An e-Learning Environment based on Open-Source Software

SEBASTIANO PIZZUTILO, FILIPPO TANGORRA and BERNARDINA DE CAROLIS,
Dipartimento di Informatica
Università di Bari
via Orabona 4, 70126 Bari
ITALY

Abstract: - The system we have built is aimed at supporting learning activities of the degree course in computer science. The defined learning environment is based on ATutor, a freeware and multiplatform Learning Component Management System developed by the University of Toronto. Among the present courses, the attention is pointed out on the computer architecture course, which also integrates a processor simulation within the courseware.

Key Words: - Educational software, Courseware, Computer architecture simulation

1. Introduction

The computer aided learning is suitable in particular to study computer science for many reasons: for the well recognized potentiality of the multimedia communication in improving the learning process, for the flexibility of the computer environment which allows the teacher to adapt contents and didactic tools to new learning needs and for the "autoreference" of teaching this particular topic using just the technology subject of the study.

The objective of the implementation of the present learning environment consists into encouraging and supporting teachers to apply e-learning techniques in addition to the progression of the regular course. In fact, e-learning approach is suitable for some peculiar conditions of our degree course regarding the high number of student (the annual enrolment is approximately 500 students) and the need to supply courses by teleconferencing to several University sites in the region. The e-learning environment, which do not substitutes the traditional course, aids the teachers into didactic activities that are impossible to give for the previously said conditions in order to follow each student in the learning process, to suggest individual training path for reinforce concepts of lectures, to verify with opportune tests the course effectiveness and so on.

The courses supported by the e-learning environment have been implemented on ATutor, an open-source Learning Component Management System (LCMS), developed by the University of Toronto [1].

The choice of the e-learning platform has been determined by the need to change an older release of the system, developed using Toolbook and composed by only one course in a prototype version.

The old system was organized in different modules (or components) devoted to the following functions:

- Management of theoretical contents and exercises (for teachers),
- obtaining statistics of the use (for teachers) ,
- evaluation of student learning process (for teachers),
- management of the learning tracks (for teachers and students),
- studying of theoretical hypermedia lectures (for students),
- making exercises (for students),
- self-evaluation (for students).

In this paper we present the new e-learning environment that provides these same characteristics and in addition new functions, like the following :
i) a more dynamic and efficient management of components, ii) allowing the use of the system on different operating systems, on the network and iii) maintaining low the costs of hw/sw resources of the clients and the server. For this aims, we have decided to use a freeware LCMS platform, compatible with standards and usable on different systems (Windows, Linux,…).

Moreover, in the paper we present an example of a computer architecture course (ARCHO2), that provides and integrates two different sub-components: ARCAL, to support the study of theoretical concepts and to verify lectures and APE, to support the laboratory activity by using a computer architecture simulator.

2. The LCMS Open-Source platforms

The LCMS platforms have acquired growing popularity in the e-learning field and constitute the evolution of the Learning Management Systems (LMS). In addition to the usual functionalities of LMS systems to manage e-learning courses, the LCMS emphasizes the organization of the e-learning course by a content point of view, as composed of basic reusable and independent learning units, called Learning Objects (LO).

We have considered the more popular available LCMS open-source platform, taking in account the following characteristics:

- easy management and maintenance of the LCMS,
- usability and accessibility of courses supplied by LCMS,
- documentation and technical support availability,
- IMS and SCORM standard conformity,
- multiplatform support,
- self-learning functionalities,
- cooperative working functions.

Therefore, we have evaluated many LCMS platforms [1-5]:

- ATutor
- SPAGHETTI LEARNING
- MOODLE
- ILIAS
- CLAROLINE

The ATutor system has emerged as the tool that better met project requirements. In fact ATutor has the following features: 1) a specific interface for the course management; 2) a very easy building and navigation among courses; 3) an import and export of learning contents, according the IMS/SCORM standard; 4) the availability of the integration of e-mail functions to contact all students of a particular course; 5) the management of forum and chat functions; 6) the user tracing by access storing; 7) producing statistics on courses access and test results; 8) context help; 9) the glossary; 10) the server multiplatform (Windows, Linux e Unix) support.

The ATutor architecture has been developed by the Adaptive Technology Resource Centre (ATRC) of the University of Toronto. The figure 1 shows the standards used to manage the learning objects used to organize and manage the independent cells of a course and to allow its use by teachers and student. It has been written using PHP, Apache and MySQL, which are useful standard to resolve our system requirements.

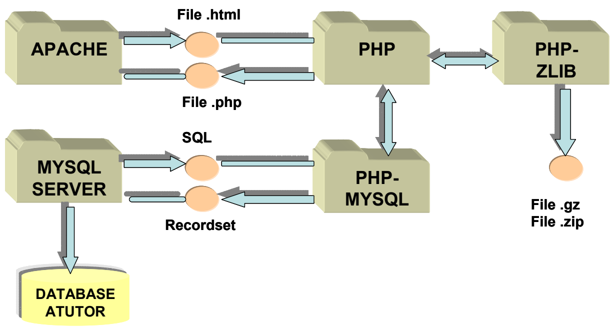


Fig. 1 Architecture of ATutor

3. The structure of the e-courses

The system offer the possibility to define on-line courses, taking in account the typical learning needs of a student user, as:

- the choice of personalized learning path, in order to acquire theoretical knowledge taking account of the more appropriate learning times for everyone. The course contents are organized in didactic units and are activated and shown progressively, following the sequence of teacher’s lectures, to avoid the complete substitution of lectures with a virtual interaction with the on-line course. In fact, in our approach the aim of e-course is to reinforce the learning process, focusing it on the interaction with the teacher and not with a computer. Moreover, the navigation trough contents of the course happens at different level of granularity. At a lower level, the student can consult the material of the teacher to support the study phase. At a higher level, the system shows the same contents in a synthetic form (eventually storing the slide used by the teacher in his lectures) in order to support the phase of revision. Links from the synthetic level to the corresponding argument in the detailed level assure the deepening of not clear contents. The complete didactic material at the end of the lecture cycle provides an e-course for not attending students.
- Testing the knowledge of contents of didactic units for a self evaluation of the obtained progress in the learning process. In case of negative results of tests, the system suggests to the student specific arguments to re-examine. In fact, tests are correlated to the arguments at the end of every didactic unit;. The results of the tests are stored in a database and they are shown a) in individual report of each student in order to follow the learning process of the single student or b) in statistical report to control the classroom trend. Moreover, the system is able to store the student’s learning path (the visited units,

the results of exercises and tests), so that it is possible to recall the learning process starting from the break point.
- Developing exercises on the visited didactic units for proving the re-elaborative ability of the students to apply the just learned concepts to new situations. Solutions, links to analogous carried-out exercises and suggestions to correlated theoretical arguments aid the student in this applicative activity.

The other main user of the course is the teacher or the on-line tutor. The system allows the teacher to support both main aspects of its institutional role : the capacity of transfer knowledge, by arranging didactic material, and the possibility to verify the learning process of the students. Therefore, the systems permits tasks for:
- creating, updating and managing didactic unit contents (at different granularity level), tests and exercises. It is also possible to establish the path of the various arguments and eventually links among them.
- Verifying tests and exercises made by students analysing their results both in individual or aggregate form. It offer the possibility to download the traceable data to control the study path of students and statistical parameters on accessed didactic units.

The course considers also a skilled technical user providing functionalities of management of aspects like as, for example, the authorizing process for the access to the course, the coordinating and monitoring of its use and so on. This type of user is identified as the learning administrator.

The overall view of the user of course is modelled in figure 2 and it includes both the functions and the quality parameters of system.

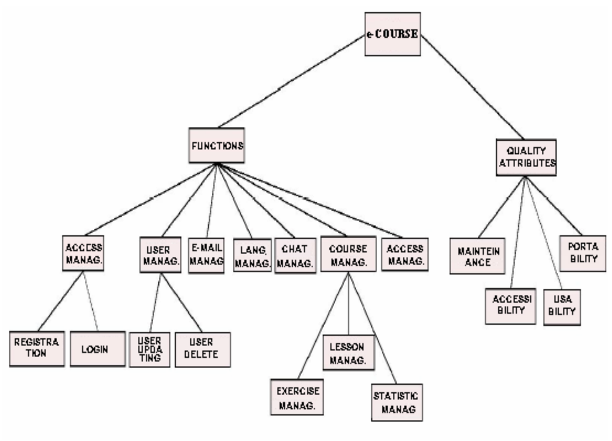


Fig. 2. The e-course structure

The model follows a top down approach, therefore the figure 3 reports the function's refinement of the lesson management.

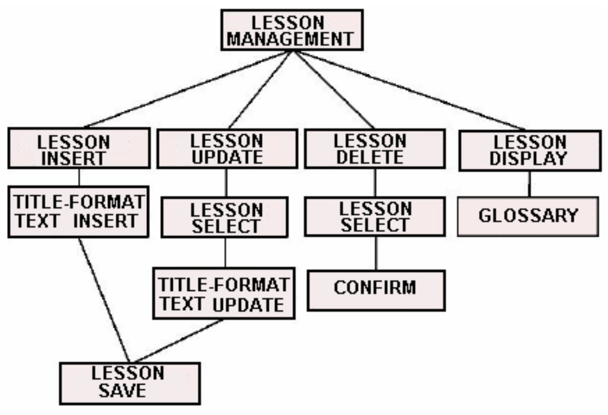


Fig. 3. The functions associated to the lesson.

The course we have implemented provides the access and the navigation for three distinct users: the student, the teacher and the learning administrator using the "ATutor" platform.

4. ARCHO2: an example of e-course

Among the courses realized in the described platform, we illustrate the ARCHO2 course, while it shows peculiar characteristics.

The ARCHO2 course presents contents (in Italian) corresponding to didactical units of the course of "Computer Architecture". In addition, the basic concepts of theoretical lectures can be reinforced with laboratory activities, that allow the student to be familiar with the internal format of data types and the assembly language level [6]. Therefore, this course includes the possibility for the students to perform laboratory activities.

The motivations of the laboratory activities in a computer architecture course are the following: a) high costs of the hardware equipment which increase with the number of workstations; b) problems to keep the laboratory up-to-date with the constant technological innovations (important for processor architecture studies); c) high costs of equipment maintenance d) the difficult to acquire on the market commercial assembly compiler, however not always compatible with current new processors.

On the other side, there are difficulties to follow pedagogical objectives of the course, as follows: a) to introducing progressively the complexity of the modern computer instruction set; b) to observe the relationship between assembly/machine level languages and the architecture; c) to compare the instruction set and performance of different computer type (RISC/CISC), because laboratory components are not so rich and refer to a specific hardware system [7].

At last, if the course must be used in addition to the normal laboratory time and considering that the course is made in teleconferencing, it can't provide to use specific hardware.

In order to overcome the previous limitations, we use a dynamic computer architecture simulation environment, suitable for the pedagogical aims of laboratory activities [8,9].

The course of *computer architecture* provides and integrates two different tools: the *ARCHO course* to support the study of theoretical concepts and exercises and the *ARCHO laboratory* to support laboratory activities using a processor simulator.

4.1 The ARCHO2 course

The contents of the lectures cover topics according suggestions of ACM/CS [10] guidelines for architecture courses and using the popular textbook by A. Tannenbaum [11].

The didactic material have been organized according to course structure examined in the previous Section and following the methodological choice to apply the same learning strategy between the lectures in classroom and the course. So the student can retrieve in the course the same guidelines of the presentations used by the teacher in the lessons. In addition, the navigation in the item of didactic units is organized by opportune links to other items or didactic units that are related to current items or exercises.

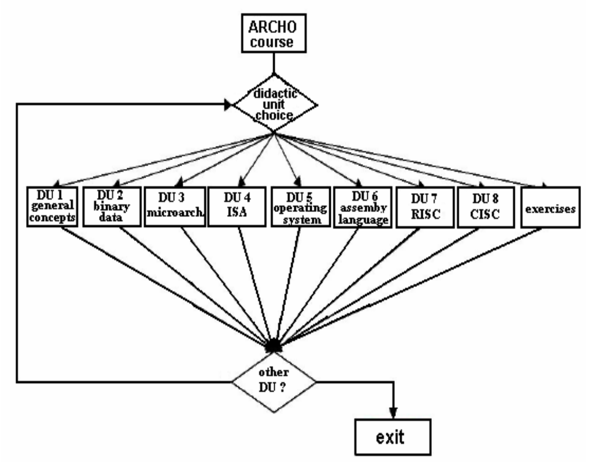


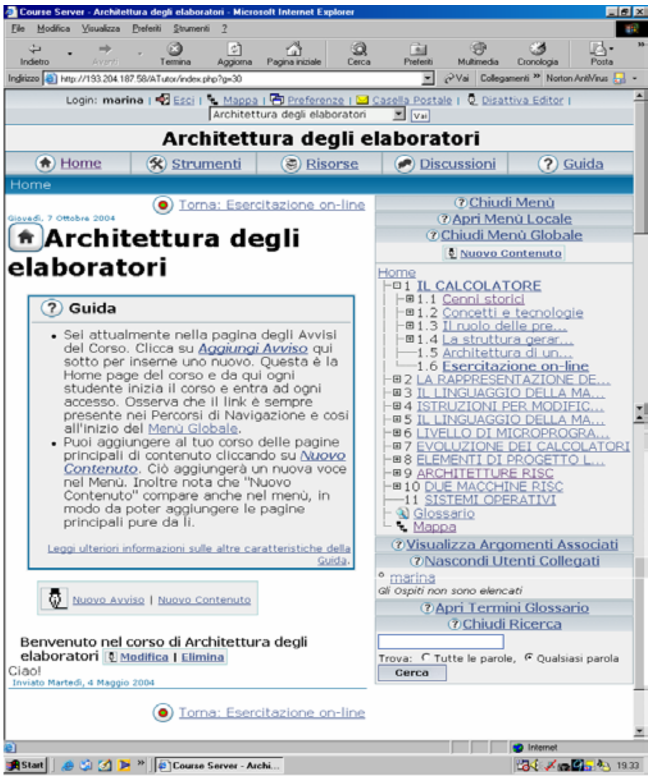
Fig. 4 the structure of the didactic units of ARCHO2

The course is accessible by a general index and presented to the student in a suggested logical order to avoid the getting lost in hyperspace [12] (each unit has a root document, where the student any time can go back)

In figure 4 the structure of the didactic units of ARCHO2 course is reported.

The system is able to store the student's learning path (the visited units, the results of exercises and tests), so that it is possible to recall the learning process starting from the break point. It uses a data base of general information about students, linked with information about the learning progress of each student (tracks on already read didactical units, exercise results and so on…) so as to permit a self evaluation of the obtained learning progress.

Figure 5 shows an example of interaction of the ARCHO2 course.


Course Server - Architettura degli elaboratori - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Avanti Termina Aggiorna Pagina iniziale Cerca Preferiti Multimedia Cronologia Posta
Indirizzo http://193.204.187.58/ATutor/index.php?g=30
Vai Collegamenti Norton AntiVirus
Login: marina | Esci | Mappa | Preferenze | Casella Postale | Disattiva Editor |
Architettura degli elaboratori
Vai
Home
Strumenti
Risorse
Discussioni
Guida
Chiudi Menù
Apri Menù Locale
Chiudi Menù Globale
Nuovo Contenuto
Visualizza Argomenti Associati
Nascondi Utenti Collegati
Apri Termini Glossario
Chiudi Ricerca
Cerca
Start
Course Server - Archi...
Internet
19.33

Architettura degli elaboratori

Home

Torna: Esercitazione on-line

Giovedì, 7 Ottobre 2004

Architettura degli elaboratori

Guida

- Sei attualmente nella pagina degli Avvisi del Corso. Clicca su *Aggiungi Avviso* qui sotto per inserne uno nuovo. Questa è la Home page del corso e da qui ogni studente inizia il corso e entra ad ogni accesso. Osserva che il link è sempre presente nei Percorsi di Navigazione e così all'inizio del Menù Globale.
- Puoi aggiungere al tuo corso delle pagine principali di contenuto cliccando su *Nuovo Contenuto*. Ciò aggiungerà un nuova voce nel Menù. Inoltre nota che "Nuovo Contenuto" compare anche nel menù, in modo da poter aggiungere le pagine principali pure da lì.

Leggi ulteriori informazioni sulle altre caratteristiche della Guida.

Nuovo Avviso | Nuovo Contenuto

Benvenuto nel corso di Architettura degli elaboratori Modifica | Elimina
Ciao!
Inviato Martedì, 4 Maggio 2004

Torna: Esercitazione on-line

Home
- 1 IL CALCOLATORE
  - 1.1 Cenni storici
  - 1.2 Concetti e tecnologie
  - 1.3 Il ruolo delle pre…
  - 1.4 La struttura gerar…
  - 1.5 Architettura di un…
  - 1.6 Esercitazione on-line
- 2 LA RAPPRESENTAZIONE DE…
- 3 IL LINGUAGGIO DELLA MA…
- 4 ISTRUZIONI PER MODIFIC…
- 5 IL LINGUAGGIO DELLA MA…
- 6 LIVELLO DI MICROPROGRA…
- 7 EVOLUZIONE DEI CALCOLATORI
- 8 ELEMENTI DI PROGETTO L…
- 9 ARCHITETTURE RISC
- 10 DUE MACCHINE RISC
- 11 SISTEMI OPERATIVI
- Glossario
- Mappa

marina
Gli Ospiti non sono elencati

Trova: Tutte le parole, [✓ Qualsiasi parola]

Fig.5 Main dialog window of ARCHO course

4.2 The ARCHO2 laboratory

The laboratory activities are performed by APE system, a computer architecture simulator.

The design aim for APE system was to provide the teachers with a tool that allows easily the rapid prototyping of processor simulators, to use in the laboratory activities. For this purpose, the system supports the processor simulator development

providing two steps: the *architecture definition* step and the *architecture test* step.
The object oriented approach has been used to describe the computer architecture: the set of the objects corresponds to the computer structure and their methods implement instructions and addressing methods [8,9]. This approach permits to proceed in an iterative way for developing the processor simulator trough an approach similar to the rapid prototyping

The teacher establishes the processor requirements and, at the definition step, chooses the components of the architecture to simulate (as the type and the number of registers, the attributes of the processor and the memory…) and defines their parameters. The system then constructs a simulator that is the representation of the defined architecture including only those objects necessary to meet the teacher requirements. It serves, in the *test* step, as a work version of the architecture simulator in order to verify the result of the design activity. During the *test* step, the teacher evaluates, by running assembly programs, the simulator's actual behaviour respect of its expected behaviour. If the simulator fails the execution, the teacher identifies the problems and establishes adjustments to the requirements for the architecture definition step repetition.

The APE system builds the desired simulator that students run with assembly programs and displays results of the execution in terms of the processor status.

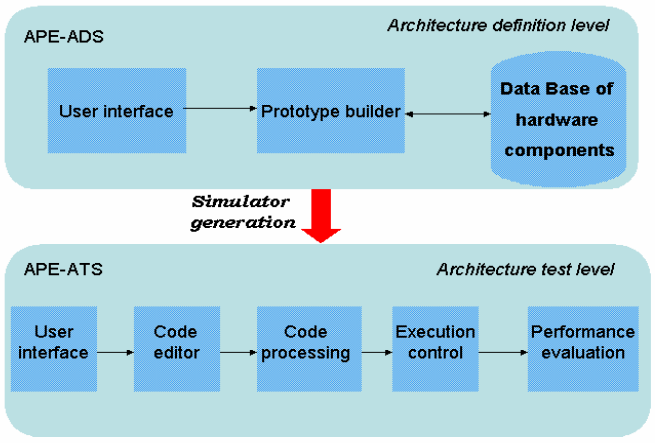


Fig. 6 The APE architecture

The overall structure of APE provides two main subsystems (figure 6), which support the two different user-interaction phases: the *Architecture Definition Subsystem* (APE-ADS) and the *Architecture Test Subsystem* (APE-ATS).

Helped by a graphical interface, the user can perform activities as: writing and translating of programs in assembly language or in symbolic language Micro Assembly Language (MAL), a didactic language for a microprogramming level [11,13]; running and debugging of machine programs and microprograms; displaying of the architecture status both at ISA level and at the microprogramming level; simulation of the performances of the designed architecture at both same levels.

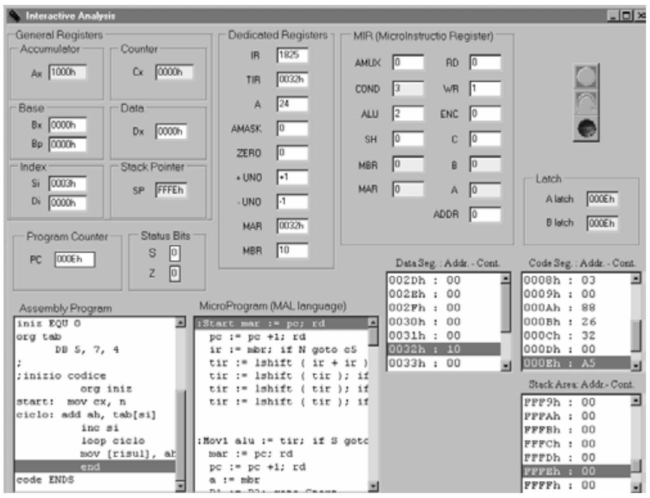
The student interacts with the simulator defined from the teacher at the instruction set level or/and at the microprogramming level. On the basis of the chosen architecture level, the program has to be written in assembly language (ISA level) or in microcode (microprogramming level). In the simulated run time, the simulator displays results of the execution in terms of the architecture component's status (figure 6).

The interaction can be performed from the student in two different ways:

a) through a *single instruction*: the user initialises the components (register, memory, etc.) and writes the instruction to be executed.

b) through a *program*: the user loads a previously saved program into the memory, or edits a new one. The execution of the program can be performed step by step to control the status of the architecture after each instruction.

In both cases, the simulator controls the program syntactically and semantically. The syntactic analysis is aimed at discovering the editing errors. In the semantic analysis, the simulator verifies that the program can be executed with the defined architecture and discovers if some of the program instructions refer to components that the user did not select.



General Registers
Accumulator: Ax 1000h
Counter: Cx 0000h
Base: Bx 0000h, Bp 0000h
Data: Dx 0000h
Index: Si 0003h, Di 0000h
Stack Pointer: SP FFFEh
Program Counter: PC 000Eh
Status Bits: S 0, Z 0

Dedicated Registers: IR 1825, TIR 0032h, A 24, AMASK 0, ZERO 0, +UNO +1, -UNO -1, MAR 0032h, MBR 10

MIR (MicroInstructio Register): AMUX 0, RD 0, COND 3, WR 1, ALU 2, ENC 0, SH 0, C 0, MBR 0, B 0, MAR 0, A 0, ADDR 0

Latch: A latch 000Eh, B latch 000Eh

Assembly Program
```
iniz EQU 0
org tab
      DB 5, 7, 4
;
;inizio codice
          org iniz
start:  mov cx, n
ciclo: add ah, tab[si]
          inc si
          loop ciclo
          mov [risul], ah
          end
code ENDS
```

MicroProgram (MAL language)
```
:Start mar := pc; rd
  pc := pc +1; rd
  ir := mbr; if N goto c5
  tir := lshift ( ir + ir )
  tir := lshift ( tir ); if
  tir := lshift ( tir ); if
  tir := lshift ( tir ); if


:Mov1 alu := tir; if S gotc
  mar := pc; rd
  pc := pc +1; rd
  a := mbr
```

Data Seg. : Addr. - Cont.
```
002Dh : 00
002Eh : 00
002Fh : 00
0030h : 00
0031h : 00
0032h : 10
0033h : 00
```

Code Seg. : Addr. - Cont.
```
0008h : 03
0009h : 00
000Ah : 88
000Bh : 26
000Ch : 32
000Dh : 00
000Eh : A5
```

Stack Area: Addr.- Cont.
```
FFF9h : 00
FFFAh : 00
FFFBh : 00
FFFCh : 00
FFFDh : 00
FFFEh : 00
FFFFh : 00
```

Fig 7. Status Simulation of a CISC architecture

During the execution, the student examines both architectural levels the ISA level and the microprogramming level, as shown in figure 7. In this way it is possible to have a complete control over the execution of the program. In fact, in the step by step program execution, the user could analyse the status of the defined architecture at the end of the execution of each instruction. By switching to the microarchitecture level, the user can perform a step by step execution of the microcode associated to the current assembly instruction.

5. Conclusions

In this paper we have presented an experience of implementation of courses of computer science on a freeware and multiplatform Learning Component Management System developed by the University of Toronto. The example of the presented course integrates a processor simulation within the courseware. In particular it has been profitably used by students of the first year of a degree course in computer science at the University of Bari .

The first results of the system evaluation by student are evidencing some rigidity in the use and in the structure of the exercises and the need of a better support in the use of all the functionalities of the system for each type of potential users. For these reasons we are planning further development in direction of supplying the previously reported limits.

In particular, we will study the possibility to implement information agents simulating on-line tutors which adapt the exercise type to the student learning level.

References:

[1] www.atutor.ca

[2] www.bigwebmaster.com

[3] www.moodle.com

[4] www.ilias.uni-koeln.de

[5] www.claroline.net

[6] D.H. Jonassen, Designing structured hypertext and structuring access to hypertext, *Educational Technology*, vol. 28, no. 1, 1998.

[7] M. De Blasi, and F. Tangorra, Prolog simulation of computer architecture in laboratory activities, *IEEE Transactions on Education*, vol. 35, no. 4, 1992, pp. 331-337.

[8] S. Pizzutilo and F. Tangorra, An Object Oriented Tool to Simulate Multi-Level Computer Architectures, *International Journal of Modelling and Simulation* , vol. 23, no. 1, 2003, pp13-21.

[9] S. Pizzutilo and F. Tangorra, A rapid prototyping environment for designing and simulating multilevel computer architectures, *Simulation*, Vol. 78, no 8, 2002, pp 512-525.

[10] ACM/IEEE Computer Society, Computing Curricula,1992.

[11] A.S. Tannenbaum, *Structured computer organization*. Prentice-Hall International, Inc., third edition, Englewood Cliffs, NJ, 1990.

[12] D.M. Edwards and L. Hardman, Lost in Hyperspace: cognitive mapping and navigation in a hypertext environment, Hypertext: theory into practice, Ablex Publishing Corporation, 1989.

[13] J. Donaldson, A Microprogram Simulator and Compiler for an Enhanced Version of Tannenbaum's Mic1 Machine, 26th SIGCSE Techn. Symp. on Comp. Science Education, *ACM SIGCSE Bulletin*, *27* (1), 1995, pp. 238- 242.