FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 18, No. 2, August 2005, 263-274

A Virtual Lab Model for an Introductory Computer Science Course

Ioanna Kantzavelou

Abstract: This paper presents a model of a virtual laboratory for an introductory computer science course. The proposed model aims at solving a number of problems involved in the educational procedure of such a course. The model architecture consists of seven modules, each one corresponds to a specific topic of the course. Every module provides several different services in order to assist students to assimilate theory with practical exercises. Preliminary results of partial implementation of the proposed model, show the solution of some problems and better understanding of abstract concepts.

Keywords: Virtual Lab, Computer Science, introductory course.

1 Introduction

The ACM/IEEE curriculum '01 states clearly the intense debate for the structure of the introductory computer science course. Therefore, the Joint Task Force on Computing Curricula has chosen not to recommend any single approach [1]. On the contrary, they described the pros and cons of programming and its position in the introductory curriculum, the length of the introductory sequence known as CS1 and CS2, and six implementation strategies for introductory computer science. The corresponding curriculum for Computer Engineering, recognizes introductory courses as extremely important, because they are the first courses that students encounter [2].

Most Computer Science departments include an introductory course to Computer Science in their undergraduate syllabi. This introductory course might have one of two different forms. The first form provides the fundamental elements of

Manuscript received February 28, 2005
The author is with Technological Educational Institution of Athens, Department of Informatics, Greece (e-mail: ikantz@teiath.gr).

information technology, making the students skilled to attend and be trained to the majority of the remainder of the courses of the curriculum. This form is usually called *Introduction to Computer Science*. The second form aims at making students able to use a computer by tutoring them in the most useful and elementary tools for word processing, internet services (eg. browsing, e-mail), spreadsheets, presentations, etc. The latter form is known as *Computer Literacy*. Although several software products can support the second form, there is a lack of an integrated software to support the first form of courses.

Some of the problems involved in the educational procedure of an introductory computer science course are listed below.

- Today's students are already familiar with several computer services (eg. e-mail, internet, etc.). Therefore, they come to the classroom with the impression that an introductory computer science course covers merely what they already know and that such a course has nothing special to add to their knowledge. For this reason, there is a need for students' awareness concerning the context of an introductory computer science course. A virtual lab aims at attracting students interest and creating motivations that stimulate this interest.
- An introductory computer science course should include lab exercises to help students to take fully into their mind and experience effects of elementary computer science knowledge.
- Due to the diverse curriculum of an introductory computer science course which forms a collection of several different topics, laboratory exercises require many different software and hardware tools (eg. an instrument for the construction of digital circuits, hardware and software to support different operating systems and networks, interpreters, compilers and linkers of different programming languages, etc.) Therefore, a virtual lab consisting of tools and simulators can effectively replace the use of the corresponding necessary equipment and provide equivalent services to support students.
- Physical labs are usually weekly scheduled two-hour and staffed by teaching assistants (TA) to instruct students to accomplish their exercises and to solve their problem sets. Students who might want to use a physical lab more often, complain about this time limitation. Virtual labs are a solution, providing unlimited use of the offering services.

2 Related Work

In 1990, Daniel Joyce from Villanova University, presented a virtual lab to support CS1 and CS2 courses [3]. This virtual lab was in place of a physical lab without

a room reservation nor scheduled hours. Joyce described in detail the reasons to implement a virtual lab, its structure and its topics which are merely programming oriented. Finally, he evaluated the use and success of this lab by questioning the students and received satisfactory positive comments.

In 1992, Baldwin and Koomen mentioned the lack of exercises in introductory computer science courses, although the ACM/IEEE curriculum '91 had already suggested the use of experiments in several laboratory courses [4]. In order to fill this gap, they proposed the use of scientific experiments as valuable teaching tools in introductory computer science courses, analogous to experiments used in other sciences (biology or physics). They described a number of benefits and minimal drawbacks encountered by the application of the proposed framework in CS1 and CS2 courses, dedicated to programming fundamentals. A similar approach has been proposed by Doran and Langan in [5] for introductory computer science courses.

In 1999, Poindexter and Heck triggered the subject of integrating the internet into courses [6]. They described virtual labs as sophisticated interactive demos and proposed them as a good substitute of physical labs in case the electronic lab material is accompanied by animation. Along with these attempts, distance learning has become more widely accepted. Very recently, Colace et al. described roughly the architecture of the SimBAD environment [7], to support Electrical and Electronics Measurements courses. This distance learning dedicated approach aims at solving the problems of the increasing number of students who access the university educational structures and the high cost of lab maintenance. Similarly, Ciubotariu et al. are currently working in developing a virtual lab for distance learning to support electrical, computer, and software engineering curricula [8]. They have already implemented some components, one related to digital logic (flip-flops) and another one related to a traffic light controller. Finally, Weaver describes in [9] the development of a programming laboratory for e-commerce. He lists the goals of this effort, the lecture topics, and a brief demonstration of his virtual lab.

Since 2000, the World ORT organization has developed a complete Information Technology foundation course, the DO.I.T [10], that covers the topics of Hardware, Software, Computer Languages, Developing an Information System, Information Representation, Networks, and the Internet. The DO.I.T's target group might have incomplete or no knowledge of computer science, learn by viewing pictures and reading text, and finally examine themselves through a number of questions associated with answers.

Currently, ACM is intending to create a new organization to help teachers and students to be aware and increase the value of career opportunities motivated by computer science [11]. The organization will be a national computer science com-

munity to support computing education for students age 5-18 (K-12). Among the current working projects supported by volunteer educators is the development of a national Web-based repository of teaching and learning materials for K-12 CS courses. The main goal of this attempt is to keep alive the inspiration of interest and excitement generated by the first experience of computing.

3 The Virtual Lab Model Architecture

This approach aims at achieving the following objectives:

- a collection of tools and programs in a virtual lab,
- to assist the educational process of an introductory computer science course,
- to increase the lab course time to 24/7, but without closing down the physical lab, nor making redundant the teaching assistants,
- to help students to acquire concrete knowledge for abstract concepts by experimentation, and
- to attract students interest and create motivations that stimulate this interest.

The proposed Virtual Lab Model is composed of seven modules which have been selected according to the described concepts covered in the introductory curriculum [1]. These modules are:

1. Gates and Circuits (GC)
2. Data Representation (DR)
3. Main Memory (MM)
4. Computer Architecture (CA)
5. Algorithms (AL)
6. Programming Languages (PL)
7. Operating Systems and Networks (OSN)

Description of the role and the functioning of each module in the Virtual Lab Model is provided in the next seven subsections. Figure 1 depicts the architecture of the proposed model.

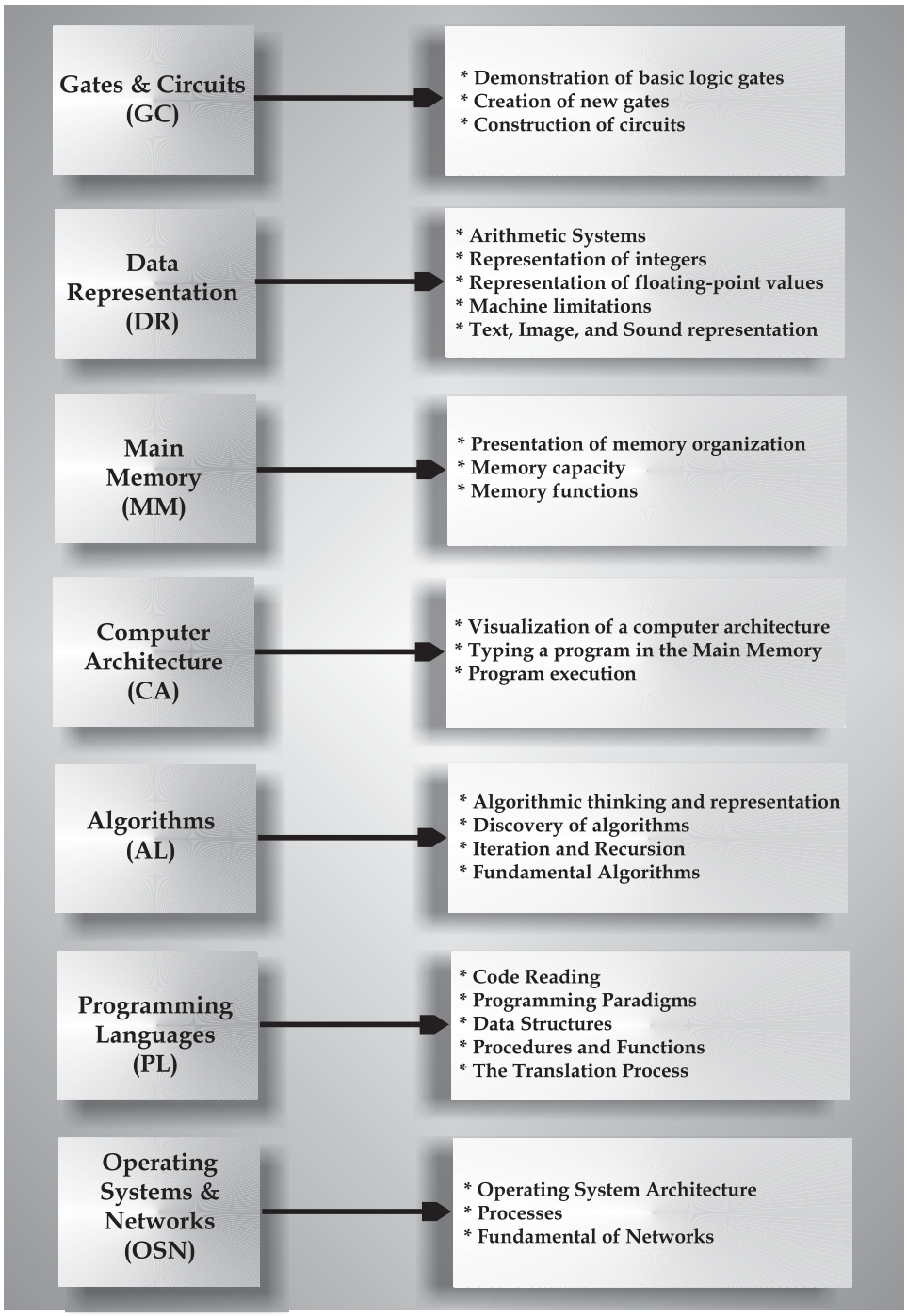

Fig. 1. The Virtual Lab Model Architecture and its seven modules with the corresponding provided services.

3.1 Gates and Circuits

The Gates and Circuits (GC) module covers the digital logic concepts by providing the following services:

- *Demonstration of basic logic gates*: It shows the functionality of elementary gates by presenting their figure, the number of inputs and outputs as well as their values.
- *Creation of new gates*: It allows the creation of user-defined gates by selecting a figure from a collection of given figures, defining the number of inputs and outputs and finally by identifying the values of these inputs and outputs.
- *Construction of circuits*: It allows the construction of circuits by using existing and user-defined gates showing the way a circuit works. Special circuits, as flip-flops are supported and explained in detail.

3.2 Data Representation

The Data Representation (DR) module covers the concepts of computer arithmetic by providing the following services:

- *Arithmetic Systems*: It familiarizes users with the use of other arithmetic systems than the known decimal system. The binary and the hexadecimal systems are mainly presented and conversion tables for every two systems are displayed to help users to understand better the data storage into a machine.
- *Representation of integers*: It describes methods for the representation of positive and negative integer values (two's complements and excess notation), instructing users to accomplish related exercises.
- *Representation of floating point values*: It describes the representation of positive and negative floating point values considering a specific notation, and instructs users to accomplish related exercises.
- *Machine limitations*: It instructs users to determine the lowest and highest number values for the specific machine, and shows examples of the overflow problem and truncation errors.
- *Text, Image, and Sound representation*: It demonstrates the text, image, and sound coding through a series of examples and instructs the users to distinguish between different file types.

3.3 Main Memory

The Main Memory (MM) module covers concepts related to the Main Memory of a machine and provides the following services:

- *Presentation of memory organization*: It familiarizes users with the organization of a main memory in general, showing the arrangement of its cells and associating the data representation services described in 3.2.
- *Memory Capacity*: It describes measures for the capacity of main memory, introduces the new proposed measures *kibi*, *mebi*, and *gibi*, instructing users to accomplish related exercises.
- *Memory functions*: It clarifies and visualizes the most important functions carried out in main memory by showing a series of examples and explains fundamental concepts related to the data storage topic.

3.4 Computer Architecture

The Computer Architecture (CA) module is a simulator of a generic computer architecture that works with a specific virtual machine language with only twelve fundamental instructions. It is based on the classical John von Neumann architecture, and provides the following services:

- *Visualization of a computer architecture*: It presents the heart of a computer architecture in a visual environment by picturing some of the cells and their addresses of the Main Memory, the general purpose registers, the program counter, the instruction register, and the Arithmetic and Logic Unit.
- *Typing a program in the Main Memory*: It allows a user to enter into the memory cells the instructions of a program of his own, written in the specific virtual machine language associated with this architecture.
- *Program execution*: It executes the program stored into the Main Memory by showing step by step the machine cycle (fetch, decode, execute). Throughout these steps, the changes made inside the machine, i.e. the contents of the program counter, the registers, the instruction register and the memory cells are recorded and updated to show the current view of the machine as the result of the program execution.

3.5 Algorithms

The Algorithms (AL) module covers the topic of algorithms, and provides the following services:

- *Algorithmic thinking and representation*: It introduces the elementary issues of algorithms and problem solving as models of computational processes through visualized examples. It instructs users to strategies of algorithmic thinking and the concept and representation of algorithms by a series of examples and exercises.

- *Discovery of algorithms*: It allows a user to discover algorithms of selected problems and represent them using knowledge acquired from the previous service, and verifies the efficiency and correctness of the user-defined algorithms.
- *Iteration and Recursion*: It presents the structure of the major controls, iteration and recursion, and allows a user to become skilled at creating algorithms which include these controls.
- *Fundamental algorithms*: It instructs users to fundamental algorithms of sorting and searching, and allows them to analyze and evaluate the characteristics of their strategy and the corresponding performance.

3.6 Programming Languages

The Programming Languages (PL) module covers the subject of programming languages by providing the following services:

- *Code Reading*: Code reading is a key method of learning how to write code [12]. This service presents useful tools and sources for code reading, and helps users to create a plan of how to be trained to any programming language.
- *Programming Paradigms*: It presents existing programming paradigms (procedural, declarative, functional, and object-oriented) and allows users to understand the relationships and differences between several programming languages with the use of visualized examples.
- *Data Structures*: It introduces primitive data structures, supports users in distinguishing arrays, records, strings, etc., and connects this subject with the corresponding described in 3.3 to show the data representation into the main memory.
- *Procedures and Functions*: It presents a variety of examples in different programming languages and allows users to comprehend concepts of breaking a program apart into procedural units, their call and parameter passing.
- *The Translation Process*: It presents the steps of the translation process by showing the different phases and the intermediate steps for the production of an executable program; it allows users to work with numerous compilers and interpreters of various programming languages and perform comparison tests between them.

3.7 Operating Systems and Networks

The Operating Systems and Networks (OSN) module covers the subject of operating systems and their actual expansion to computer networks, by providing the following services:

- *Operating System Architecture*: It presents the main components of a generic operating system and relates them to real operating systems by visualization of selected processes such as the booting process.
- *Processes*: It demonstrates the distinction between programs and processes, displays examples of processes present in single and multiple processor systems and their administration by the operating system, and allows users to tackle case studies of process competitions as deadlocks.
- *Fundamentals of Networks*: It presents ways of network classification, basic network equipment, the Internet and its services, and allows users to assimilate theory by accomplishing a series of exercises.

4 Preliminary and Expected Results

The Technological Educational Institution of Athens operates on a semester system. The introductory computer science course of the Department of Informatics is divided into the lecture course (2h/w) and the lab course (2h/w). The entering students attend also one course for Algorithms and an introductory programming course.

Last semester, the lab part of the introductory computer science course was supported by a collection of selected tools found at various sites. The topic of gates and circuits was supported by the circuit builder developed by the Johns Hopkins University as part of a virtual engineering laboratory [13]. The topic of data representation was supported by a number of related sites which offered exercises and problem sets. Finally, a virtual machine simulator (Figure 2) was implemented and instructed to entering students who experienced a generic computer architecture together with its corresponding virtual machine language.

The results showed that students assimilated digital logic, data representation and the structure and the functioning of a simple abstract computer architecture better and intensively. This consequence was verified by the theory exam results which increased the number of passed marks by 20%.

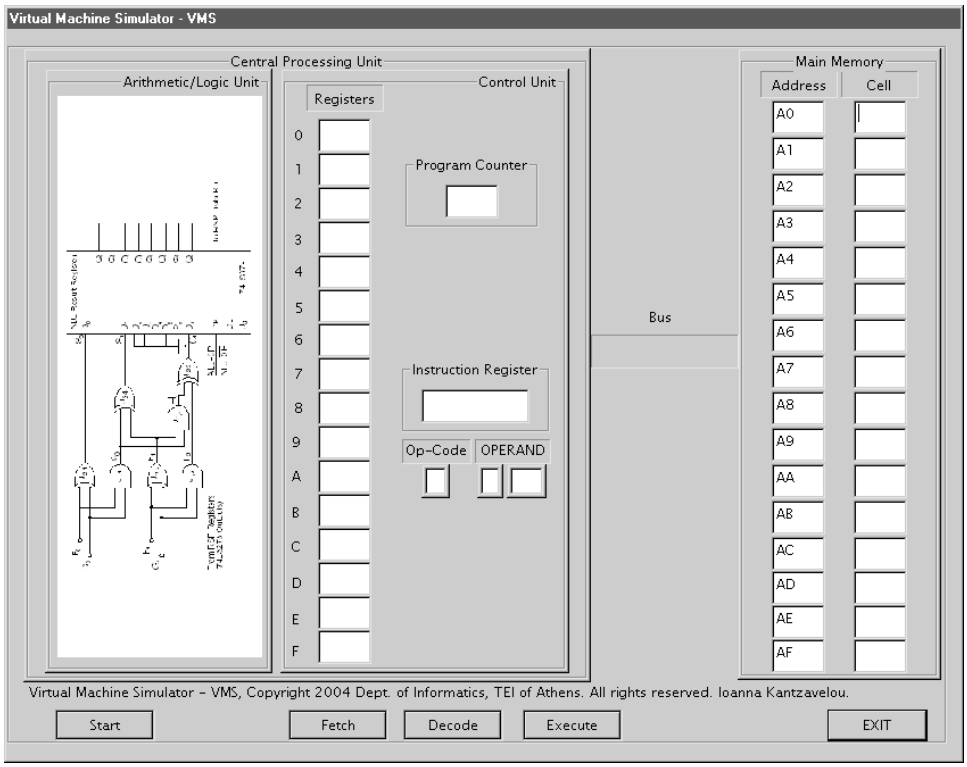

Fig. 2. The Virtual Machine Simulator (VMS) of the Computer Architecture Module.

5 Future Work

The proposed model should be fully implemented in the near future. An appropriate implementation platform will be first selected, associated with the corresponding tools to ensure a working product. In addition to the described model components, other topics such as, the theory of computation, artificial intelligence, data base structures, and security issues might be considered to expand the proposed architecture.

6 Conclusion

Although there is not a general consensus about the syllabus of an introductory computer science course, a virtual lab model has partially been developed to assist the educational process. This model is under development and based on preliminary results, the expected results are promising. The model might be expanded in

the future to include additional topics, and it might form the foundation for implementations to support other courses.

7 Acknowledgements

I would like to thank my TAs, Andreas Zoupas and Markos Plytas, who have instructed the students to use the virtual lab model, and the students who have enthusiastically endorsed the project.

References

[1] The Joint Task Force on Computing Curricula, "Computing curricula 2001 - Computer Science," IEEE Computer Society and Association for Computing Machinery," Final Report, Dec. 15, 2001.

[2] ——, "Computing curricula 2004 - Computer Engineering," IEEE Computer Society and Association for Computing Machinery," Final Curriculum Report, Dec. 15, December 12, 2004.

[3] D. T. Joyce, "A virtual lab to accompany CS1 and CS2," *ACM SIGCSE Bulletin*, vol. 22, no. 1, pp. 40–43, Feb. 1990. [Online]. Available: http://portal.acm.org/citation.cfm?id=319059.319077&coll=GUIDE&dl =ACM& CFID=39556168&CFTOKEN=53247351

[4] D. Bladwin and J. A. G. M. Koomen, "Using scientific experiments in early computer science laboratories," *ACM SIGCSE Bulletin*, vol. 24, no. 1, pp. 102–106, Mar. 1992. [Online]. Available: http://portal.acm.org/citation.cfm?id=135250.134532&coll=GUIDE&dl =GUIDE& CFID=38926914&CFTOKEN=88331108

[5] M. V. Doran and D. D. Langan, "A congnitive-based approach to introductory computer science cources: Lessons learned," in *Proc. of the SIGCSE Technical Symposioum on Computer Science Education*, Nashville, TN USA, March 1995, pp. 218–222.

[6] S. E. Poindexter and B. S. Heck, "Using the web in your courses: What can you do? what should you do?" *IEEE Control Systems Magazine*, Feb. 1999.

[7] F. Colace, M. D. Santo, and A. Pietrosanto, "Work in progress - virtual lab for electronic engineering curricula," in *34th ASEE/IEEE Frontiers in Education Conference T3C*, Savannah, GA, Oct. 20–23, 2004, pp. 22–24.

[8] C. Ciubotariu and G. Hancock, "Work in progress - virtual laboratory with remote control instrumentation component," in *34th ASEE/IEEE Frontiers in Education Conference T3C*, Savannah, GA, Oct. 20–23, 2004, pp. 18–19.

[9] A. C. Weaver, "A programming laboratory for electronic commerce," in *34th ASEE/IEEE Frontiers in Education Conference T3H*, Savannah, GA, Oct. 20–23, 2004, pp. 25–30.

[10] World ORT. (2000) Information technology foundation cource. electronically. World ORT. [Online]. Available: http://doit.ort.org/

[11] C. Stephenson, “Creating a national k-12 computer science community,” *Communications of the ACM*, vol. 48, no. 1, pp. 29–31, Jan. 2005.

[12] D. Spinellis, *Code Reading - The Open Source Perspective*. Addison-Wesley, 2003.

[13] Johns Hopkins University. Virtual laboratories - experiments. electronically. [Online]. Available: http://www.jhu.edu/ virtlab/virtual-laboratory/