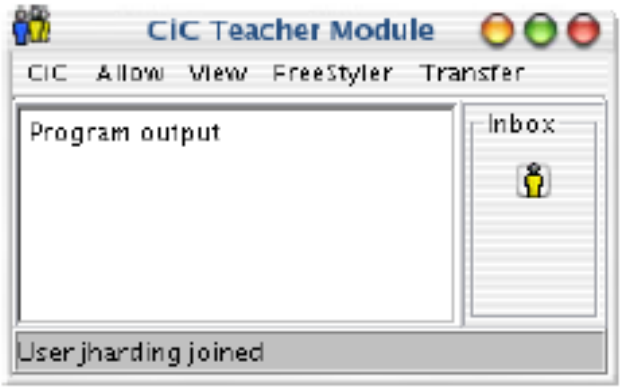

Figure 2: The teacher's application interface

File Transfers

The Student and Teacher applications implement several kinds of file transfer actions. A key design principle in CiC to achieve a swift transition between the various activities of the face-to-face lecture was to identify the most frequent actions requiring file transfer and to automate them. We identified the following ones for the teacher:

- Distribute Assignment: A file is selected from a location that may include the repository or the local working directory. This file is sent to each student participating in the active session, and they are notified of its availability.
- Collect Assignment: After the students have worked on their assignments, the teacher uses this action to save a copy of the file each student has been working on into a location in the repository. Those copies will not be student-modifiable anymore.
- Distribute Homework: A file is selected from a location that may include the repository or the local working directory. This file is copied to a directory inside the student home directory on the repository, for those students attending the course.
- Collect Homework: The teacher retrieves the files delivered by the students using the "Commit Homework" action (see below).

Actions available for students are:

- Process Homework: A student connects to the repository and searches for homework distributed by the teacher using the "Distribute Homework" action. The homework gets copied to the local working directory and the student can manipulate the file.
- Commit Homework: After finishing homework, the student would use this action to send the file back to the repository. Then, the teacher will collect these files at a predefined time.
- Perhaps the most important consequence of having identified and implemented these functions is the teacher does not have to remember "physical locations" of the files. The system has already assigned directories and names for the various types of files and for each student. These directories and names are automatically retrieved.

There are also two commands available to distribute and collect files specifying the sources and targets. All these actions require just to select a menu option or to press a button in order to execute them. They are easily accessible from the GUI.

Documents in the CiC

CiCv2 can handle any type of file transfer, but in order to take the most advantage of the CiC situation it is necessary to allow for the synchronized use of the learning material inside the classroom. We could use general-purpose

software to achieve this, like Virtual Network Computing (VNC) or Microsoft NetMeeting. However, that kind of software achieves the synchronizing of applications based on the view (window sharing) which generates much network traffic. This solution is almost unfeasible when there are more than 10 users working simultaneously. We opted for a more specific but not less useful solution using special software for handling interactive documents with a software called *FreeStyler* (Hoppe, 2002). Interactive documents are electronic documents which have interactive elements combined with freehand annotations. The documents may include a workflow description about how and when they should be distributed to and collected from the students, where they should be stored, etc. This allows the system to offer useful default alternatives to the user, minimizing the input required to perform each task and keeping the focus on the session. FreeStyler files are stored in XML format. In addition to the advantages of a standard format in general, XML makes it easy to retrieve information.

Using a MatchMaker server (Jansen, 2003), it is possible to share a page of a FreeStyler document across several applications. When a page is coupled, every user participating in the session receives a coupled copy of the document in the personal computer. This means a change on the document made by any user sharing the same MatchMaker session is replicated to the other applications, thus all connected users can see the changes almost instantly. This is an important resource for collaboration inside a classroom. Coupling and decoupling of documents can be performed dynamically and involving selected users of the whole group. This means the teacher can build different working groups independently from each other by coupling different users in different coupled sessions. Moreover, since every student has his/her own copy of the document, he/she can work offline alone after the coupled session. Because of this, the coupling mechanism can also be used to distribute homework or assignments to the students.

A very powerful feature of Freestyler is the usage of ad-hoc pluggable modules called *palettes* to define new functionalities. Each palette contains elements called nodes that can be placed in the documents. A palette contains also various types of edges that connect the nodes. In this way, different models can be used for different learning subjects or complexity levels. A palette for supporting the teaching and learning of Java programming was developed for our experiment. The Java palette contains several nodes and one link type (Figure 3). The nodes were designed to allow to present, comment and demonstrate (run) one or more Java programs in the same document. The node types are the following ones:

- **Java Code Node**: a node which contains, shows and runs a Java program.
- **Program Input Node**: a node which contains (or receives) and shows the standard input for a Java program.
- **Program Output Node**: a node which shows the standard output of a program.
- **Class Node:** a node intended to include external class files or jars during the compilation and execution of the program.
- **Text Node:** It contains formatted text following http rules. This node type was created in order to add some explanations to the programs being compiled and executed.

Associating an Output or Input Node to a Program node requires drawing a link between them. Association of an Output Node to a Program Node is accomplished by drawing an arrow from the second node to the first one. Figure 3 shows a Freestyler document with the Java palette. The palette is shown at the right hand side of the screen and contains the three kinds of nodes which can be added to the document by drag-and-drop operations. There is an icon for creating links and another one for deleting them. The document of the example shows a Program node connected to both an Input and an Output Node. The palette also defines some functionality, which appears on a pull down menu. These functionalities are the following ones:

- **Create Java Code**: this is an alternative way to create a Java node, which may be more comfortable for some electronic boards, where drag-and-drop operations may be not so easy to perform (e.g., on a SmartBoard). Nodes are created without any content.
- **Add Java Code**: it creates a Java Program Node with contents taken from a Java program file.
- **Edit**: it changes the font for displaying the Java code, the input and output.
- **Compile**: it compiles the code contained in a selected Java Program Node.
- **Execute**: it executes the code of an already compiled Java Program Node.

Using the CiC environment and the Java palette, teachers can show how to write a Java program, run it and show its output in a consecutive breakless sequence of steps. Since a Freestyler document can include handwriting and elements from other palettes (e.g., a discussion palette), a pedagogically meaningful document can be created during

the lessons and stored as learning material for the course. Documents containing program fragments or programs with errors can be distributed to the students for completion or correction as homework or assignment, by using the "distribute" functionality of the CiC environment. The teacher can enable a student to show her solution to a certain problem by establishing a coupling session and synchronizing the documents of this student with the document on the electronic board.

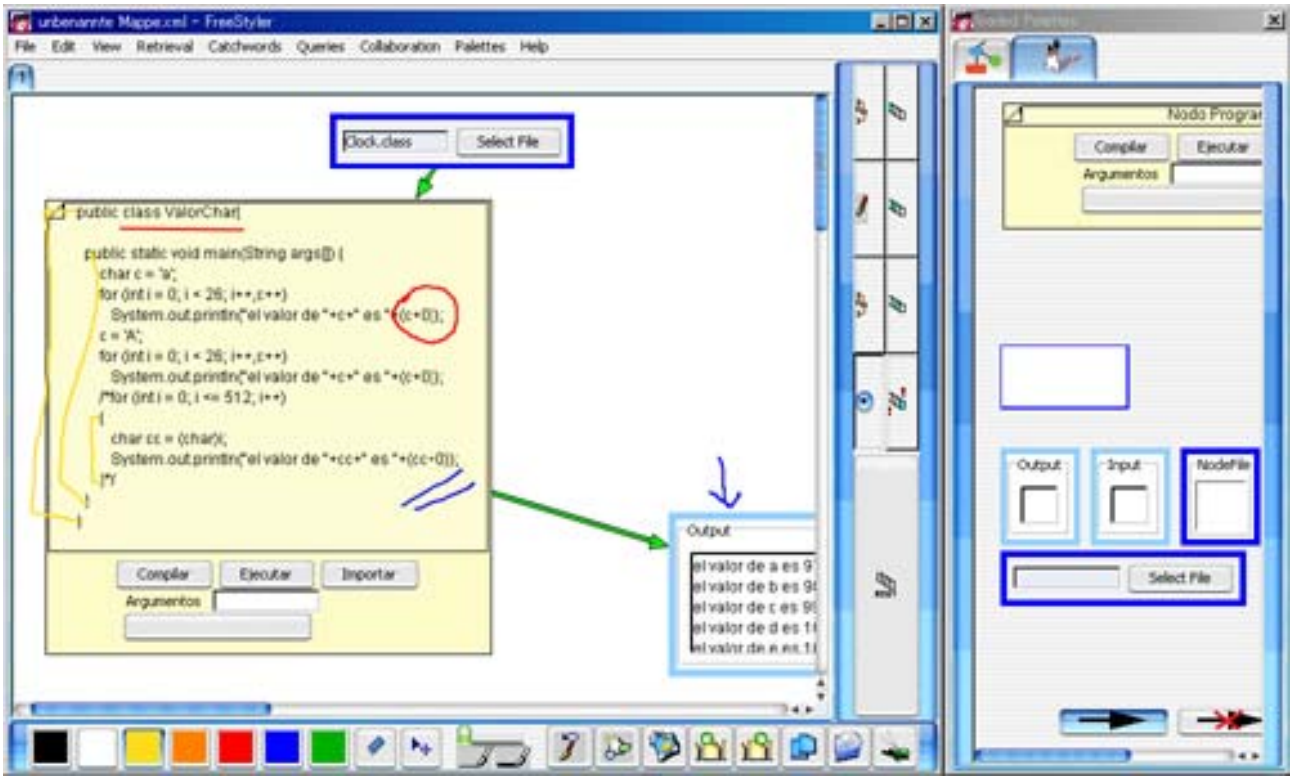

Figure 3. Screenshot of Java FreeStyler working with CiC. The framework is at the left, the palette with the nodes is at the right

Technical Evaluation of the Proposal

Evaluation of this proposal is made by comparing an in-classroom, face-to-face teaching/learning scenario supported by hardware and standard software and the same situation with the same hardware but supported by the CiC software. The scenarios we will describe correspond to actual situations experienced by one of the authors while teaching a Java computer programming introductory course as mentioned in section "Characterizing the attention problem". The methodology used for the lecture is the same for both cases. The situation is described as a sequence of five learning activities consisting of: 1) presentation of the learning material by the teacher, 2) individual work by the students, 3) review of the solutions with discussion, 4) collaborative work in small groups of students, and 5) final discussion and summary. The four transitions between these activities are important to be studied.

The lecture content is about implementing a *set class* with the Java programming language. The set allows storing, deleting and retrieving elements in different groups as well as testing if a given element belongs to a certain group. The set class concept implementation requires a data structure capable of containing a collection of elements. The concept of arrays appears then naturally. Table 1 presents teaching/learning activities in the second column. The third column describes how to perform them with standard software and the last column how to do them using the CiC.

A close analysis of the advantage of using CiC instead of off-the-shelf software needs to show that using CiC reduces the time the teacher needs to spend for performing the necessary actions. Table 2 shows the required fine grain operations to make the transitions between the classroom activities of Table 1. It is easy to see CiC requires fewer operations than the standard approach and each of its operations takes less time than the corresponding ones

for the standard approach. Moreover, the operations marked with a star mean the starting of a new program. The teacher can, of course, start programs before the beginning of the class. However, this is not a good strategy because it means many windows will have to be managed during the class and thus, a filled screen will increase the likelihood of getting lost among those windows.

Inspired by the KLM approach (Card, 1983), we tried to analyze and compare the time the group has to invest to carry out the activities in both situations by counting the involved actions. Table 2 shows the number of required actions. These actions are mouse clicks (C), the number of characters to be typed (T) and the waiting for the start of a new program or changing from one program to another one (W), and other actions, like walking or plugging in a memory device (O).

Instead of calculating the actual time involved in each action it is better to compare the difference in the number of actions required. In any case, typing or waiting actions take much longer and are more disrupting than mouse clicking. Table 2 clearly shows that typing and waiting is totally absent in CiC transitions and the number of clicks needed for all cases is smaller than the number with standard software.

Experiments with CiC

Of course, the CiC should also undergo field evaluation, i.e., in the classroom during a real lecture. We wanted to validate the strategy of using an integrated tool instead of using traditional off-the-shelf software when using computational technology for supporting lectures inside the classroom more than the software itself. For this purpose, we have set up two scenarios: *standard* and *CiC*. They were used during a one week intensive Java Lecture at the GIT Institute, Waseda University, Japan, in 2005. The setting for the standard scenario included MS PowerPoint as a slides-based presentation tool, MS Explorer as a file manager, Eclipse for showing and running Java programs, and MS NetMeeting for supporting collaborative work. Each student had a laptop computer and the teacher used an interactive electronic blackboard. The CiC scenario used the same hardware as the standard scenario, but of course, including the CiCv2 software. The standard scenario was used during the first three days of lecture and the CiC during the last three days of the course.

At the end of the course, the 21 mainly novice participating students answered a questionnaire on general information, class conditions, presentation quality in the various environments, and general evaluation. About 95 percent of them considered the interactive board as a useful support for the lecture. The students rated their level of agreement with 19 statements using a 5-point scale for both scenarios. The scale ranged from -2 to +2 with negative values indicating rejection and positive values indicating agreement with the respective statement. Some statements were positively phrased and other ones were negatively phrased in order to avoid a pattern in the response set. At the end of the exercise, negatively phrased questions were positively rephrased and assessments were recoded accordingly. We transformed the -2 to +2 range to 1 to 5 to simplify the analysis.

Since four items served for explorative purposes for future research they were left out of the following analysis. The other items were summed up in six categories which (except for “preparation”) proved to be sufficiently reliable. “Preparation” refers to statements regarding possible disturbances caused by logging into the system before the class and learning how to use the new software. “Presentation” comprises items relating to the visual presentation of content, such as adequate colors, fonts, and layout. Items yielding to the flexibility provided by the software for the ongoing flow of the class, i.e. the flexibility to create, modify, or highlight content, and to allow situational adaptation to unexpected questions, were summed up in “flexibility”. The “highlighting” category referred to pointing features to help focus students’ attention. The “absence of disruption” category sums up statements on possible disruptions caused by the use of other input devices than the electronic blackboard, like keyboard and mouse. The last category referred to the usefulness of documents created in class for “follow-up” work by the students.

In order to test conservatively, nonparametric analysis was applied. The nonparametric Friedman test for dependent samples revealed significant group differences for the scales *presentation*, *flexibility*, *highlighting* and *follow-up*. To check which of the scenarios differ significantly from one another, pair-wise comparisons were conducted afterwards. Bonferroni adjustment was applied and the alternative hypothesis was accepted for $p<0.016$. Analyses showed that CiC was evaluated more positively than the standard scenario regarding *flexibility* and *absence of*

disruption. Also *presentation* and *follow-up* received better evaluation in the CiC scenario but the difference was not that significant. *Preparation* received a better score in the standard setting, which can be explained by students having to adapt themselves to a new environment (CiC software). However, the difference does not lead to a concluding answer in this latter case. The results are shown on Table 3.

Table 1: A comparison between the actions necessary with off-the-shelf software and CiC for delivering a specific Computer Science lecture

#	Action	Standard tools implementation	CIC implementation CiC CiC
1	First Activity: Teacher presents the concepts defining a set and shows the implementations of some operations	Teacher presents concepts using a prepared presentation in PowerPoint. Teacher uses a developing environment like Eclipse or NetBeans for showing the implementation	Teacher opens a FreeStyler file which makes use of the Java palette for presenting the concepts and showing the programs
2	Transition: Teacher asks students to complete the rest of the methods for implementing the class. They have to work individually on their computers	The unfinished program is previously stored on a web page and the students download it using a web-browser	Teacher selects the corresponding function from the menu. On a student's screen, it appears a blinking item. She clicks on it, and the skeleton appears on her screen as a separate window.
3	Second Activity: Teacher asks students to try to complete the implementation of the set by writing the rest of the operations individually	Teacher walks around and watches what the students are doing, giving some advice	Teacher can issue queries for: a) check if students are making progress in their work, b) search for solution patterns. Or, she can send messages with suggestions to the students trapped in a problem, or she can simply do as in the standard tool case
4	Transition: Teacher collects the students' work	Students upload their work in a predefined directory using ftp: save their work, open ftp tool, choose file, press send.	Students choose from the "send assignment" menu.
5	Third Activity: Teacher searches for solutions which contain certain patterns, for example, different ways how to implement the delete procedure and shows them simultaneously in order to compare and discuss the various approaches	Teacher has to remember which students implemented which solution. Otherwise she can check each solution to find the one she wants to show. Else, she can ignore the current solutions, and to present the prepared typical solutions	Based on the queries, teacher knows who has which solution, and thus, she begins to show them in the preferred order.
6	Transition: Teacher asks the students to form groups in order to work developing the full solution for each approach and discuss their efficiency (in time and memory usage) (2-3 groups)	Start discussion tools, such as Microsoft NetMeeting or VNC	The assigned student opens a collaboration session; the remaining students connect to it. All these activities via menu.
7	Fourth Activity: Students work in groups	Share screen views	Continue working with CiC but in a synchronized fashion
8	Transition: Teacher collects the students' work	Each group uploads. Same as transition from Second to Third	Teacher collects assignments through a menu option.
9	Last Activity: Each group shows the work in front of the class The whole class discusses advantages and drawbacks of each solution	Each presenter opens the uploaded file and presents it.	Teacher gives control of the workspace running on the board to a student. Each student presents from her own seat (no waste of time) or may walk to the eboard.

Table 2: Comparing the fine-grain operations for Table 1 actions

Action #	Interaction#	Interaction with off-the-shelf software	Operations	Interaction with CiC	Oper-ations
1		No comparison necessary		No comparison necessary	
2	1	Open web Browser *	C,W	[Teacher] Click on Collaborate menu	C
	2	Click to change URL	C	[Teacher] Click on Couple this page	C
	3	Change URL	T	[Student] Click on Icon	C
	4	Press Enter key	C		
	5	Click on the file	C		
Total	for Action 2		4C,T,W		3C
3		No comparison necessary		No comparison necessary	
4	1	Open FTP program *	C,W	Open Transfer Menu	C
	2	Click to change IP address	C	Select "Commit Homework"	C
	3	Change Address	T	Select Folder	C
	4	Click to input Username	C	Select File	C
	5	Input username	T	Press OK	C
	6	Click to input password	C		
	7	Input password	T		
	8	Click Connect	C		
	9	Change folder	C		
	10	Select File to upload	C		
	11	Click Upload	C		
Total	for Action 4		8C,3T,W		5C
5		No comparison necessary		No comparison necessary	
6	1	Click Start menu	C	Click on collaborate menu	C
	2	Click on Netmeeting *	C,W	Click Join Session	C
	3	Click on IP address field	C	Select Session	C
	4	Change Address	T	Click Join Session Button	C
	5	Click OK	C		
	6	Click on Password field	C		
	7	Change password	T		
	8	Click Connect	C		
Total	for Action 6		6C,2T,W		4C
7		No comparison necessary		No comparison necessary	
8	1	Same as 4		[T.] Click on Transfer Menu	C
	2			[T.] Click on CollectAssignment	C
	3			[Teacher] Select Folder	C
	4			[Teacher] Select File	C
	5			[Teacher] Click open	C
Total	for Action 8		8C,3T,W		5C
9	1	The student goes to the front of the classroom	O	Click on collaborate menu	C
	2	She plugs the memory stick in	O	Click Join Session	C
	3	Select File menu	C	Select Session	C
	4	Select File	C	Click Join Session Button	C
	5	Open file	C		
Total	for Action 9		2O,3C		4C

Table 3. Mean rank values for the standard and CiC scenarios, chi-square and significance

Category	Standard scenario	CiC scenario	Chi-square	Significance
Preparation	2.21	1.81	2.655	0.265
Presentation	1.67	2.33	6.644	0.036
Flexibility	1.67	2.60	14.952	0.001
Highlighting	2.31	2.43	4.079	0.453
Absence of Disruption	1.57	2.24	12.868	0.002
Follow-Up	1,71	2.26	9.172	0.10

Differing indices (standard versus CiC) represent significant differences.

Since *highlighting* of some of the contents being presented in class is a typical use of the interactive electronic blackboard, this last result may depend on the teacher applying this feature frequently when working with the interactive electronic blackboard, independently of the software being used. Students appreciated this feature as a useful means to focus attention. According to the initial observations of in-class activities we also found significant results with respect to *disruption* of the normal flow of the class. A promising result for the evaluation of these works is the high flexibility that participants attributed to the system applying the FreeStyler environment, compared to the standard software. It indicates that the system may in fact seamlessly guide the normal flow of the class by providing flexible support for the creation and modification of learning materials, and to flexibly handle unexpected occurrences.

Discussion and Conclusions

Many things can fail in a technology-supported lecture: The equipment may not work properly, the basic software may be inappropriate, the teacher may not know the subject well enough, the courseware may not be adequate, etc. We presented a case in chapter 3 in which all these variables were under control and yet, the results were unsatisfactory. In particular, the teacher was lecturing the subject for about ten years with good results in previous surveys, he was competent, he had prepared the courseware, the equipment did not fail, etc. This is why it was most probable that the cause for this dramatic drop in the course evaluation had to do with the change to intensive computer technology usage in the classroom. This suspicion was further reinforced while reviewing the students' feedback: a good number of them complained about the slow switches between learning activities (transitions) and the long time the teacher wasted doing activities not directly related to teaching. This was distracting and seemed to upset the students.

In order to test our hypothesis about the possible causes we developed the CiC software: an integrated environment in which switches take shorter time and the instructor does not have to remember file names, ad-hoc procedures, etc. We analyzed both the number of operations in such transitions compared to those required for the standard case. The CiC had a smaller number of operations, and those operations were easier and faster to accomplish than the corresponding ones for the standard case. A field evaluation confirmed our hypotheses and the CiC was much better appreciated by the students in almost every field. Only the preparation part of the lecture was rated better with standard software than with the CiC software by the students (although not significantly). All the other parts of the lecture were rated the opposite, i.e., CiC was evaluated better (and significantly so in flexibility and absence of disruption). The preparation part result was explained because the students had to adapt to new software. This is very reasonable: people have to make an effort to learn a new system, but this is compensated with the payoff they obtain. Of course, this occurs with any system (see, e.g., the Technology Transition Model of Briggs, 1998). The point is to determine whether or not it is worth to invest time and effort to learn a new system to get the benefits provided by such system. In the case of the CiC, the subjects definitely agreed the effort to deal with the new system was a good investment.

These results cannot be obtained using traditional LMS systems. First of all, these systems are not designed with the main goal of being used within the classroom; hence, many of them do not provide mechanisms needed to share material in real time. Secondly, file transfer from teacher to student or vice versa in these systems can only be done through the repository or central server for learning materials; thus, data sharing is very slow and idle times during the class appear again. Finally, LMS do not solve the problem of switching from one application to another one: since LMS are generally designed to manage any type of files, the handling of file contents is left to application

programs not embedded in the system and consequently, displaying and manipulating these files imply the initiation of a new program sometimes on a separate window with the corresponding time overhead and cognitive overload.

The generality of the solution is another issue to discuss. Long and distracting transitions are a problem that should be taken into account when deploying in-classroom computer-supported lecturing, independently of the solution to be used. The CiC is clearly one solution, but it may not be the only one in the future, since other integrated environments may appear. We do not primarily intend to advocate the use of our CiC software but to show the need of an integrated approach, independently of which software is used to implement it. If the CiC software is used for teaching other courses, then new palettes should be included. Currently, various palettes for teaching other subjects have been already developed (system dynamics, probabilities, function analysis, genetics, etc.).

Despite the fact our study was done for in-class lecturing support, the results may also be applicable to distance education. Students may also get distracted when transitions are long, and thus, integrated software like the CiC can be useful. In fact, the CiC software can be used in a distance education case, combined with suitable hardware to transmit video images of the lecture delivery and the electronic blackboard.

Finally, we do not think that the focus of attention is the only problem while using computer technology in the classroom, nor is the time and effort needed to accomplish tasks not directly related to teaching the only cause for this problem. However, based on our findings, focusing on and tackling these specific issues apparently leads to considerable improvement of the learning conditions.

Acknowledgements

The authors would like to thank Dr. Jens Hardings for the drawings contained in this paper and the development of a robust and extended version of the software. This work was partially supported by grant No. 1080352 from Fondecyt (Chile).

References

Angel LMS (2007). *Angel*, retrieved March 30, 2008 from http://www.angellearning.com/products/lms.

Baloian, N., Hoppe, H., & Kling, U. (1995). Structured authoring and cooperative use of instructional multimedia material for the computer-integrated classroom. *Paper presented at the ED-MEDIA 95 Conference*, June 17-21, 1995, Graz, Austria.

Baloian, N. Pino, J.A. & Hoppe, H.U. (2000). A Teaching/Learning Approach to CSCL. *Paper presented at the HICSS-33 Conference*, January 4-7, 2000, Maui, HI, USA.

Berge, Z., & Collins, M. (1995). *Computer-mediated Communication and the Online Classroom*, Cresskill, N.J.: Hampton Press.

Briggs, R.O., Adkins, M., Mittleman, D., Kruse, J., Miller, S., & Nunamaker, J.(1999). A Technology Transition Model Derived form Field Investigation of GSS Use aboard the U.S.S. Coronado. *Journal of Management Information Systems, 15* (3), 151-195.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*, Hillsdale, N.J.: Lawrence Erlbaum.

Colazzo, L. & Molinari, A. (1995). To see or not to see: tools for teaching with hypertext slides. *Paper presented at the ED-MEDIA 95 Conference*, June 17-21, 1995, Graz, Austria.

Ferreira, A., & Antunes, P. (2007). On the Need for a Framework for Attentive Groupware Systems. *Paper presented at the 1st Workshop on Adaptation and Personalization in Social Systems: Groups, Teams, Communities*, June 26, 2007, Corfu, Greece.

Grosvenor, I., Lawn, M., & Rousmaniere, K. (1999). *Silences and images: The social history of the classroom*, New York, N.Y.: Peter Lang.

Hoppe, H. U., Baloian, N., & Zhao, J. (1993). Computer support for teacher-centered classroom interaction. *Paper presented at the ICCE '93*, December 15-17, 1993, Taipei, Taiwan.

Hoppe, H.U., & Gaßner, K. (2002). Integrating Collaborative Concept Mapping Tools with Group Memory and Retrieval Functions. In: Stahl, G. (Ed.), *Computer support for collaborative learning: Foundations for a CSCL Community*, Hillsdale, N.J.: Lawrence Erlbaum, 716-725.

Hoppe, H.U., Lingnau, A., Machado, I., Paiva, A., Prada, R., & Tewissen, F. (2000) Supporting collaborative activities in computer-integrated classrooms - the NIMIS approach. *Paper presented at the 6th International Workshop on Groupware (CRIWG 2000)*, 18-20 October 2000, Madeira, Portugal.

Jansen, M. (2003). Matchmaker tng - a framework to support collaborative java applications. In Hoppe, U., Verdejo, F. and Kay, J. (Eds.), Proceedings of the AIED 2003 Conference, Amsterdam: IOS Press, 535-536.

Lingnau, A., Kuhn, M., Harrer, A., Hofmann, D., Fendrich, M., & Hoppe, H. U. (2003). Enriching traditional classroom scenarios by seamless integration of interactive media. Paper presented at the ICALT 2003 Conference, July 9-11, 2003 Athens, Greece.

Lund University (2007). *Luvit System*, retrieved March 30, 2008, from http://www.luvit.com.

Marjanovic, O. (1999). Learning and teaching in a synchronous collaborative environment. *Journal of Computer Assisted Learning, 15*, 129-138.

Mulder, M. C., Lidtke, D., & Stokes, G. E. (1997). Enterprise enhanced education: an information technology enabled extension of traditional learning environments. *Paper presented at the Technical Symposium on Computer Science Education (SIGCSE '97)*, February 27 – March 1, 1997, San Jose, CA.

Norman, K. (1994). Hypercourseware for assisting teachers in the interactive electronic classroom. *Paper presented at the Fifth Annual Conference of the Society for Technology and Teacher Education*, March 16-19, 1994, Washington, D.C., USA.

Ochoa, S.F., Guerrero, L.A., Fuller, D., & Herrera, O. (2002). Designing the Communication Infrastructure of Groupware Systems. *Lecture Notes in Computer Science, 2440*, 114-133.

Ochoa, S.F., Pino, J.A., Baloian, N., & Fuller, D. (2003). ICESEE: A Tool for developing Engineering Courseware. *Computer Applications in Engineering Education, 11* (2), 53-66.

Reichen, J. (1991). *Lesen durch Schreiben. (German teacher's guide to "Reading through Writing")*, Hamburg, Germany: Otto Heinevetter Lehrmittel GmbH.

Retalis, S., Makrakis, V., & Skordalakis, E. (1997). EONT courseware development methodology. *Paper presented at the ED-Media '97 Conference*, June 14-19, 1997, Calgary, Alberta, Canada.

Santoro, F. M., Borges, M. R. S., & Santos, N. (2005). Learning to Plan the Collaborative Design Process. *Lecture Notes in Computer Science, 3168*, 33 – 44.

Scardamalia, M., & Bereiter, C. (1994). Computer support for knowledge-building communities. *The Journal of the Learning Sciences*, 3 (3), 265-283.

Scholar360 (2007). *Scholar 360*, retrieved March 30, 2008, from http://www.scholar360.com.

Sparks, R., Dooley, S., Meiskey, L., & Blumenthal, R. (1999). The leap authoring tool: Supporting complex courseware authoring through ruse, rapid prototyping and interactive visualizations. *International Journal of Artificial Intelligence in Education, 10*, 75–97.

Tewissen, F., Lingnau, A., Hoppe, H. U., Mannhaupt, G. & Nischk, D. (2001). Collaborative writing in a computer-integrated classroom for early learning. *Paper presented at the Euro CSCL 2001 Conference*, March 22-24, 2001, Maastricht, The Netherlands.

Treisman, A., & Gelade, G. (1980). Feature Integration Theory. *Cognitive Psychology, 12*, 97-136.

University of British Columbia (2007). *The WebCT system*, retrieved March 30, 2008, from http://www.webct.com.

Vertegaal, R. (2002). Designing of attentive Interfaces. *Paper presented at the Symposium on Eye Tracking research and applications*, March 25-27, 2002, New Orleans, LA.

WBTSystems (2007). *TOPCLASS*, retrieved March 30, 2008, from http://www.wbtsystems.com.

Wu, C.-C., & Lee, G.C. 2002). ICT Integration at a Taiwan Secondary School. *Paper presented at the CATE 2002 Conference*, May 20-25, 2002, Cancun, Mexico.