20

How to distribute learning facilities by means of a network: some issues and a case study

G. Adorni[1], M. S. Barbieri[2], D. Bianchi[1], E. Calabrese[1] and A. M. Sugliano[3]

[1] Dipartimento di Ingegneria dell'Informazione Università di Parma
43100 Parma, Italy,
e-mail: adorni, bianchi, calabrese@ce.unipr.it
[2] Dipartimento di Psicologia Università di Trieste
Via Università 7, 34123 Trieste, Italy,
e-mail: barbieri@univ.trieste.it
[3] Dipartimento di Scienze Antropologiche Università di Genova
Vico S.Antonio 5/7 16126 Genova, Italy,
e-mail: sugliano@disa.unige.it

Abstract

In this paper we present a distributed client-server architecture which integrates hypertext tools with laboratory, evaluation, and communication tools, as a support for learning. A part of the architecture has been implemented and tested with undergraduate students. The preliminary results are encouraging and an extension of the system is planned with other communication tools such as bulletin board, blackboards, voice and images.

Keywords

Distance education, Web applications, Prolog, Evaluation

The Virtual Campus F. Verdejo & G. Davies (Eds.)

1. INTRODUCTION

According to a well-known theory in psychology, active learning produces better results than other forms of learning (Piaget, 1947; 1964). The possibility of acting on the learning material, observing the results of different actions and reasoning upon them, deepens the understanding of the learning material and consequently affects learning outcomes.

First hand experimentation with the learning material is however not sufficient. Students need supervision by the tutor as well as interactions with other students to discuss their own views and ideas and compare what they learned with their peers (Steffe and Gale, 1995). All these resources are usually provided in a traditional on-site learning environment where students sit in a classroom with a tutor and other students. This environment involves social comparison and interaction between the students as well as supervision by the tutor. It can also entail work in laboratory sessions which involves active experimenting with the educational materials.

By exploiting Internet facilities, computer and network technology have recently made all these resources easily available even to students who are not on-site. Access to the Internet allows distance students to sit at their personal computer and perform in a single place a number of different actions. For example, they have access to learning materials, contact with tutors and peers, do practical tasks and use the virtual laboratory. These are usually performed by on-site students by means of a number of different media (books, computers, telephone, face-to-face interaction), and sometimes in a number of different places (classroom, laboratory, cafeteria).

This new type of technology sometimes includes extra resources that may also improve the effectiveness of on-site learning such as hypertexts which can emphasise the conceptual links inherent in the learning material, or computer conferencing which can support topic-based asynchronous discussion among the students.

Though widespread acceptance of new learning technologies might require a long time (Cuban, 1986), it is worth trying out new resources to exploit their possibilities and assess their impact through different forms of evaluation.

The evaluation of these information technology resources is also a matter of extensive debate (Crook, 1997; McAteer et al. 1997). Usually, the focus is on the technical features of the system and their user-friendliness as well as on their learning effectiveness. However, the way the students interact with the new resource might clash with their already acquired study habits. It might be important therefore to know what the users actually did with it and why. Subjective data are important in order to know which features of the system the users felt met their learning needs and which ones were overlooked. As a consequence, a comprehensive evaluation of the resource requires putting together different pieces of information such as the students' activity patterns, their reason for proceeding the way they did, their acceptance of the resource, and lastly their learning outcome.

Further on in this paper, we discuss a distributed architecture for learning purposes which integrates different kinds of media. More precisely, section 2 is an introduction to the proposed architecture. Section 3 discusses a first implementation of part of the architecture used as a support for a course on "Logic Programming". Along with some concluding remarks, section 4 presents a preliminary evaluation of the implemented system performed during the present academic year for students enrolled in the first level degree called Diploma of Computer Engineering at the Engineering Department of the University of Parma.

2. PROPOSAL OF AN ARCHITECTURE

The Internet can be an important means for the diffusion of learning material and culture. In fact, a lot of news groups and Internet sites around the world offer learning material and information about educational topics. In this direction, the long-term goal of our project is to "break down the walls" of the classroom, which is the traditional learning environment, by means of a client-server distributed architecture through a network.

Each client supports a user-friendly man-machine interface integrating monitor, keyboard and mouse with a microphone and a CCD camera.

The server allows on-line presentation of a course module and its related tutorials. In the same environment, it integrates virtual instruments and tools to directly experiment with the learning material and to verify the "learning level of the student". Such an environment offers the possibility of asking questions to a tutor via e-mail facilities. Notes, comments and reflections related to the learning material can be posted on an on-line bulletin board and made accessible to all the students of the virtual classroom and to the tutor. The student can interact with the tutor (when he is on line) or with other students (if there are any) through the microphone. They can write text and draw sketches on a shared blackboard. By means of the camera they can even look "into the eyes" of the participants in the discussion.

The above facilities are available through the server by means of the network. More precisely, the server hosts the following: 1) teaching material organised as hypertext, integrating also animated text and parts of films; 2) software packages simulating instruments and tools for laboratory activities; 3) software packages for remote cooperative work which include management policies of the blackboard and management of voice communication as well as images; 4) tools for self assessment and final evaluation of the learning level (see Figure 1).

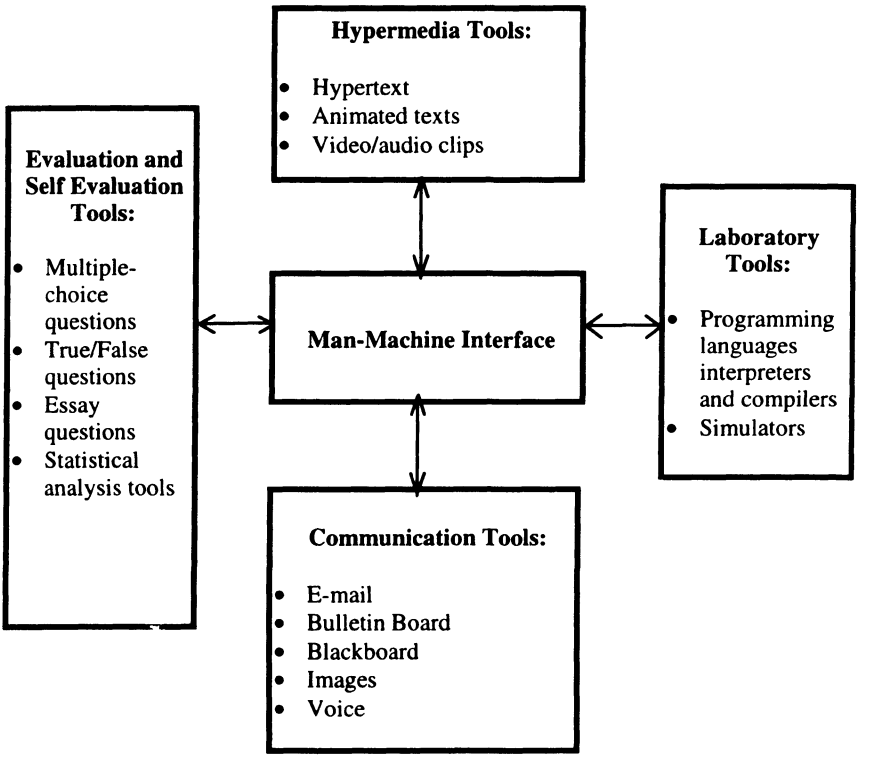

Figure 1. The proposed architecture.

3. A CASE STUDY

The architecture introduced in the previous section has been partially implemented as a system devised to allow the on-line use of a course module and its related tutorials. The subject matter of the module is "Mathematical Logic, Logic Programming and Prolog", which is used in courses for the diploma degree in Computer Engineering at the Engineering Department of the University of Parma. The client-server experimental setup is shown in Figure 2.

The theoretical part of the subject matter is presented through hypertext. Linked to the main topics of the key chapters there is a series of tutorials (guided training exercises), with questions and problems that the students are invited to solve.

Students can actually try out their answers and solutions by using within the browser an available Prolog interpreter on the server together with a number of

files related to the examples presented in the tutorials. These sample files can be directly loaded and tried out in this environment which we called "PrologLab".

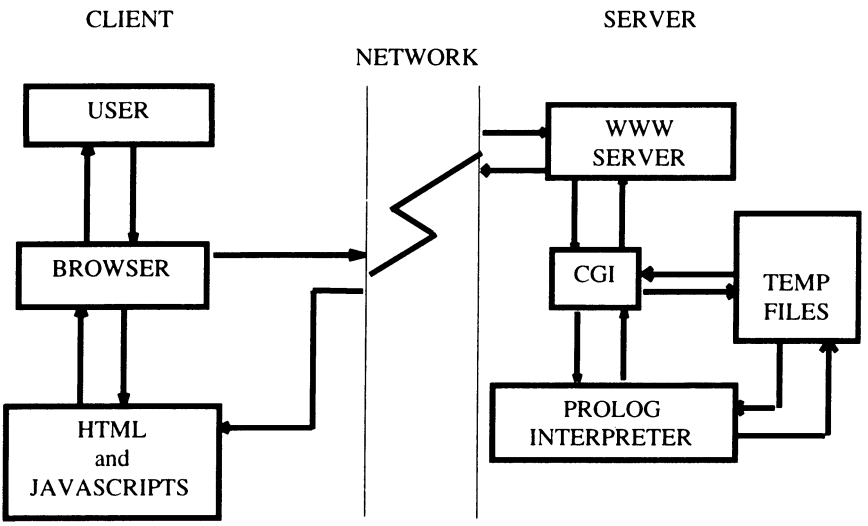

Figure 2. HyperProlog: experimental setup.

The students can easily switch from the hypertext (part of Hypermedia Tools) to the PrologLab (part of Laboratory Tools) or use both concurrently.

At the end of each tutorial there is a self-assessment test which the students can take and an e-mail facility where students can write their comments and send questions to the tutor.

The above implemented modules are integrated through a friendly Man-Machine Interface (see Figure 3).

The client-server architecture is based on a WWW browser and server. A platform independent user interface can be developed using HTML which was designed mainly as a markup language for creating hypertext. The use of CGI programs allows one to develop general user interfaces to almost any application. These interfaces can be further improved by using tools like Javascripts and Java applets. Anybody with a browser and an Internet access can use the system.

A Prolog interpreter is accessed as a part of a CGI (Common Gateway Interface) in order to process HTML forms. (Carpenter, 1996; Cabeza, 1996; Loke, 1996). A CGI program is started by the WWW server in response to the submission of an HTML form and stops when a response is generated for delivering to the client. This interaction procedure has two disadvantages. The first is that no state is preserved from one interaction to the next. States can be maintained by means of temporary files, Netscape "cookies", etc., but in some applications the state should

be maintained by the application itself. An example would be if one wants to keep the contents of the Prolog database. The second problem is that starting and

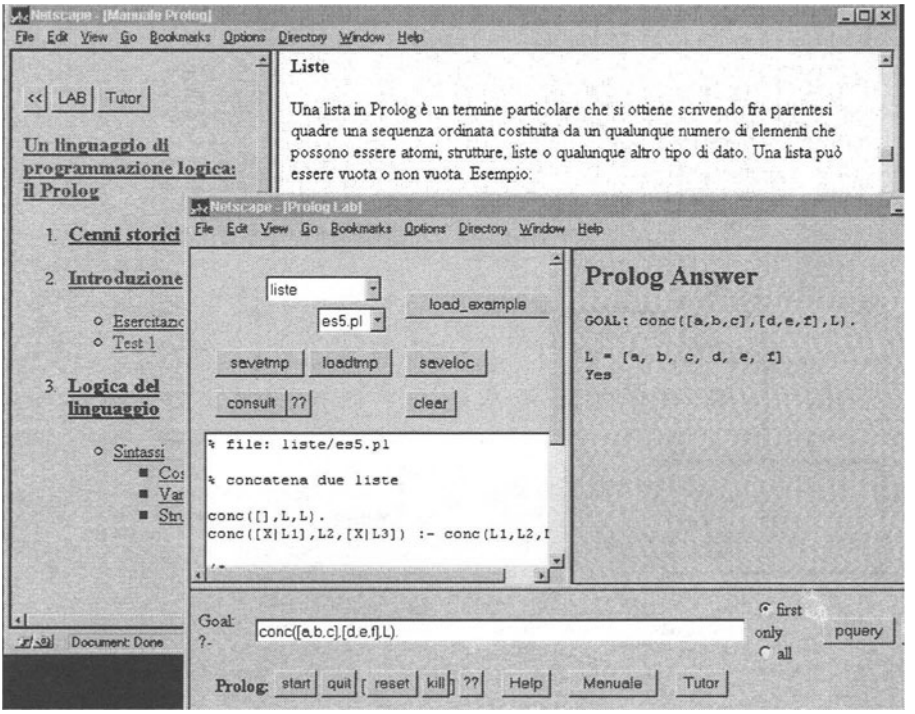

Figure 3. Man-Machine Interface.

stopping the application may be very inefficient. For example, it takes a long time to load the Prolog interpreter and to reload the previous state. A better approach is to have the application running continuously maintaining its status while the CGI program is only an interface between the WWW server and the application (the Prolog interpreter in our case).

In our architecture, the Prolog interpreter is the server, while the WWW browser, the client, is the user interface process. The CGI program or script converts the information coming from the browser to a Prolog query that is submitted to the interpreter. On the other side, the response of the Prolog interpreter is converted in an HTML format and is delivered to the client by the WWW server.

Many WWW-Prolog systems are designed to interface the client to a particular application such as a searching engine or an expert system, etc. The user input has the form of a query that Prolog has to satisfy.

Our purpose on the other hand is educational, so we have to allow the client to make basic actions with the interpreter. For example:

- consulting a user program in the Prolog database;
- executing a goal and showing the response to the user and the resulting binding of the variables.

It is also necessary to handle user errors like syntax errors in the program or in the query passed to the interpreter, or execution errors. These errors should be notified to the client.

Some debugging facility should also be provided. In our case, to avoid an overloading of the server only a trace facility is enabled.

The user can create and edit a temporary file containing his program. This temporary file, which resides on the server, is used for the consulting operation. For the sake of simplicity, the goals submitted to Prolog are also written to a file which makes it easy to check the goal's syntax. A log file managed by the Prolog interpreter is used to notify the occurrence of errors to the user.

Once started, the Prolog interpreter executes the following cycle:

```
while (communication is active) {
 wait for a command from the client (a Prolog goal);
 execute the goal;
send the output, in HTML format, back to the client;
}
```

An important issue is how to manage multiple HTML clients at the same time. In this case, it is necessary to keep track of each client's status. Many solutions were devised for this problem using some form of parallelism (Szeredi, 1996). We have preferred a solution in which each client has a different copy of the interpreter in execution. Each copy of the interpreter knows an identifier of the client, and a different set of copies of the temporary files is created for each client.

Going into more details, HyperProlog (which is how we called our integrated system) requires a browser that supports frames, forms, and Javascripts. It has been extensively tested with Netscape 3.0, but a few tests have also been done with Internet Explorer 3 with apparently no problems. The HTML files and the CGI's reside on an Apache server on a Pentium Pro computer running Linux.

The hypertext is composed of several HTML files. The browser window is split in two frames. The left frame contains the table of contents, while the text is in the right frame. By clicking on an item in the left frame, the corresponding paragraph is opened in the right frame and, if present, a more detailed table of contents is opened in the left frame. Navigating buttons are present in both frames. They allow one to move between chapters and to go back one level. All these buttons use Javascript functions.

Besides the usual hypertext links, the right frame contains buttons to go to the exercise section, to the test section, and to the PrologLab. In the latter case, the examples presented above the button are automatically copied to the edit area of PrologLab.

PrologLab starts as a new browser window and allows the user to interact with the Prolog interpreter on the server. The window is made up of three frames: the "edit" frame, the "query" frame, and the "answer" frame (see Figure 3).

The "edit" frame contains form elements to edit a Prolog program, to select one of the many examples which are part of or complement the hypertext, to insert the selected example in the edit area, to save, to retrieve, to clear the edit area, and to "consult" the Prolog program contained in the edit area. Context-sensitive help buttons are also available.

The "query" frame contains form elements to insert the Prolog goal and to submit the query. It also contains several buttons which allow one to start and quit Prolog, to kill the Prolog process in those cases in which the regular "quit" doesn't work, to jump to the hypertext window, to send an email message to the tutor. Context-sensitive help buttons are also available.

The "answer" frame is where the Prolog answers and information messages are displayed.

The PrologLab window and frames are generated and handled by CGI scripts written in Perl, which interact with the Prolog interpreter through some temporary files. One Prolog process is started and one set of temporary files is allocated for each user. The temporary files contain the user input, the Prolog output, the error log, the program to be consulted, etc.

Some buttons in the exercise section of the hypertext let students take a test. These buttons activate a CGI on the server which handles the whole test in a new browser window.

The information about the test and about the student taking it together with his/her answers is kept on a separate file for each student. This file is used throughout the test and its name is sent back and forth between client and server using a Netscape cookie, which allows keeping state information between separate browser requests.

The test window contains two frames. The right frame displays one question at a time, while the left frame contains buttons to jump to a particular question or to cycle through the questions. It also displays a "Finish" button. One can go back and forth between questions to check the answers given before clicking the "Finish" button.

When the student clicks "Finish", the CGI creates a summary table listing the answers given, the correct answers, and the ratio of the guessed, missing, and wrong answers. It displays the table together with all the questions in the test and the answers given by the student. At the same time, an e-mail is automatically sent to the tutor with all the information needed to evaluate the results of the test. While the multiple choice and the true or false questions are corrected automatically, the essay questions need to be graded by the instructor.

4. SYSTEM EVALUATION

The system was tried out by a group of 22 students attending the third and last year of a Computer Science program. They were all males and their age ranged from 21 to 23 years (mean 23). They worked with the system for about 30 hours.

A second group of 20 students of the fifth year of the Electronic Engineering program worked on the system for four hours in order to evaluate the system in terms of its robustness and the friendliness of the interface.

In the evaluation of this system we relied on three types of information:

1) Analysis of the students' patterns of activity by means of system logs. This analysis will give details about the students' use of the system; that is, which pages of the text they looked at most, which facilities they used the most (hypertext, tutorials, self tests, e-mail, PrologLab) or which ones they overlooked.

2) Effectiveness of the system in terms of the students' learning outcomes. This evaluation is based on the results of the final test.

3) Students' attitudes toward the resource. A questionnaire was administered to the students to assess how much they liked this resource in comparison with a traditional course. They were asked which part of the system they liked most and which ones least. They were also asked which parts they felt to be the most useful and why. The results of the questionnaires were integrated with the information gathered during group discussions on the topic.

Patterns of usage can be obtained by two sources. The first source is provided by the WWW server that automatically saves each transaction in a log file. In this file there is a record for each page accessed and for each form submitted. In the latter case, the associated query string is also registered.

By observing these data we can obtain statistics about page usage. For example, we can find out the topics the student read, or the operation that the user performed, such as loading a file in the edit area, consulting the file, submitting queries to the interpreter, etc.

Other data about the user activity can be directly recorded by the system. For example, when a form is submitted to the server and a CGI program is started, it can record information in a file. Every time that a student starts a session our system saves his user name, host name, date and time, etc. The data obtained from the system logs show that students did not access the general presentation of mathematical logic very frequently. The four related pages - propositional logic, introduction to mathematical logic, logic of first order predicates, logic for problem solving - register an average of 50 accesses each. On the contrary, the Prolog page registers almost 700 accesses.

Table 1. Number of accesses to tutorials, solution of exercises and to example files for laboratory use. (Tutorial contents: I - facts, rules and queries; II - syntax and structures; III - lists; IV - backtracking and cut; V - arithmetics).

	I	*II*	*III*	*IV*	*V*	*Tot*
tutorials	111	111	168	249	110	749
solutions	24	46	46	33	30	179
example files	184	65	312	255	74	928

The tutorials were also read a lot. Working on the tutorials includes checking the solutions of the exercises and loading the example files. Table 1 shows the number of accesses to the tutorials with the related files and demonstrates the interest shown by the students for practising what they learned in the Prolog presentation. We can see that tutorials 3 and 4 are the most accessed. This can be explained by the difficulty of their topics. Tutorial 5 is the very last and has not been accessed very much simply because of the lack of time. Self assessment tests were not meant for final grading, but were devised to give the students feedback about their understanding and mastery of the subject matter. There were five self assessment tests. Each one included a number of multiple-choice and true/false questions and

one or two essay questions. These tests were automatically sent to the tutor for the correction of the essay questions and for general supervision. Multiple-choice and true/false questions were automatically graded. Immediately after having sent the test to the tutor, the students could see a screen showing the following information: how many answers to the multiple-choice and true/false questions were correct, how many were wrong and how many were missing. They could also reread the questions and since they knew now where they had made a mistake, they could then try to understand the reason for their error. Later on, the tutor would check the answers to the essay questions and send each student the necessary suggestions and advice by e-mail.

Self assessment tests were randomly generated from an available database. Therefore, each student received a different version of the test. This allowed the students to repeat the tests as many times as they felt it necessary. These tests were largely used by the students who appreciated the possibility of checking up on their progress. As it is shown in Table 2, many of them tried each test more than once. Some did it out of pure curiosity while others only wanted to be reassured about their performance.

Table 2. Median and mode of students' test usage. Each test refers to a tutorial (see Table 1).

Test #	*I*	*II*	*III*	*IV*	*V*	*Tot.*
Median	2	3	3	3.5	1.5	13
Mode	1	2	2	0	0	9
Total	45	77	75	78	52	327

It should be noticed however that the range of variation is quite large, especially for the last tests. Some students did not try them at all, while others tried them up to eleven times. This repeated trying however contributed to their learning.

Table 3 shows an increase in the mean percent of correct answers from the first to the last trial of each test. The increase is particularly evident and statistically significant for the last three tests. Students tried self assessment tests both during the course and in preparation for the final test.

At the end of the course, students were graded by means of a test also given through the Web. This test was the same for all the students. It included 10 questions and covered all the topics of the course. Although two of the students did not take the final test, those who did performed very well. The median of correct answers is 87 (range 60 to 100) and the mode 94. The students were motivated to do a good job, but these very high results also show that our environment was indeed very effective.

Table 3. Mean percentage of correct answers to the first and last trial of each test and overall mean percentage of correct answers to the same tests.

Test #	*I*	*II*	*III*	*IV*	*V*	*Tot.*
First	96.8	89.3	77.5	65.6	77.4	81.8
Last	97.9	91.7	86.0	85.6	90.5	90.4
Overall	95.6	85.9	84.2	85.1	86.2	86.7

Lastly, we collected some subjective data by means of questionnaires and interviews. The questionnaire included twelve questions. Students were asked whether they were familiar with the hypertext format and how much they liked it. Then, they were asked whether they had ever used an integrated tool such as HyperProlog. They were asked whether or not they liked it better than using a traditional book plus a separated Prolog interpreter and why. They were asked whether they had had any difficulty with HyperProlog and, if so, of what type. Lastly, they were asked what parts of HyperProlog they used most and least and to give their reasons.

Only a few of the students had ever had a course supported by hypertext material (18%) and none of them had ever followed a course with an integrated didactic tool such as HyperProlog. Nevertheless, they appreciated and enjoyed the experience. The majority of them (60%) said that in case they had to repeat the course they would rather choose an integrated tool instead of a traditional book and an interpreter. In a 7 point scale ranging from 1 (preferred the traditional book plus an interpreter) to 7 (preferred the integrated tool), with 4 as the indifference point, the median value was 5 and the mode was 6. This is a clear indication that they gave a positive evaluation to the use of this tool. Those who preferred the integrated tool justified their choice saying that it allowed them to try practical activities related to the theory they had just learned and that this made learning faster. Those who said that they would have preferred the traditional book plus a Prolog interpreter expressed their desire for face to face interaction with the teacher and also mentioned fatigue induced by reading the screen. They also said that they liked the PrologLab, but would have liked to have a personal copy of the material and felt limited by the fact that they did not have access to Internet from their homes.

The parts of the system that they said they had used the most were self tests (71%), tutorials (67%), and PrologLab (43%). These were also the parts they felt were the most useful: tutorials (67%), self tests (57%), PrologLab (52%). They felt that these three parts helped them to clarify their thoughts and doubts. They appreciated the richness of the examples and the possibility of practising. In the self test they liked the immediate feedback available in the computer corrected questions. They would have preferred not to have the questions sent to the tutor for correction though, because they thought that this had limited their possibility to try the self tests several times.

The parts that they used the least were the theoretical presentation (81%) and e-mail. In their opinion, the theoretical presentation was unclear and difficult. They

do not justify their overlooking the e-mail facility but it should be remembered that the experience was on a campus where tutors were available. They preferred direct interaction as demonstrated by their active participation during face to face lectures.

In conclusion, it seems quite evident that this integrated environment has been appreciated by the students who liked the possibility of activity included in the tutorials by means of the PrologLab and used these resources effectively for their learning. In order to fully exploit the possibilities of this system the following will be necessary: a larger diffusion of technological resources such as individual access to Internet; changes in the students' study habits with more independent study activity and less reliance on immediate feedback from the tutor as in a face to face interaction; changes in the features offered by the system such as the availability of more different types of activity; a better automatic feedback in the correction of students' assignments and their evaluation; a wider range of different communication tools.

In this paper we presented a distributed client-server architecture as a support for learning. A part of the architecture has been implemented and experimented with undergraduate students. The preliminary results are encouraging and a more extensive use of the system is planned. The integration of hypertext with the virtual laboratory and the on-line evaluation which was presently integrated only with an e-mail facility for communication exchanges will be extended to the other mentioned communication tools such as bulletin boards, virtual blackboards, voices and images

5. REFERENCES

Cabeza, D., Hermenegildo, M., and Varmaa, S. (1996) The PiLLoW/CIAO Library for INTERNET/WWW Programming Using Computational Logic Systems, in *Proceedings of the 1st Workshop on Logic Programming - Tools for Internet Applications*, Bonn, Germany, September 2-6,1996.

Carpenter, B., (1996) A Prolog-based CGI handler. *http://macduff.andrew.cmu.edu/cgparser/prolog_cgi.html.*

Crook, C. (1997) Making hypertext material more interactive: some undergraduate reactions. In press *Journal of Computer Assisted Learning*, **13**.

Cuban, L. (1986) Teachers and Machines. New York, Teachers College.

Loke, S.W., and Davison, A. (1996) Logic programming with the Word-Wide Web, *The 7th ACM conference on Hypertext*, ACM Press. Available from *http://www.cs.unc.edu/~barman/HT96/P14/lpwww.html.*

McAteer, E., Tolmie, A., Duffy, C. and Corbett, J. (1997) The evaluation of Computer Mediated Communication as a learning resource in Higher Education. In press *Journal of Computer Assisted Learning*, **13**.

Piaget, J., (1947) La psychologie de l'intelligence. Alcan, Paris.

Piaget, J., (1964) Development and Learning, in *Piaget Rediscovered* (eds. R.E. Ripple and V.N. Rockcastle), Cornell University Press, Ithaca, NY.

Steffe, L.S. and Gale, J. (1995) Constructivism in Education. Erlbaum, Hillsdale, N.J.

Szeredi, P., Molnar, K. and Scatt, R., (1996) Serving Multiple HTML Clients from a Prolog Application, in *Proceedings of the 1st Workshop on Logic Programming - Tools for Internet Applications*, Bonn, Germany.

6. BIOGRAPHIES

Giovanni Adorni is a Professor of Computer Engineering at the the University of Parma and Representative of the Rector of the University of Parma for "Information and Communication Technologies". His research activities are in the area of computer vision, autonomous vehicles navigation, planning, man-machine interaction, multi-agent systems. He coordinated several national research programs on artificial intelligence, and partecipated to different european programs, including the seven years EUREKA PROMETHEUS project, where he has been the national coordinator and a member of the european steering sommittee for the PRO-ART sub-project. He received, among others, the Systems Research Foundation Award in recognition of his research on "knowledge based planning", and the ICL European Artificial Intelligence Prize on Cognitive Modelling. He has authored and coauthored over hundred scientific papers in international journals, contributed volumes, and conference proceedings. Professor Giovanni Adorni is a member of AI*IA and AAAI.

Maria Silvia Barbieri was born in 1947. She obtained a degree in Philosophy at the University of Pavia, Italy, in 1970, and a *specializzazione* in Psychology at the University of Milano, in 1974. Since 1978 she has been teaching at the University of Trieste, where presently she is full professor of Developmental Psychology. Her research interests concern cognitive development, the effects of social interaction on language and cognition, and, recently, computer mediated communication in education.

Dario Bianchi was born in 1946. He received the Dr.Ing. degree in nuclear engineering from the Polytechnic of Milano in 1971 and a *perfezionamento* in Solid State Physics at the University of Parma, in 1976. Since 1983 he is associate professor at the University of Parma. His research activity has been in solid state physics and in the usage of microcomputers in experimental physics. Presently his main interests are in the areas of natural language processing, of massively parallel systems, of genetic algorithms and fuzzy logic. He is a member of AI*IA and ACL.

Eduardo Calabrese was born in Messina, Italy on 1943. He obtained a Ph.D. degree in Physics at Lehigh University, U.S.A. in 1971. Since then, he has been employed as a professor at the University of Parma, where he has been teaching courses in Physics and Informatics. Currently, he holds a position as associate professor of Informatics at the University Engineering Department. His

professional interests include educational uses of the computer which involve Internet and distance learning.

Angela Maria Sugliano was born in Genoa in 1968. She received the Laurea Degree in Philosophy of Science from the University of Genoa in 1992, and the Diploma Degree in Journalism in 1996. She is a Ph.D student in Methodology for Research in Psychology at Faculty of Psychology of the University of Genoa. Her research activities are in the area of social and psychological aspects of computer mediated communication.