An Online Environment Supporting High Quality Education in Computational Science

Luis Anido, Juan Santos,
Manuel Caeiro, and Judith Rodríguez

Grupo de Ingeniería de Sistemas Telemáticos
ETSI Telecomunicación
University of Vigo, SPAIN
{lanido, jsgago, mcaeiro, jestevez}@det.uvigo.es

Abstract. Education in Computational Science is a demanding task, especially when online support is required. From the learner's point of view, one of the most challenging issues is to get used to the supporting tools, especially with the computing devices employed (e.g. DSPs). For this, students need to understand concepts from the Computer Architecture area. This paper presents a WWW-based virtual laboratory that provides learners with a suitable environment where Computer Architecture concepts, needed to face Computational Science education, can be acquired. Our contribution can be described as an open distributed platform to provide practical training. Services offered support the seamless integration of simulators written in Java, and include features like student tracking, collaborative tools, messaging, task and project management, or virtual file repositories. A CORBA-based distributed architecture to access real computer systems, available in labs at University facilities, is also described.

1 Introduction

Computer Architecture has some specific properties that makes it a challenging matter when trying to provide adequate supporting tools for e-learning. While the implementation of the first phases of the learning process in this field, typically through theoretical lectures, poses no major problems today given the help provided by available authoring systems and course tools and resources, remote laboratory settings are far more demanding. Student interaction with virtual equipment (i. e. simulators) or remote devices like Digital Signal Processors (DSPs) and complex computing equipment should be guaranteed to adequately support virtual presence, which demands efficient transmission protocols. Note that, to acquire the skills needed to adequately interact with Computational Science related equipment, hands-on experience is a must. Additionally, to provide an adequate learning environment, interactions among students and lecturers should be also supported, which poses a need for distributed communication services.

With respect to the target equipment themselves, typically virtual devices or remotely accessed laboratory premises, their implementation and configuration

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 872–881, 2002.

is not a simple task either. They should be robust beyond their standard, non pedagogical versions, and they should support educational-oriented features like guided operation, improved fault tolerance or activity logging for further study by lecturers.

In this paper we describe our approach to virtual laboratories in the field of Computational Science, whose main objective is to make students gain acquittance with the complex hardware and software tools used in the field. Our contribution can be described as an open distributed platform to support practical training. Services are offered to easily integrate third-party developed educational simulators. More specifically, they support the seamless integration of simulators written in Java, and include features like student tracking, collaborative tools, messaging, task and project management, or virtual file repositories.

Additionally, we propose a CORBA-based distributed architecture to access real computer systems, available in labs at University facilities, which can be seen as a complementary approach to that based on simulation.

All these approaches can be easily integrated to provide a true practical distance learning experience.

2 Teaching Computer Architecture

Distance practical training in Computer Architectures is not straightforward. Hands-on practice is a must to adequately grasp the fundamental concepts in this field. Skills on machine and assembly programming, microprocessor architectures and low-level computer communications can only be adequately obtained when the students test their own developments and see how computers evolve as directed by them. For this, we have to develop an adequate environment to offer practical training, taking into account lessons learnt from previous experiences. Current approaches to practical training over the Internet can be classified into two groups [1]: (1) those based on educational simulations and (2) settings that provide access to real laboratory equipment. In our case, for the former approach students would be provided with simulators of computer systems to study the simulated behaviour of the real equipment. In the latter case, students would control real computer systems, placed in academic institutions, via the Internet.

In our case, distance learning in this field is intended to provide students the needed skills to take the most of available equipment in Computational Science labs (e.g. DSPs, Single Board Computers, etc.). As a general rule, simulation is adequate for first- and second- year students. There are many educational simulators of many different architectures available, and for this target group a simulator provides additional pedagogical advantages, like a safe, controlled environment, ability to easily reconstruct student activities to analyze errors, or guided simulation.

On the other side, access to the real thing is targeted to more advanced students. For them, practice with real computer systems is a natural step ahead after introductory courses on the matter.

3 SimulNet: Simulators over the Network

SimulNet [2] provides students with a teleteaching environment where the "learning-by-doing" paradigm is possible. Unlike other distance teaching systems whose aim is to achieve a virtual classroom, SimulNet provides a virtual laboratory to put theoretical knowledge into practice. Because SimulNet is a 100% pure Java system, our labware can be run on any computer and operating system. Our approach is based on the simulation of the actual laboratory tools that are delivered through the Internet (Java applets) or by CD-ROM technology (Java applications). Although SimulNet can be used in a remote access way, Java allows us to provide always the highest level of interactivity, which is an essential feature in any distance education system. In addition, SimulNet also provides a set of communication and tutoring tools for learners and instructors, providing a full cooperative learning atmosphere. We believe distance education should not mean to study alone and, therefore, we made an extra effort to provide an environment where students and teachers feel as is they were in a virtual lecture room.

3.1 Architecture

The implementation of SimulNet is based on currently available Internet technologies, especially in the Java computing. Thanks to the Java computing technology we have achieved several important features in SimulNet:

- Platform independence. SimulNet client and server applications may be run on any computer whatever its operating system or architecture is.
- Java eases the development of WWW-based applications. SimulNet provides a WWW client (based on Java applets) and a stand alone client (based on Java applications) delivered by CD-ROM, see Fig. 1.

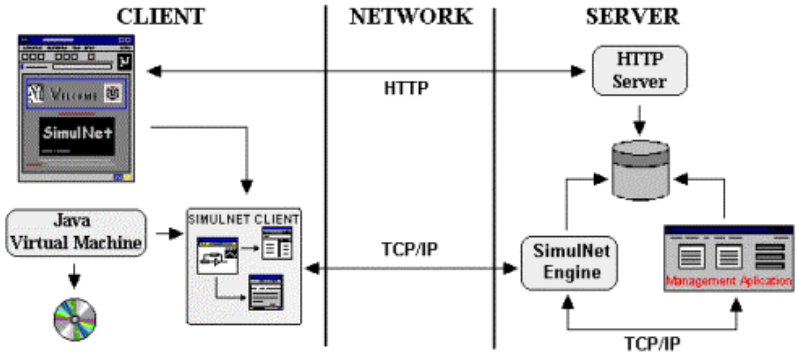

Fig. 1. Simplez Structural Model

The simulators are interactive. There is no network overhead as the simulators run on the student's own computer. This is an essential feature to provide to trainees the same feeling as if they indeed were in the laboratory using the real training tools.

Users access SimulNet using a standard WWW browser, which presents an HTML document provided by the HTTP server at the server side. This document contains an embedded Java applet that would start and stop any other Java application provided by the server side through the Internet. In this way, no additional software is required apart from the browser itself.

Alternatively, students can use a fully independent Java application delivered by CD-ROM. In this case, the SimulNet simulators could be used in a standalone way. At the same time, the user could connect the Java application to the server side to benefit from the virtual laboratory advantages: communication channel, trainees' traces, etc.

3.2 Collaborative Learning

SimulNet provides several communication tools to support cooperative learning. All of them were developed from scratch by our team, and they can be easily included in new simulators thanks to the SimulNet API. In this way, students are provided with an easy-to-use set of collaborative facilities to ease learning on Computational Science or any other field. On the one hand, there is no need to set up or use additional software at client computers, which may be difficult for trainees. The needed software is automatically downloaded from the network and can be run wherever the user may be, regardless of the particular hardware platform or operating system. The developed communication tools are (c.f. Fig. 2): Mail Tool, Bulletin Board, Talk, Multi-Talk, Whiteboard and Project Management Tool.

3.3 First steps in Online Computational Science Education: acquiring experience with supporting and related concepts

Our platform provides a set of tools to acquire the needed experience with computational devices. For this, we have used pedagogical computers that are good enough to explain and practice with the fundamental concepts needed to manage more elaborated artifacts used in Computational Science Education.

We have developed simulators of the computers described in [3], from the simplest one to the most complex. Thus, students are guided in the right way, starting from a simple assembler language and ending with the complex world of microinstructions, firmware and datapaths. In the following subsections we introduce the main features of three pedagogical computers: Simplez, Simplez+i4 and Algoritmez.

All of our simulators have several common features: an editor, an assembler and the simulator of the computer itself. Programs written by students in assembler code using the editor are transformed into object code that can be run by the computer model. This task is done by the assembler. It also offers information

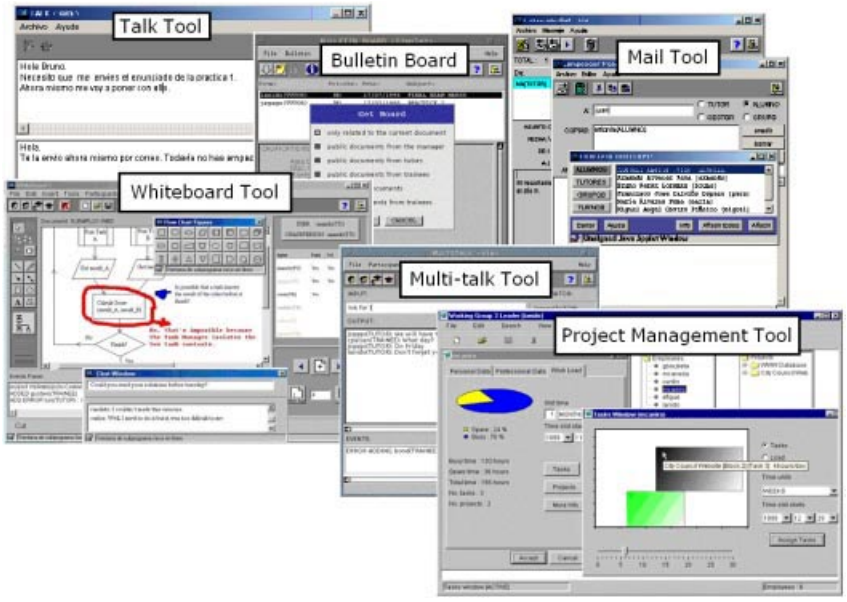

Fig. 2. SimulNet Communication Tools

about different labels and constants used by the student. The simulator of the computer model executes the object code, which can also be displayed through the memory viewer. It has several execution modes: complete run, trace, step by step. Students can also use breakpoints or modify memory or registers at any time. All of these are common features the student must be familiar with in order to be able to manage himself properly in Computational Science labs.

Simplez Simplez has 512 memory locations, 1 register, 8 instructions, I/O mechanism via memory mapped I/O and 1 addressing mode (direct addressing mode). With this architecture, trainees are shown how to implement basic algorithms and how difficult could be to implement the more complex ones. The Simplez simulator embodies an editor, an assembler, the processor simulator itself, a code viewer and the Simplez monitor.

Simplez+i4 Simplez+i4 is a more complex computer. It is the next step in our students learning process. This computer is based on Simplez but adding three different addressing modes: indirect (the first i), indexed (the second i) and indexed+indirect (the third i). It also includes a simple interrupt mechanism (the fourth i) to implement communication with two peripherals (keyboard and monitor). Its conventional machine level incorporates an index register, 4096 memory locations and a more elaborated format for the eight instruction set. After students have acquired the fundamental concepts of Simplez operation,

they are able to be introduced to new ones with a higher level of difficulty such as computer interruptions and addressing modes.

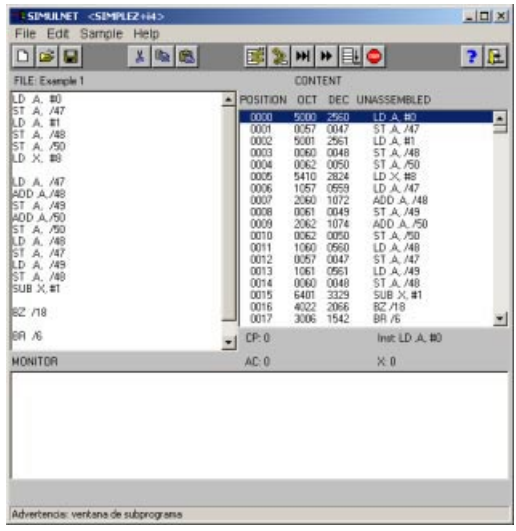

Fig. 3. The Simplez+i4 simulator

Algoritmez Algoritmez is a pedagogical computer closer to commercial ones. Its structural model presents a 64K local memory (including a stack), 256 I/O ports, 16 different registers (two of them used as the program counter and the stack pointer), a status register with several flags and 54 different instructions (related to the arithmetic and shift units, access to stack and memory locations, I/O and, of course, flow control). Through the functional model that we set up over this structural model, students gain further insight into the actual behavior of computers.

Furthermore, the simulator of Algoritmez gives the possibility of microprograming. The implementation of a computer model at micromachine level can be internally microprogrammed allowing designers to easily change its contents and, thus, adapting the system to different instruction sets or data paths. Students are allowed to use the Algoritmez's data path and microprogram the instruction set. In this way, a higher degree of abstraction is offered and the student is not restricted to a fixed model. We usually redefine many of the characteristics of our pedagogical computer to let students interact with different (virtual) machines that are emulated using the own Algoritmez (simulated) hardware.

We include a datapath viewer, see Fig. 4, that shows the connections among the different parts of the computer and let the user to check the signals that are being generated by the control unit, how data flows through the buses and

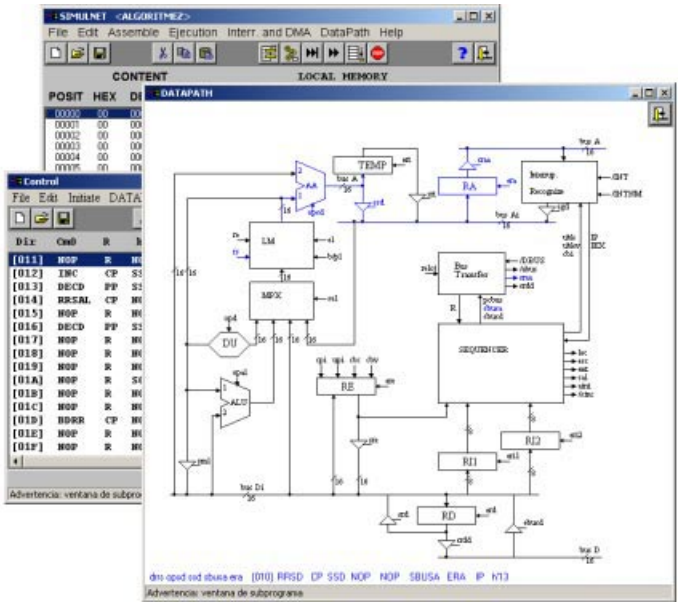

Fig. 4. The Algoritmez simulator

arrive to registers, memory locations, etc. This machine is complex enough to let students understand the more elaborated concepts behind the computational resources used in most Computational Science labs.

3.4 Tutoring

The most experienced teachers provide information about what actions performed on the simulator should be considered as important from a pedagogical point of view. These action will be reflected in students' traces. Whenever a student performs a given task, a report is sent to the tutor responsible for monitoring his or her actions (c.f. Fig. 5). So, without being in the same room, the teacher is able to follow students' performance and, if necessary, to teach how to do the training practice via several available communication tools, see section 3.2.

4 Accessing to real equipment

Although it is possible to use simulation to teach many practical skills to students, there exist several situations where the use of the real equipment is compulsory: either the development of a simulator from scratch is not feasible or real industry equipment is too complex to simulate. In this case, in order to design a distance education environment, we need to manage real instruments

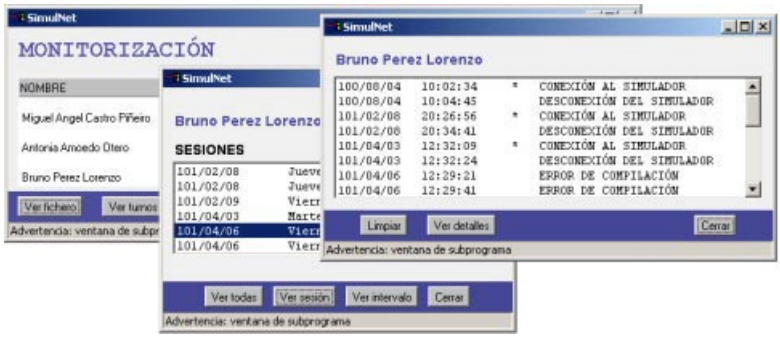

Fig. 5. Monitoring students' behaviour

and equipment remotely. We have developed a system that is centered in this remote operation context: we have to carry out the experiment as if students were in the actual laboratory, and we need to provide them with the output and results of every action, command or modification, as the experiment takes place.

In our introductory laboratory to Computational Science, students are provided with a Java/CORBA-based environment [4] where real DSP devices can be controlled remotely via the Internet. Apart from student accessibility advantages, this solution generates important savings for the institution responsible for maintaining laboratory facilities, both in equipment and staff.

4.1 Description of the environment

In order to acquire the needed experience in DSP programming, students have full access to the real computer where Flite Electronic's DSP25 Cards [6] are used. These cards includes a TMS320C25 DSP chip programmed by students to put theoretical concepts into practice.

DSP programming learning would not be feasible without a proper theoretical introduction. For this, we have included as part of our environment an online course. After this introduction, learners are supposed to be able to deal with the real DSP using the virtual lab.

In this virtual lab learners access DSP25 cards using a WWW browser. This client provides a complete development environment, since it handles all applications used in the conventional laboratory (editor, assembler, debugger, etc.) First practices are typically quite simple as its aim is just to familiarise with the working environment [7]. Last steps consist of developing a small project cooperatively among a group of partners (SimulNet provides both communication and task-based learning supporting tools). Typical practices would be the development of digital filters. Eventually, in the very last phases they need to use the conventional lab facilities where they test the programmes developed using the analog instruments available at the lab (signal generators, oscilloscopes, etc.)

4.2 System Architecture

Our remote access system is based on the distributed objects paradigm, and specifically in CORBA [4]. Using CORBA, we can create object-based distributed applications in a simple and easy way, with all the advantages of distributed object-based programming. The overall system architecture is depicted in Fig. 6. There are several modules that can be clearly identified:

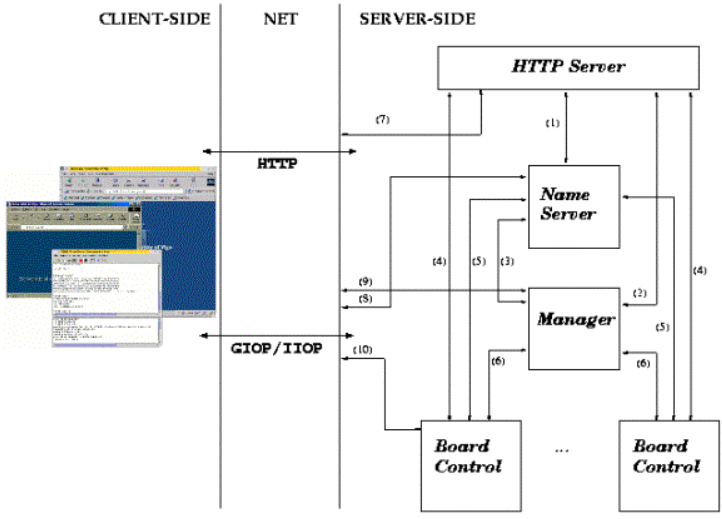

Fig. 6. System architecture

- Client. It is the application that wants to establish remote access to a DSP25 board. It must negotiate with server-side processes to get a free board, and with board-control processes to actually access the board and use additional features.
- Name server. It acts as a bridge used by clients and board-control processes to access the system.
- System manager process. It is responsible for two different tasks. The first one is registering new boards in the system. The second is guaranteeing that, in case there is an available board, any authenticated client can use it.
- Board-control processes. Each of them controls a DSP25 board and other resources accessible by the client in the server file system (files, I/O, execution of programs, etc.).
- HTTP server. It is used as a main door to the system, used to download WWW laboratory pages and also to access the name server.

With this configuration, we can obtain savings up to 84% comparing to the real lab [5]. A reduction of the system availability allows this savings. We use the

fact that in a virtual laboratory, with no schedules at all, students will not access the system simultaneously. Although there might be some access rejections at peak time, our experience demonstrates savings are worth enough as the rejection rate is low.

5 Conclusions

We have presented our experience in designing practical training support for remote learning in the field of Computational Science. The systems described allow students to gain expertise in the use of complex computing devices, like the ones they will encounter in laboratories in the field. Two approaches have been discussed: training through educational simulators of real systems, and remote access to the systems themselves, including additional supporting services to provide pedagogical added value.

We think that this distance learning is particularly adequate in this case, because our main objective is to provide previous hands-on experience to students that will have to interact with real computing equipment in regular courses in Computational Science. In this sense, e-learning serves as a complement to the target audience, where lectures can be taken independently of time constraints or physical location. These complementary remote courses will enable students to get the most from the corresponding regular courses.

References

1. Anido, L., Llamas, M., Fernández, M.J.: Internet-based Learning by Doing. IEEE Transactions on Education, Vol. 44, No. 2, CD-ROM Folder 09. ISSN 0018-9359 (2001)
2. Llamas, M., Anido, L., Fernández, M. J.: Simulators over the Network. IEEE Transactions on Education, Vol. 44, No. 2, CD-ROM Folder 09. ISSN 0018-9359 (2001)
3. Fernández, G.: Conceptos básicos de Arquitectura y Sistemas Operativos. Curso de Ordenadores. Sistemas y Servicios de Comunicación, S.L. ISBN 84-605-0522-7.
4. Harkey, D., Orfali, R.: Client/Server Programming with Java and CORBA, 2nd Edition. John Wiley & Sons. ISBN 047124578X.
5. Castaño, F.J., Anido, L., Vales, J., Fernández, M.J., Llamas, M., Rodríguez, P.S., Pousada, J.M.: Internet-based Access to Real Equipment at Computer Architecture Laboratories using the Java/CORBA Paradigm. Computers & Education, Vol. 36, No. 2, pp. 151-170, Elsevier Science. ISSN 0360-1315 (2001)
6. Flite Electronics Ltd. web page at http://www.flite.co.uk/index.html
7. Fuchiwaki, Y., Usuki, N., Arai, T., Murahara, Y.: *The DSP Experiments for Under Graduate Students.* ICASSP (IEEE) 6:3526-3529 (2000)