Enseñanza y Aprendizaje de Ingeniería de Computadores

Revista de Experiencias Docentes en Ingeniería de Computadores

Número 4, Mayo 2014

Edita: Departamento de Arquitectura y Tecnología de Computadores

Colabora: Vicerrectorado para la Garantía de la Calidad

ENSEÑANZA Y APRENDIZAJE DE INGENIERÍA DE COMPUTADORES
Revista de Experiencias Docentes en Ingeniería de Computadores

TEACHING AND LEARNING COMPUTER ENGINEERING
Journal of Educational Experiences on Computer Engineering

Número 4, Año 2014

Comité Editorial:

ISSN: 2173-8688 **Depósito Legal:** GR-899/2011
Edita: Departamento de Arquitectura y Tecnología de Computadores
Imprime: Copicentro Editorial

Printed in Spain Impresa en España

Índice

DESDE EL PUPITRE
(Experiencias de estudiantes)

Editorial

La aparición de este cuarto número de la revista Enseñanza y Aprendizaje de Ingeniería de Computadores coincide con el final del periodo lectivo del curso 2013/2014. Se completan así los cuatro cursos de la primera promoción en los nuevos grados de Ingeniería Informática y de Ingeniería de Tecnologías de Telecomunicación. Precisamente en el primer curso de estas nuevas titulaciones se inició esta iniciativa editorial con la que el Departamento de Arquitectura y Tecnología de Computadores quiere contribuir a la tarea de coordinación docente que se plantea con los nuevos planes de estudio de Bolonia. Este año tendremos los primeros graduados por la Universidad de Granada dentro de estos planes de Bolonia y, por tanto, se han impartido ya todas las nuevas asignaturas de estos dos grados en los que imparte docencia el Departamento. Los siguientes retos en el ámbito de la docencia provienen de los nuevos Másteres de carácter profesional para Informática y Telecomunicación, y el nuevo Máster de Investigación en Ciencia de Datos e Ingeniería de Computadores en el que se ha integrado el Máster de Ingeniería de Computadores y Redes.

Como en los anteriores números de la revista, los contenidos de este número están relacionados en gran medida con las Jornadas de Coordinación Docente y Empresas, JCDE (http://atccongresos.ugr.es/jcde/), que en su cuarta edición, se celebraron en Diciembre de 2013. Las JCDE de este año han tenido un carácter ciertamente especial, dada la relevante participación de estudiantes que hemos tenido a través de mesas redondas en las que se ha debatido, con detalle y con un espíritu constructivo muy de agradecer en los estudiantes, las principales cuestiones relacionadas con la docencia de asignaturas de la especialidad de Ingeniería de Computadores, y las demás asignaturas de grado que les preceden impartidas por el Departamento. Consecuencia del estado de opinión que se ha suscitado, y del compromiso de muchos de los estudiantes de la especialidad en la calidad de la docencia que se imparte, también son los artículos elaborados por estudiantes que se publican en este número, que suponen el inicio de una sección que hemos denominado “Desde el pupitre”, y que abrimos con el propósito de que la revista refleje también la perspectiva de los estudiantes en la formación Universitaria que reciben.

Junto con esta nueva sección, en este número también estrenamos un Comité Editorial más amplio, en el que se han integrado profesores de distintas Universidades españolas que imparten docencia en asignaturas relacionadas con la Ingeniería de Computadores. Este Comité no está ni mucho menos cerrado, e intentaremos incorporar a profesores de nuevas Universidades tanto españolas como extranjeras. De hecho, en este número no solo se incluyen artículos de profesores de otras Universidades españolas, sino que

también tenemos tres contribuciones de profesores de Universidades hispanoamericanas. Muchas a gracias a todos ellos por su contribución a la revista.

En cuanto a los artículos, en este número se publican trece, de los cuales tres corresponden a contribuciones de estudiantes donde ponen de manifiesto su visión tanto de asignaturas ("Ingeniería de Servidores desde la perspectiva de un estudiante" de J. Chamorro, y "Fundamentos de Informática: exposición y utilidad desde la perspectiva de un alumno" de M. Hoyo, ambos de la ETSIIT de Granada), como de la realización de un Proyecto de Fin de Carrera ("Concepción de un sistema de iluminación inteligente en Smart Cities como Proyecto Fin de Carrera de Ingeniería Informática" de V. Martín, M. Damas, J. González y H. Pomares, de la ETSIIT de Granada).

Respecto a los diez artículos restantes, se puede distinguir entre los que proponen herramientas hardware o software para la docencia, y los que realizan propuestas docentes para impartir distintas asignaturas. En el primer grupo tenemos otros tres trabajos, "Teaching 'Embedded Operating Systems' using Raspberry Pi and Virtual Machines" de D. Llanos (Universidad de Valladolid) que ilustra el uso de máquinas virtuales y la Raspberry Pi en la docencia de sistemas operativos empotrados; "HLStool: Una herramienta de Síntesis de Alto Nivel para el aprendizaje del estudiante en asignaturas de ATC" de A. del Barrio y G. Botella (Universidad Complutense de Madrid) donde se propone una herramienta de síntesis para la docencia de asignaturas de arquitectura de computadores; y "Avances en el desarrollo de un Repositorio para Recursos Educativos Abiertos" de S. V. Torres, M. S. Zangla y M. C. Chiarani (Universidad Nacional de San Luis, Argentina), que describe el estado de un proyecto para generar un repositorio de herramientas docentes de software libre.

En cuanto a las propuestas docentes, en "16 Cards to Get Into Computer Organization" de S. Tabik y L. F. Romero (Universidad de Málaga) se propone utilizar un juego de cartas para conseguir la participación activa de los estudiantes en la docencia de Estructura de Computadores, asignatura que también constituye el objetivo de la "Propuesta docente para la asignatura Organización de Computadoras - Lic. en Ciencias de la Computación" de C. Zanellato y J. Balladini (Universidad Nacional de Comahue, Argentina). La motivación al estudio de una asignatura, en este caso el estudio de Arquitectura de Computadores a través de referencias a su evolución histórica, también se considera en "Relatos, Mapas Conceptuales y Arquitectura de Computadores" de J. Ortega y M. Anguita (Universidad de Granada). El uso de tecnologías multimedia para el aprendizaje basado en problemas se aborda en el trabajo de J.M. Orduña, R. Sebastián, y R. Olanda de la Universidad de Valencia ("Introducción de metodologías de Aprendizaje Basado en Problemas mediante tecnologías multimedia"), y el aprendizaje basado en proyectos en "Arquitectura Orientada a Servicios, un enfoque basado en proyectos" de M. Castro-León, F. Boixader, D. Rexachs y Emilio Luque (Escola Universitària d'Informàtica Tomàs Cerdà y Universidad Autónoma de Barcelona). La docencia de una asignatura de servidores en el perfil de Tecnologías de la Información se describe en "Propuesta de una asignatura de Diseño de Servidores para la especialidad de Tecnologías de Información" de A. J. Rivera, M. Espinilla, A.

Fernández, J. Santamaría, F. Charte (Universidad de Jaén). Y para terminar esta lista de contribuciones en este número de 2014, en "Una experiencia en la enseñanza de la programación para la permanencia de los alumnos de Ingeniería Electrónica", N. Arellano, M. V. Rosas, M. E. Zuñiga, J. Fernández , y R. Guerrero (Universidad Nacional de San Luis, Argentina) analizan las cuestiones relacionadas la docencia de una asignatura de programación en Ingeniería Electrónica.

Por tanto, los artículos que se recogen en este cuarto número de "Enseñanza y Aprendizaje de Ingeniería de Computadores", tocan aspectos de la docencia de asignaturas como Arquitectura de Computadores, Arquitecturas Orientadas a Servicios, Estructura de Computadores, Fundamentos de Informática, Ingeniería de Servidores, Programación para Ingenieros Electrónicos, Sistemas Operativos Empotrados, y Proyectos Fin de Carrera.

Como en los números anteriores queremos terminar esta presentación/editorial con nuestro agradecimiento a todos los que han contribuido a la celebración de las IV JDCE en Diciembre de 2013. En primer lugar, agradecemos a Joaquín Fernández Valdivia, director de la E.T.S.I.I.T. por su participación en el acto de inauguración de la JDCE y al equipo de dirección de la Escuela por facilitarnos el uso de las instalaciones; a los ponentes en las mesas redondas y a los representantes de la empresa Fujitsu por sus presentaciones y su participación en la mesa redonda posterior. A todos ellos, muchas gracias por su participación y su trabajo. También agradecemos al Departamento de Arquitectura y Tecnología de Computadores, y especialmente al Vicerrectorado de Garantía de la Calidad de la Universidad de Granada el apoyo y la financiación recibida.

El Comité Editorial

16 Cards to Get Into Computer Organization

Siham Tabik and Luis F. Romero

Department of Computer Architecture
University of Málaga
29071 Málaga, Spain
E-mail: (felipe, stabik)@uma.es

Abstract. This paper presents a novel educative activity for teaching computer architecture fundamentals. This activity is actually a game that uses 16 cards and involves about twenty active participant students. Executing this activity in the first class of the course allows the student in only 45 minutes to acquire the fundamental concepts of computer organization. The results of the surveys that evaluate the proposed activity together with the grades obtained by the students at the end of course corroborate the importance of the proposed game in the assimilation of more complex concepts in computer architecture.

Keywords: Computer architecture basics, educative games, collaborative learning.

1 Introduction

The use of didactic games in higher education is a valuable educational process despite the reluctance of some professors who consider it a relaxation of the rigorous and solemn traditional methodology in university teaching. Indeed, Plato, the Greek philosopher and mathematician, considered the educational games an essential means for the formation of a perfect citizen and an excellent alternative to traditional methods, because they allow working with the skills of students by combining education, participation and fun.

According to Jean Piaget, the acquisition of new knowledge is built upon the previous knowledge and the learning process must be active in a way that the student receives from the professor only the tools to build knowledge but not the knowledge itself. These ideas acquire more relevance once extended to the collective learning in an interactive space [1, 2].

The combination of a structural and functional learning together with an analysis process in an educational game provides the essential requirements of a complete epistemic activity [3]. The pillars of the elaborated knowledge must be simple to guarantee that the student will focus on the essentials rather than having to worry about the details. Finally, the collaborative learning, in which students work together helping each other, guarantees the social dimension of the educational process [1–3]. These are the basics of the proposed game in which the students reproduce the functioning of the CPU.

On the other hand, the Computing Curricula 2001 [4] identifies 14 areas of basic knowledge in the discipline of Computer Science and each area specifies a set of units that comprises the core knowledge that a student should assimilate to obtain different types of degrees in Information Technology. The common requirements among several types of degrees is that the student must have a thorough knowledge of the fundamentals from the earliest computer courses.

The Instruction Set Architecture, i.e., the set of binary sequences that a computer is able to interpret as orders, has been traditionally identified as computer architecture. In fact, this instruction set is an abstraction of the considered hardware [6]. Therefore, understanding the concept of instruction and instruction cycle at ISA level is essential not only to understand the functioning of the computer but also to indirectly understand the foundations of the modern society.

There exist few works about active collaborative learning processes for novice CS students to help them understand the functioning of computers. For instance, Powers [8] proposed an active learning lab exercise, called "The living CPU", to help students to actively deduce the functioning of the CPU. The task of the teacher consists of providing the opportunity for this to occur. However, the content of this exercise is rather limited. Sherry [3] proposed an epistemic game based on a model processor to allow students to comprehend the role, function, structure and mechanism of a real computer. Nevertheless, the collaborative and collective aspect of this activity is limited. The activity proposed in this work not only covers the entire level of abstraction of the Instruction Set Architecture (ISA), which is of paramount importance for understanding the functioning of a processor, but also implies a high participation of both students and teacher.

This paper presents an educational activity in form of a role play in which the students represent the functioning of a computer without the need of previous knowledge. This game is specially appropriate for the first or second class of the subject of fundamentals of computers and takes about 45 minutes. The proposed activity consists of representing the functioning of a computer at ISA level via the execution of a code that occupies only 16 bytes in memory. The main contribution of this work is a careful selection of the content of the proposed code, where by means of only 14 instructions, 2 data in memory, and executing the mathematical factorial operation, it introduces all the essential concepts of the subject. The evaluation process showed very satisfactory results. In addition, this activity not only plays a crucial role to encourage student-teacher relationship, but also has shown to completely fulfill the drawn academic goals.

2 The 16 Card Game

The proposed activity is appropriate for the first or second class after giving a brief description of the history of computer science and after presenting a short biography of Von Neumann. Recall that the main objective of the proposed game is that the student comprehends the relevance of the architecture of Von Neumman. This section i) underlines the most important concepts for the un-

derstanding of Von Neumann architecture, ii) gives a description of the proposed 16 cards activity and its dynamics and, iii) provides the concepts introduced by each card.

2.1 Fundamental Concepts

The proposed activity aims i) to provide a suitable and complete introduction to the Instruction Set Architecture (ISA) and ii) to develop the knowledge of the student at the ISA level considering its duality hardware-software. We carefully selected a set of 15 specific concepts to substantially facilitate the learning process of the student [6, 7]. In particular, we identified:

- five concepts associated with the hardware aspect (Control Unit and DataPath (CU&DP), Input Output (I/O), General Purpose Registers (GPR), Arithmetic Logic Unit (ALU) and, STacK (STK)),
- five concepts associated with the software aspect (JuMP (JMP), CoNDitionals (CND), routines and functions (CLL), variables and Direct Memory Addressing (DAM) and, pointers (PTR)) and,
- five concepts associated with the functioning mode and the hardware/software interface (Instruction CYcle (ICY), Instruction SeQuencing (ISQ), Loads and Stores from memory (L/S), IMplicit Addressing (IMP) and , state of the computer (FLA)).

Introducing each one of these concepts is carried out using the every-day life analogies. For example, to understand the concept of pointers, the student finds in a box the number of the card that contains the data. As the game goes, all the concepts are revisited and the essential concepts are reinforced by adding details to ensure the progress of the learning process. Furthermore, during the course, as the concepts included in the agenda are appearing the professor will explicitly make reference to the activity proposed in this work. In this way, the spiral learning is going to be completed with an appropriate time spacing that favors the assimilation of the concepts in the long term memory [5].

Content of the 16 cards written in assembly language

- 0 LOAD R0,C
- 1 RESET R1
- 2 INCR R1
- 3 CALL A
- 4 MUL R1, R0
- 5 DECR R0
- 6 CMP R0,#0
- 7 BRNE 4
- 8 STORE D, R1
- 9 JUMP E
- A OUT [R1],R0
- B RET
- C –data– 4
- D –data– any
- E EXCH R0, R1
- F CALL A

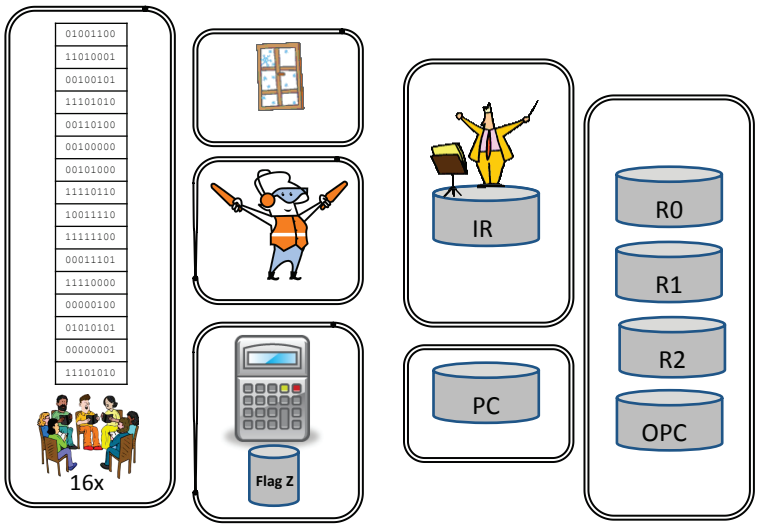

Fig. 1. The role of each player in the game.

2.2 The Cards

The professor prepares 16 cards printed in both sides. The first side, i.e., the side in green color, contains one of the instructions (or data) of the code shown in Figure 1. This side has a number that indicates the order of the instruction in the code written in base 10 system, i.e., from 0 to 15, and the instruction in plain English. In the back of each card, i.e., the side in orange color, there is a double translation of the content of the card in assembly and binary programming languages. The professor provides an additional dictionary material that provides the translation of all the used instructions from binary to assembly language, e.g., RESET=0010, C=1100, so that the student can understand that the content of the back of the card is in fact the translation of what the computer stores in memory and registers. Only the first side of the cards is used in this activity, and the additional information in the back will be revisited in next classes.

Fourteen cards contain instructions and two cards contain data of type integer, one of them is modified during the game to store the partial result of the factorial.

2.3 The Players

The professor selects a group of students, generally 22 students, and distributes the previously prepared 16 cards among them. The rest of the students are assigned a role each one. The professor plays the role of Control Unit that governs the computer and the students are assigned the following tasks:

- 16 students represent 16 memory positions. Each one is assigned one of the 16 cards. A random distribution of the cards will help later to understand the concept of Random Access Memory (RAM)
- 1 student represents the ALU unit. He is assigned a calculator and a box labeled Flag Z.
- 1 student represents the bank of registers (BR). He is assigned 4 boxes labeled R0, R1, R2 and OPC.
- 1 student represents the memory controller (MC).
- 1 student plays the role of the I/O unit. This role is usually assigned to a student close to a door or a window of the class.
- 1 student represents the program counter (PC).
- 1 student or the instructor plays the role of instruction register (IR)

Figure 1 sketches the role of each player.

2.4 Dynamics of the Game

The professor informs student PC that the game starts. This student tells his content which is always a number initially equals 0. Student MC searches for the student whose card contains this number and asks him to tell his content loudly. This content has two orders. The first order is a common sentence to all the memory cards, which increments PC. The second order is a simple instruction written in plain English. The professor asks PC to sum one to the value he memorizes and write down in the board the mnemonic of the second instruction of the card.

After fetching the first instruction, the professor indicates that the process is mechanical and normally it is performed in picoseconds. The professor also points out that in practice the used language is the binary and the increment of PC is not explicitly written in the machine instructions. Computer architectures have implicit sequencing.

After the search operation is completed, the professor executes the instruction by selecting the student that has been assigned the corresponding role and asking him for example: "MC, please, could you localize card C" then, "card C, could you please tell us your content", "BR, could you please store the value in register R0".

The game keeps executing all the instructions till PC reaches the value 16. In summary, 28 instructions have to be executed including 30 memory accesses, two writes in the ports (the first writes the value to which the factorial is calculated and the second writes the result of the operation). Figure 2 shows the flow of the instructions of the code.

As the game advances, the professor can make jokes, for example, about the cards executed out of order like card 4 and A. After the execution of card 3, the professor can tell the student with Card 4:"Are you ready? You are going to work more then anyone else". Normally from the second iteration of the loop, the professor stops again to indicate that the processor is substantially much faster and that he should do the same.

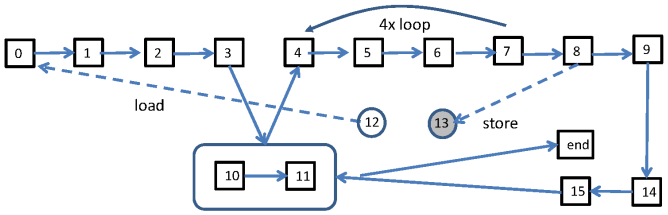

Fig. 2. Sequence of execution of the cards.

At the end of this activity, the professor stresses that the tasks performed by each participant are in fact very basic tasks, however, thanks to the high number of resources or transistors and the high rhythm of work, on the order of Gigahertz, we can perform relatively complex tasks, such as the calculation of the factorial of a number, in a small interval of time.

2.5 Relationship Concepts-Cards

The content of the cards and its relationship with the 15 fundamental concepts fixed previously is as follows:

- **Card 0** (LOAD R0, C): This instruction copies the content of Card C into Box R0. It is used to introduce two fundamental concepts in computer structure, the Central Processing Unit (CPU) and the buses that communicate different memories, in addition to three concepts from computer architecture, instruction cycle, instruction sequencing and memory cycle. It is recommended that the professor spends five minutes in this first card since it will serve simultaneously to introduce the pillars of the subject.
- **Card C** (–data– 4): This read-only data address has two objectives: i) to reinforce the memory cycle and differentiate it from the instruction cycle and ii) to explain the idea of the unique memory of instructions and data in Von Neumann architecture. The professor points out at this stage that a well designed program should find data and instructions where they are supposed to be.
- **Card 1** (RESET R1): Set R1 to 0. This card reinforces the concept of implicit sequencing through the program counter and the instruction cycle. It is also used to explain the concept of equal sign, e.g., A=B, in programming languages. That is, what the left hand side term stores after the execution of the instruction is what was calculated in the right hand side term during the instruction.
- **Card 2** (INCR R1): This card is used to introduce very different concepts although it is very similar to the previous one. The first concept to be pointed

out is that from the computer structure point of view, there is a big distinction between CU (represented by the professor) and the ALU. The second concept consists of introducing the idea of the code of operation, addressing by register and implicit addressing. In addition, the implicit increment of PC should be recalled at this stage too.

- **Card 3** (CALL 10): This instruction copies PC in OPC then sets PC to 10. This in probably one of the most important cards. It introduces the explicit sequencing and the imperative programming. The students with some previous basic knowledge in high level programming languages are able to identify the call to routine as low level operations. This card is also used to indicate the reason behind why the implicit increment of PC is performed at the beginning of the instruction cycle.
- **Card A**(OUT [R1], R0): This instruction consists of showing the content of box R0 at the port indicated by box R1. This card introduces three new concepts:
  - indirect addressing, i.e., pointers.
  - Input/output interface, which communicates the information to the exterior through one of the windows of the classroom.
  - General purpose registers
- **Card B** (RET): Return instruction indicates the end of the routine. It complements the description of the procedural programming. In addition, if considering a recursive call, this allows introducing the idea of substituting OPC by a Stack of registers.
- **Cards 4 and 5** (MULT R1, R0 and DECR R0): The first instruction multiplies box R0 by Box R1 and stores the result in box R1. The second instruction decrements the content of box R0. Student ALU is asked to execute these two instructions. At this stage the separation between control unit and data path is emphasized.
- **Card 6** (CMP R0, #0): Does box R0 has the value 0? If this is the case, store the value 1 in box Flag Z. This instruction introduces the concept of the flag and state register. The first execution of this sentence isolates form the next bifurcation to increase the student intrigue.
- **Card 7** (BRNE 4): Branch to 4 if flag Z is 0. This is also one of the most important instructions, it allows the student to understand that the program can change its execution in terms of the data.
- **Card 8** (STORE D, R0): This instruction stores in position D in memory the content of box R0. It introduces the operation of writing in memory and reinforces the concept of Von Neumann architecture.
- **Card D** (–data– any): The value of box R0, which should be 24, is written in this card. This is the only card that will be modified. This aspect is emphasized due to its relevance in the design of the memories. The concepts ROM and RAM are also introduced.
- **Card 9** (JUMP E): Using this card allows the student to distinguish between 4 sequencing modes in a computer, implicit, procedural, branch and explicit jump sequencing.

- **Card E** (EXCH R0, R1): This instruction switches the contents of R0 and R1. Together with the previous card, this card reinforces the concept ISQ. This instruction *per se* is not didactically relevant although it is used in class to talk for the first time about computers with complex instructions (CISC) versus computers with simple instructions (RISC).
- **Card F** (CALL 10): The execution of this instruction copies PC in OPC, then sets PC to 10. This instruction is used to reinforce the concepts of imperative programming. The second call to the same function allows the student to check that the use of functions reduces the size of the code.
- **Cards A and B** ($print(R0, R1)$ and RET): Already explained. The multiple execution of these function reinforces the concept of procedures and the use of pointers.

Table 1. The concepts attributed to each card. The symbols -, + and, ++ indicate an increasing intensity of emphasis on each concept.

Card	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ICY	++	++	+	++	-	-	-	-	-	-	-	-	+	+	-	-
ISQ	++	++		++	+			++		++	+	++			++	++
JMP				++				++		++		++			+	++
FLA							++	++								
CND							+	++								
CLL				++				+			+	+			+	++
STK				+								++				+
CU&DP	+	++	+		++	++			+							
ALU		+	++		++	++		+							++	
L/S	++								++		+		++	++		
GPR	+	+	++		-	-	-		+		++				-	
I/O											++					
PTR										+	++					
DAM				+					++	++	+		+	+		+
IMP		+	++		+	++	-					++				

Table 1 summarizes the concepts associated to each card indicating the intensity of emphasis on each one. As some cards are reused many times during the game, the intensity may vary slightly either in an ascending or descending way.

3 Evaluation of the activity

Although the practice of this activity started in 1997/98 with the students of the third course of industrial engineering, subject Computer Fundamentals, it was not evaluated till the academic year 2000/2001. Since then, 538 surveys were carried out with the objective of determining the incidence of the activity on the

Table 2. Level of knowledge of the concepts and degree of influence of the activity on the learning of each one of the concepts in the course 2011-2012

	Level of knowledge			Influence of the activity		
Concept	stage 1	stage 2	stage 3	stage 1	stage 2	stage 3
ICY	5.0	4.1	5.0	4.2	4.2	**4.3**
ISQ	4.8	4.0	4.6	4.3	4.3	**4.1**
JMP	4.8	3.3	4.9	4.5	4.5	**4.0**
FLA	3.5	2.6	4.4	5.0	5.0	2.2
CND	4.2	2.9	4.7	4.0	4.2	2.7
CLL	4.0	3.8	4.6	4.0	4.2	**3.4**
STK	3.8	2.1	4.3	4.4	4.6	1.9
CU&DP	4.6	3.5	4.9	4.9	4.9	**3.2**
ALU	4.8	4.3	4.9	4.3	4.0	2.3
L/S	3.7	2.5	4.8	4.6	4.7	1.5
GPR	4.4	4.3	4.9	4.8	4.9	1.0
I/O	4.9	4.7	4.9	5.0	5.0	**3.2**
PTR	3.2	1.5	4.1	4.8	4.8	0.8
DAM	3.3	2.0	4.7	5.0	5.0	1.1
IMP	4.6	3.8	4.8	5.0	4.9	**3.0**

learning process of the subject. The survey was partitioned into three stages. The first evaluation is performed right after the activity in the first class, the second evaluation is performed two weeks after the week of the activity and the third evaluation is done at the end of the course.

The survey asks the students two questions i) to evaluate their own level of knowledge in each one of the 15 concepts described in Section 2.1 using marks between 0 and 5 and also ii) to quantify the degree of influence of the activity in the assimilation of the concepts, an evaluation equals 0 means the activity did not have any influence in the knowledge of the concept while 5 means that the student learnt the concept only through the activity.

In general, the results, appropriately filtered[1], are very satisfactory since in the first stage the students think that they understood well the concepts, see Table 2. Moreover, two weeks later the results of the activity which has not been mentioned yet in class are still present in the memory of the students. At the end of the course, the students confirm that they knew 7 concepts (ICY, ISQ, JMP, CLL, CU&DP, I/O and, IMP) mainly through the activity presented in this work. For the rest of the concepts, the students think that about 20% of their knowledge is also thanks to the activity. The only concept that has not been affected by the activity proposed in this work is the pointers due to the fact that another specific activity were especially designed for this purpose.

The grades obtained by the students along the course could have been considered in the evaluation of the influence of the activity in the learning of the

[1] Discarding invalid and malicious responses

fundamental concepts. However, we should not consider it because the improvement of the average grade also depends on other factors such as the experience of the professor, the increase of the average mark for accessing the degree and, a better preparation of the students in the field of information and communication technologies.

On the other hand, for example in the course 2012-2013, the professor who implemented this activity got a very good evaluation by the students, 4.79/5 in 38 independent surveys carried out by *Centro Andaluz de Prospectiva* [10]. This evaluation exactly equals the mark that the student attributed to this specific question: "The professor uses didactic resources to facilitate the learning?". It is interesting to indicate that the average evaluation attributed by the students to the professors in the University of Málaga, is 4.70. This data indirectly provides positive additional evaluation about the relevance of this activity in teaching fundamentals of computers.

4 Conclusions

This work presented an activity that substantially easies the assimilation of the fundamental concepts in the functioning of a computer. The contents have been carefully elaborated to maintain simplicity. The results of several surveys demonstrated that the proposed activity is extremely useful to reach the drawn objectives.

References

1. Moreno L., Gonzalez C., Castilla E., Gonzalez E., Sigut J., (2007). Applying a constructivist and collaborative methodological approach in engineering education. Computers & Education, 49(3):891–915.
2. Ben-Ari M., (2001). Constructivism in Computer Science Education, Journal of Computers in Mathematics and Science Teaching, 20(1):45–73.
3. Sherry, L., (1995). A model computer simulation as an epistemic game, SIGCSE Bull, 27(2):59–64.
4. ACM, IEEE Computer Society (2001). Computing Curricula 2001. The Overview Report, Los Alamos, CA, The Join Task Computing Curricula.
5. Bruner J., (1960). The Process of Education, Massachusetts: Harvard University Press.
6. Hennesey J.L., Patterson D.A., (2012). Computer architecture, a quantitative approach, Massachusetts: Morgan Kaufmann.
7. Patterson D.A., Hennesey J.L., (2009). Computer Organization and Design: The Hardware/ Software Interface, Massachusetts: Morgan Kaufmann.
8. Powers K.D., (2004). Teaching Computer Architecture in Introductory Computing: Why? And How?, Conferences in Research and Practice in Information Technology, 30: 255-260.
9. Stanley T.D., Wang M., (2005). An emulated computer with assembler for teaching undergraduate computer architecture, Proceedings of the 2005 Workshop on Computer architecture education, 7.
10. Centro Andaluz de Prospectiva. http://canp.es/.

Propuesta de una asignatura de Diseño de Servidores para la especialidad de Tecnologías de Información

A. J. Rivera[1], M. Espinilla[1], A. Fernández[1], J. Santamaría[1], F. Charte[2]

[1]Departamento de Informática. Universidad de Jaén
{arivera, mestevez, ahilario, jslopez}@ujaen.es
[2]Dpto. De Ciencias de Computación e Inteligencia Artificial. Universidad de Granada
francisco@fcharte.com

Resumen. En este trabajo se presenta la asignatura de Diseño e Implantación de Servidores de la especialidad de Tecnologías de Información perteneciente a los estudios de Ingeniería en Informática de la Universidad de Jaén. El objetivo de esta asignatura es cubrir la formación que se establece en diferentes competencias relacionadas con en el desarrollo, despliegue e implantación de infraestructuras hardware necesarias para el soporte y funcionamiento de los sistemas de información actuales. Estos sistemas de información se caracterizan por presentar necesidades como acceso ubicuo, gran capacidad de cómputo o alta disponibilidad, entre otras. De esta manera la asignatura presenta un programa teórico/práctico donde se recorren desde principios básicos en el diseño de sistemas, su monitorización, referenciación o evaluación hasta conceptos de diseño actuales como alta disponibilidad, escalabilidad o balanceo de carga.

Palabras Clave: Diseño de servidores, monitorización, referenciación, evaluación, disponibilidad, escalabilidad, balanceo de carga.

Abstract. This paper presents the subject Design and Deployment of Servers belonging to the Information Technologies of the Computer Science Engineering at the University of Jaén. The objective of this subject is to provide the training established in the different competences related to the development and deployment of hardware infrastructures that supports the currently information systems. These information systems have characteristics such as ubiquitous access, high computational costs or high availability, among others. Thus, the subject addresses concepts from design of systems, monitoring, benchmarking or evaluation, to high availability, scalability or load balancing.

Keywords: Design of servers, monitoring, benchmarking, evaluation, high availability, scalability, load balancing.

1 Introducción

En los últimos estudios realizados por Gartner (http://www.gartner.com) o Forrester (http://www.forrester.com) sobre nuevas tendencias en el campo de las Tecnologías de la Información encontramos que estas se corresponden con tecnologías como la computación en la nube, la inteligencia de negocios, la computación de grandes cantidades de información (Big Data), la computación ubicua, etc. Todas estas disciplinas tienen como factor común la necesidad de una infraestructura para la computación de altas prestaciones o supercomputación.

Entre las prioridades que se establecieron para el Espacio Europeo de Educación Superior (EEES) de la década actual en la Conferencia de Ministros europeos responsables de educación superior [1] podemos encontrar la empleabilidad. En esta prioridad se enfatiza la orientación de las enseñanzas del mercado hacia la vida y el mercado laboral destino de los egresados.

En este sentido se han introducido diferentes competencias en las recomendaciones para el diseño de los grados en Ingeniería en Informática. Algunas de estas competencias se recogen y desarrollan en la asignatura de Diseño e Implantación de Servidores perteneciente a la especialidad de Tecnologías de la Información del grado en Ingeniería en Informática de la Universidad de Jaén.

El objetivo de este artículo es presentar dicha asignatura para lo cual en la sección 2 se expondrá la especialidad en la que se encuadra, temporalidad o créditos ECTS. En la sección 3 se describirá la asignatura en sí: sus competencias, contenidos, actividades formativas, programas teórico y práctico, y bibliografía.

2 El marco de la asignatura

Cómo se ha mencionado la asignatura de Diseño e Implantación de Servidores se enmarca dentro de la especialidad de Tecnologías de Información de la Ingeniería en Informática de la Universidad de Jaén. Esta ingeniería se diseña conforme a las directrices del Ministerio de Educación para la profesión de Ingeniero Técnico en Informática en la Resolución 12977 de la Secretaria General de Universidades publicada en el Boletín Oficial del Estado (BOE) de 4 de Agosto de 2009.

La especialidad de Tecnologías de Información consta de 48 créditos ECTS que el alumno debe cursar obligatoriamente. El módulo se compone de tres materias y 8 asignaturas, su correspondencia se muestra en las Tabla 1. Temporalmente se imparte entre tercer y cuarto curso. Esta planificación se muestra en la Tabla 2.

Tabla 1. Relación de materias y asignaturas que componen el módulo de Tecnologías de Información

Denominación de la materia	Asignatura	ECTS	Carácter de la materia
TECNOLOGÍAS WEB	SISTEMAS MULTIAGENTE	18	Obligatorio
	TECNOLOGÍAS BASADAS EN WEB		
	TECNOLOGÍAS PARA LA GESTIÓN DE LA INFORMACIÓN		
MULTIMEDIA	SISTEMAS MULTIMEDIA	12	Obligatorio
	PROCESAMIENTO DE INFORMACIÓN VISUAL		
INFRAESTRUCTURAS TECNOLÓGICAS	**DISEÑO E IMPLANTACIÓN DE SERVIDORES**	18	Obligatorio
	REDES E INFRAESTRUCTURAS DE COMUNICACIONES		
	DESARROLLO SOFTWARE PARA DISPOSITIVOS MÓVILES		

Tabla 2. Distribución temporal del módulo de Tecnologías de Información

Asignatura	Curso	Semestre
SISTEMAS MULTIMEDIA	3	2
TECNOLOGÍAS BASADAS EN WEB	3	2
DESARROLLO SOFTWARE PARA DISPOSITIVOS MÓVILES	3	2
SISTEMAS MULTIAGENTE	3	2
PROCESAMIENTO DE INFORMACIÓN VISUAL	3	2
DISEÑO E IMPLANTACIÓN DE SERVIDORES	4	1
REDES E INFRAESTRUCTURAS DE COMUNICACIONES	4	1
TECNOLOGÍAS PARA LA GESTIÓN DE LA INFORMACIÓN	4	1
OPTATIVA 1	4	-
OPTATIVA 2	4	-

3 Descripción de la asignatura

Tal y como aparece en la ficha VERIFICA, los datos básicos de la asignatura Diseño e Implantación de Servidores se resumen en que es una asignatura de la especialidad de Tecnologías de la Información en la Ingeniería en Informática de la Universidad de Jaén. Dispone de 6 créditos ECTS, se imparte en el primer semestre de cuarto curso y tiene un carácter obligatorio. A continuación, en la sección 3.1 se describen las capacidades contenidos descriptivos y actividades formativas de la asignatura, en la sección 3.2 los programas teóricos y prácticos, en la sección 3.3 la metodología de evaluación y en la sección 3.4 las recomendaciones bibliográficas.

3.1 Capacidades, contenidos y actividades formativas

Las capacidades de la especialidad que según la ficha VERIFICA el alumno deberá adquirir son:

- CTI2: Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados.
- CTI5: Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.

Los contenidos descriptivos que aparecen en la ficha para la asignatura son:

- Instalación, configuración, administración y mantenimiento de servidores.
- Monitorización y Evaluación de prestaciones.
- Conexión remota y Virtualización.
- Hardware para servidores: placas base, procesadores, almacenamiento e interconexión.
- Centros de Proceso de Datos.
- Seguridad y normativas.

Más pormenorizadamente las actividades formativas, con su contenido en créditos ECTS, su metodología de enseñanza-aprendizaje, y su relación con las competencias que debe adquirir el estudiante se describen en la Tabla 3.

Tabla 3: Actividades formativas, con su contenido en créditos ECTS, su metodología de enseñanza-aprendizaje, y su relación con las competencias

Actividades y Metodologías	Créditos ECTS	Horas presenciales	Horas trabajo autónomo del alumno	Competencia/s (códigos)
Clases expositivas en gran grupo: M1.1, M1.2, M1.4, M1.5	2,5	25	37,5	CTI2, CTI5
Clases en grupos de prácticas: M2.1, M2.2, M2.4, M2.5, M2.6, M2.7	3	30	45	CTI2, CTI5
Tutorías colectivas e individuales: M3.1, M3.4, M3.5, M3.6	0,5	5	7,5	CTI2, CTI5
Totales	6	60	90	

Concretamente las posibles metodologías a utilizar se enumeran en la Tabla 4.

Tabla 4: Posibles metodologías a utilizar

Actividades	Metodología	Código
Clases expositivas en gran grupo	Clases magistrales	M1.1
	Exposición de teoría y ejemplos generales	M1.2
	Actividades introductorias	M1.3
	Conferencias, etc	M1.4
	Otros	M1.5
Clases en grupos de prácticas	Actividades practicas	M2.1
	Seminarios	M2.2
	Debates	M2.3
	Laboratorios	M2.4
	Aulas de informática	M2.5
	Resolución de ejercicios	M2.6
	Presentaciones/exposiciones	M2.7
	Otros	M2.8
Tutorías colectivas/individuales	Supervisión de trabajos dirigidos	M3.1
	Seminarios	M3.2
	Debates	M3.3
	Aclaración de dudas	M3.4
	Comentarios de trabajos individuales	M3.5
	Presentaciones/exposiciones	M3.6

3.2 Programa teórico y práctico

Tanto los contenidos descriptivos como las competencias establecidas para la asignatura en la ficha VERIFICA, citados en el apartado anterior, intentan delinear una asignatura que recoja la formación necesaria que debe de tener un alumno que curse la especialidad de Tecnologías de Información en el ámbito del diseño de las infraestructuras de computo actuales.

Esta formación quedará totalmente delimitada con los siguientes programas de teoría y prácticas. Concretamente el temario de teoría que se establece para la asignatura es:

Tema 1. Introducción
- 1.1. ¿Qué es un servidor?
- 1.2. Tipos de servidores
- 1.3. Caracterización de servidores
- 1.4. Rendimiento y evaluación
- 1.5. Comparación de prestaciones

Tema 2: Internet. Administración y servicios
- 2.1. Introducción
- 2.2. Direccionamiento IP
- 2.3. Enrutamiento
- 2.4. DHCP
- 2.5. DNS
- 2.6. NAT

Tema 3: Monitorización y referenciación
- 3.1. Introducción
- 3.2. Tipos de monitores y sus parámetros
- 3.3. Herramientas de monitorización
- 3.4. Índices de referenciación
- 3.5. Software de referenciación

Tema 4: Servidores de altas prestaciones
- 4.1. Introducción
- 4.2. Alta disponibilidad
- 4.3. Escalabilidad
- 4.4. Balanceo de carga
- 4.5. Virtualización

Tema 5: Análisis operacional de servidores
- 5.1. Introducción
- 5.2. Redes de colas
- 5.3. Variables y leyes operacionales
- 5.4. Límites asintóticos del rendimiento
- 5.5. Algoritmos de resolución de modelos de redes de colas

Tema 6: Hardware de servidores y Centros de Proceso de Datos
- 6.1. Introducción
- 6.2. Hardware de servidores
- 6.3. Sistemas de almacenamiento
- 6.4. Formatos de servidores
- 6.5. Centros de proceso de Datos

El programa de prácticas diseñado intentará apuntalar los conceptos explicados en teoría y tendrá como hándicap una sincronización adecuada con esta. Concretamente se propone el siguiente programa de prácticas:

Práctica 1: Introducción a la administración de servidores (2 sesiones)
- 1.1. Herramientas para la administración servidores
- 1.2. Configuración básica de redes

Práctica 2: Configuración de redes y routers (1 sesión)
- 2.1. Configuración de redes
- 2.2. Tablas de enrutamiento

Práctica 3: Configuración de DHCP y DNS (1 sesión)
- 3.1 Configuración de un servidor dnsmasq

Práctica 4: Configuración de IPTables (1 sesión)
- 4.1. Cortafuegos
- 4.2. Enmascaramiento
- 4.3. Redireccionamiento

Práctica 5: Virtualización (3 sesiones)
- 5.1. Introducción a la virtualización
- 5.2. Virtualización de servidores

Práctica 6: Monitorización del sistema (1 sesión)
- 6.1. Herramientas básicas de monitorización del sistema
- 6.2. Monitorización con SAR

Práctica 7: Referenciación (1 sesión)
- 6.1. Protocolos y plataformas de referenciación

Práctica 8: Proyecto de prácticas (5 sesiones)

Las 7 primeras prácticas tienen un formato de desarrollo clásico donde hay un material teórico que es preparado y explicado por el profesor. Tras esto los alumnos pasan al desarrollo práctico conducidos por el correspondiente guion.

La última práctica se plantea como proyecto de la asignatura y consistirá en la instalación, configuración y puesta en marcha de un servidor de altas prestaciones. Concretamente al alumno se le propones varias tecnologías que caracterizan los servidores de altas prestaciones actuales como: alta disponibilidad, balanceo de carga, cluster de computación, … y se le pide que desarrolle e implante una. El formato de las sesiones cambia completamente, ya que ahora la idea es que el alumno no esté tan dirigido y que busque la documentación necesaria para llevar a cabo el proyecto.

Todas las prácticas, incluyendo el proyecto, se desarrollan sobre Linux, concretamente utilizando la distribución CentOS, la cual dispone de los paquetes software adecuados para el desarrollo de un servidor de altas prestaciones.

3.3 Metodología de evaluación

La metodología de evaluación es uno de los componentes más importantes que definen una asignatura ya que van a caracterizar fuertemente el proceso de enseñanza-aprendizaje de un alumno.

Así, el diseño de sistemas de evaluación adecuados va a incentivar el estudio habitual en el alumno y le va a servir para autoevaluarse en la compresión de los

contenidos teóricos o prácticos correspondientes. Para el profesor también van a ser un instrumento muy valioso ya que le permiten conocer el nivel de comprensión de los alumnos.

En este sentido, potenciando el trabajo individual y diario del alumno, se ha diseñado la metodología de evaluación de la asignatura, la cual se muestra en la Tabla 5.

Tabla 5. Metodología de evaluación

Aspecto	Criterios	Instrumento	Peso
Asistencia y Participación	-Participación activa en la clase. -Participación en el trabajo grupal	Observación y notas del profesor	10%
Conceptos teóricos de la materia	-Dominio de los conocimientos teóricos y operativos de la materia.	Examen teórico (prueba objetiva de respuesta breve y prueba objetiva tipo test)	35%
Conceptos prácticos de la materia	-Dominio de los conocimientos prácticos de la materia	Entrega de guiones de sesiones prácticas. Pruebas de tipo test periódicas	40%
Realización de trabajos o casos	-Entrega y exposición de trabajos investigación. En cada trabajo se analizará: -Estructura y calidad de la documentación -Originalidad	Trabajo por grupos	15%

3.4 Bibliografía

La bibliografía de una asignatura del ámbito de tecnologías de información cómo es la actual suele pivotar sobre una serie de recursos más o menos clásicos donde se recogen conceptos fundamentales relacionados y sobre otra serie de recursos más actuales, más técnicos y normalmente disponibles en la web.

Este es el caso de la asignatura de Diseño e Implantación de Servidores, donde se propone para el programa teórico una bibliografía compuesta de libros clásicos de diseño de sistemas o de evaluación y modelado de éstos [3,5,7]. También y para cubrir las últimas tendencias en estas materias se proponen obras cómo [2,4]

Para el programa de prácticas además de un best-seller en administración de sistemas en Linux como es [6] se propone la documentación electrónica disponible en el sitio web de Red Hat para la administración de sistemas. https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/

Referencias

1. Comunicado de la Conferencia de Ministros europeos responsables de educación superior, Lovaina / Louvain-la-Neuve, 28-29 de abril de 2009. http://viceees.ujaen.es/files_viceees/Comunicado-Lovaina-2009.pdf
2. Alger, D.: Build the best data center facility for your business. Cisco Press, (2012)
3. Jain, R.: The art of computer systems performance analysis: techniques for experimental design, measurement. John Wiley, (1991)
4. Josyula, V., Orr. M., Page, G.: Cloud computing : automating the virtualized data center. Cisco Pres, (2011)
5. Lilja, D.: Measuring computer performance: a practitioner's guide. Cambridge University Press, (2000)
6. Nemeth, E., Snyder, G., Hein, T., Whaley, B.: Unix and Linux system administration handbook. 4th ed. Upper Saddle River, Prentice Hall, (2011)
7. Molero, X.: Evaluación y modelado del rendimiento de los sistemas informáticos. Prentice Hall, (2004)

Teaching "Embedded Operating Systems" using Raspberry Pi and Virtual Machines

Diego R. Llanos
Dpto. de Informática
Universidad de Valladolid, Spain
diego@infor.uva.es

Abstract

Embedded systems design, implementation and programming is an important topic in many curricula of Computer Science and Computer Engineering. This paper describes the structure of a course on Embedded Operating Systems included in the Degree in Computer Science at the University of Valladolid. The course core elements are the use of virtual machines and Raspberry Pi kits. Our experience shows that the topics covered and the project-based methodology lead to excellent results in terms of student progression.

1 Introduction

Embedded systems design, implementation and programming is an important topic in many curricula of Computer Science and Computer Engineering. In the Degree in Computer Science offered by the University of Valladolid, study of embedded systems are divided in two optional courses. Both courses are credited with 6 ECTS credits (corresponding to sixty hours of lessons). The courses are Embedded Hardware and Embedded Operating Systems.

The former one, Embedded Hardware, covers the lower-level aspects of embedded devices, such as memory technologies, typical devices, buses architecture, transceivers, and electric specifications. This course is oriented to low-level design and programming, and it will not be covered in this paper.

In this paper we will focus in the last one, Embedded Operating Systems. This subject assumes the existence of an embedded hardware capable of running a tailored version of GNU/Linux. To do so, such a system should have at least a microprocessor with MMU management, 4 Mb of RAM and about 8 Mb of secondary storage, typically solid-state disks such as memory cards. Note that this was the standard configuration of a commodity PC in early 90s. However, miniaturization and cost reduction, together with the advent of low-powered Systems on Chip (SoC) have allowed the production of computers with these or better characteristics for less than forty euros. This is precisely the case of the Raspberry Pi microcomputer.

The Raspberry Pi [1] is a credit-card sized computer board composed of an ARM-like SoC and a board that includes all the I/O and storage interfaces

needed for practical use, including HDMI output, one or two USB ports, Ethernet port, slot for an SD memory card, and even a video output to connect the system to an analog TV. The Raspberry Pi was designed to teach Computer Science to teenagers, allowing an easy I/O access (including exposed GPIO pins) to facilitate the use of the system not only for programming, but also as the core of hardware projects of any size.

The existence of a computer such as the Raspberry Pi, capable of running GNU/Linux at a cost of some tens of euros, allows us to think that in the following years any embedded system will be able to do so, even systems with a cost of one or two euros. This fact would make the design of tailored embedded systems a comparatively expensive process, only useful when very special characteristics were needed. Besides this, the use of an operating system in embedded computing greatly simplifies the portability of embedded applications. These reasons justify the study of the development of tailored operating system for such devices.

We have found that the use of the Raspberry Pi is a good starting point to teach the difficulties of building an operating system for such devices, when memory and/or disk space constraints appear. Besides this, the ARM ISA of the Broadcomm processor integrated in the Raspberry Pi forces developers to use of cross-compiling tools to generate object code. Fortunately, as long as the Raspberry Pi is capable of running GNU/Linux, there exist many open-source tools available.

In the following sections we will first present the compromise solutions we have arrived to set up the needed hardware, and then we will describe the contents of the course.

2 Hardware and software requirements

Our course on Embedded Systems aims to teach the following concepts:

1. How to install and set-up a not-so-friendly GNU/Linux distribution in a personal computer (that we will call the *host* system).
2. How to compile the Linux kernel natively for the host system.
3. How to cross-compile the Linux kernel for a *target* system with a different ISA, in our case the Raspberry Pi.
4. How to compile the GNU operating system natively for the host system.
5. How to cross-compile the GNU operating system for the target system.

The development of specific applications for the target system is not a topic of this subject. The reason is that the knowledge about the cross-compilation process acquired in this course is the same that the one needed for application development. Besides this, the development of a C application for Linux is covered in depth in many other courses of our Grade in Computer Science.

To teach these concepts, two elements are needed:

- A personal computer for each student, that will act as the *host* system.
- An embedded-like system, to act as the *target* system.

2.1 Choice of a host system

The first choice we faced was to select an appropriate host system. Since the first objective listed in the previous section is to install a Linux on it, we needed complete (administrator-level) access to the host chosen. The use of the PCs that belong to the University laboratories for this purpose was quickly discarded because of several reasons. First, these machines are needed for other courses (and students) as well, so their configurations should not be modified by students. Second, to allow students to have root access to systems that are wired to the University network implies a security risk. Third, to assign a computer to each student would force the student to use the same physical terminal during the entire course, not allowing students to work after hours or at home.

To solve all these problems, we decided to use virtual machines, assigning one to each student at the beginning of the term. Virtual machines are processes running in a single server. Each process emulates the behavior of a PC architecture. The console of each virtual machine can be accessed remotely via a Java-enabled, web-based interface, that shows the console as a window, with buttons that allow to turn the virtual machine on and off, or to reset it, among other functionalities. The system administrator of the virtual environments can define the basic characteristics of each virtual machine, including processor type, amount of memory, hard disks types and sizes, number and types of network interfaces, and attach ISO images as DVD devices.

In our case, we created three dozens of virtual systems, one per student, configured as a single Pentium-compatible processor, 2 Gb of RAM, a 10 GB virtual hard disk, and a virtual DVD drive with an image of the GNU/Linux distribution chosen.

Besides granting unlimited access to their virtual machines, this solution has the additional advantage of being ubiquitous: Students can stop working at any time, closing the window that represents the console, and resuming their work later from a different location, just by accessing to the virtual server website with their credentials, and opening their console again. Note that to close the console window does not imply to turn off the virtual machine: When the student resumes his/her work, the console will show the exact contents the student may expect from a real machine that has been left out for several hours.

2.2 The target system

Regarding the target system, we have decided to use the Raspberry Pi type B microcomputer. Once chosen the architecture, we had two options: To ask students to buy their own, or to lend one to each student at the beginning of the semester. We have chosen the second option for a couple of reasons. First, to obtain a Raspberry Pi is not straightforward. The interested user should buy it online, and depending on the stocks available, his/her request may take several weeks to be served. Unless all students buy their systems well in advance, they may not have them ready for use at the beginning of the academic term. Second, to own a Raspberry Pi is not enough. Several additional components are needed, including:

- An USB keyboard.
- An USB mouse.

- An 1000 mA charger.
- An SD card.
- A video cable.

These items combined, together with the Raspberry Pi type B, cost around 80 euros. We wanted to avoid such a cost to our students whenever possible.

What we did is to buy enough Raspberry Pi kits as the one described above, plus a small, tablet-size bag to transport everything —except the keyboard[1]. The cost of the entire kit is around 100 euros. We have bought them with the support of the University of Valladolid. Regarding the video cable, as long as the Raspberry Pi comes with an HDMI output, we have chosen to buy HDMI-to-DVI cables, because the monitors of our laboratory have both VGA and DVI interfaces, and in this way we can use the monitor to work concurrently with the Raspberry Pi (selecting the DVI input) and with the laboratory PC that allows the student to use their virtual machines (using the VGA input). Moreover, right now it is more likely that the students have an DVI-capable monitor at home than a monitor that supports HDMI, although this situation will likely change in a near future. At that moment, to upgrade our kits with a pure HDMI cable will be straightforward.

3 Course structure and contents

The course has been divided into eight personal projects, designed to cover everything from the basic installation of a GNU/Linux system in a host machine to the cross-compilation of an entire GNU/Linux system from scratch. Two weeks (eight laboratory hours plus homework when needed) are devoted to each project, except Project #2, that only requires one week (four lab hours). We will briefly describe their contents in the following paragraphs.

Project #1: Installing a GNU/Linux system

The first task that our students face is to install a GNU/Linux distribution in the virtual machine that will be later used as host system, using an ancient, text-mode installation procedure. The time devoted to this first project (eight lab hours) may seem excessive, but two out of ten students fails when installing the operating system and should repeat the process, even several times.

Instead of using a mainstream distribution such as Ubuntu, we have chosen the Slackware GNU/Linux distribution. Slackware has a "package" classification that is easy to understand, allowing to install just the packages needed. Besides this, the distribution has a much simpler script structure, with very few dependences among packages. This is not the case with Ubuntu, which, for example, requires the installation of a web browser to run a barebone GIMP desktop manager (!). The advantages of using Slackware will become apparent in the following projects. One disadvantage is that Slackware is less "friendly", for example requiring the use of `fdisk` for disk partitioning. The project guidelines include all the intermediate steps that should be carried out.

[1] We discourage the use of "portable" keyboards: their keys distribution is awful!

We ask the students not to install the X interface, since all the projects should be completed through text-only interfaces. Only a subset of Slackware packages will be needed in this course, namely A, AP, D, F, K, L, N, and TCL.

Project #2: Installing a Raspberry Pi GNU/Linux distribution

This task is very simple. It consists in downloading an image of an Raspberry Pi GNU/Linux image from www.raspberrypi.org, transferring it to the Raspberry Pi SD card, and boot up and configure the operating system. As it can be seen, it is not related with the previous one, but it allows the students to face the use of the (potentially dangerous) dd command.

Project #3: Native compilation of a Linux kernel in the host

It consists in downloading a tarball with a recent Linux kernel source to the host system, uncompressing it in a working directory, configuring and compiling it natively, and installing it in our virtual machine. Since Slackware uses LiLo as the boot loader, the installation of the new kernel requires changes in the /etc/lilo.conf file. The use of LiLo is far better than using GRUB for the same purpose, for a couple of reasons. First GRUB uses a name convention for disk drives and partitions that is different from the classical names that can be found in /dev. Second, with LiLo it is always clear when we are performing physical changes in the disk drive and when we are not, a situation not so clear with GRUB.

After a first, successful installation, the students are challenged to tweak the kernel configuration (using make menuconfig) to arrive to the smallest possible kernels. Students receive six points out of ten for completing the installation of a fresh kernel. The four remaining points are proportionally assigned to the students depending on how successful they were reducing the space needed by the kernel and the associated kernel modules.

Project #4: Cross compilation of a Linux kernel for Raspberry Pi in the host

To compile the Linux kernel in a Raspberry Pi may take several hours to finish. To speed up the process, in this project we use cross-compilation tools. Therefore, the first task is to download and install the crosstool-NG cross-compiling toolchain[2].

Having a cross compiler, it may seem feasible to start with a standard, platform-independent kernel tarball and configure it to run on a Raspberry Pi. However, this is not the case. Kernel configuration for a specific platform requires an in-depth knowledge of both the kernel and the target system, and this task would require considerably more than the eight hours devoted to this project. Instead, we have chosen to instruct our students to download a certain kernel source code package that is known to work with Raspberry Pi[3]. With the help of a .config file obtained from a Raspberry Pi running kernel, that gives a

[2] Available at https://github.com/raspberrypi/tools.
[3] Available at https://github.com/raspberrypi/linux.

basic set of options, the students should make oldconfig to use this basic configuration as a starting point. Then they should cross-compile the kernel in the host system and transfer the kernel image, together with the corresponding modules tree, to the Raspberry Pi.

A final remark: Root access to the host system is not strictly necessary to cross-compile the Linux kernel. fakeroot can be used instead.

Project #5: Creating a *naïve* root filesystem (RFS) for the host system

It consists in the manually creation, in the host system, of the structure of a root filesystem (RFS). This RFS will be stored into a special partition created in Project #1 as part of the Slackware GNU/Linux installation. This project makes students to face the problem of how to create an entire RFS by re-using components already available in a Linux distribution. Let us remark that this task only involves the development of a RFS to be used natively by the host system: It has nothing to do with the Raspberry Pi (yet).

The students should manually create a RFS structure, using *mkdir* to create directories, chmod to change their permissions, and populating them with some user commands (namely bash, cat, date, elvis, grep, ldd, lddlibc4, loadkeys, login, ls, mkdir, mknod, more, mount, passwd, ping, pwd, rmdir, setterm, sh, sleep, ssh, stty, sync, umount, uname, vi, which, who, whoami) and some system commands (namely agetty, fsck, halt, ifconfig, init, insmod, klogd, kmod, ldconfig, lsmod, modprobe, mount, pwconv, pwunconv, reboot, rmmod, route, shutdown, sulogin, syslogd, telinit, umount.). This process implies not only copying the commands to their final locations into the RFS (preserving their ownership by using cp -a), but also to copy the dynamic libraries needed by them (using ldd to get the library names and manually copying them too). This allows the student to understand the role of dynamic libraries in a Linux system. The result is a RFS that will be converted in the following project as a bootable partition. Other options for creating a RFS, such as the use of files formatted as partitions and mounted in *loopback* filesystems, and the use of *ramdisks* for the same purpose, are discussed but not used.

The rationale behind this project (and the following one) is to make students to really understand why this method for building a distribution is not a good idea: It is a prone-error task, and the resulting RFS takes comparatively a lot of disk space.

Project #6: Adding support for standalone boot to our host-based RFS

This project consists in completing the transformation of our new RFS in a bootable one. This implies to copy a kernel image and its corresponding modules, add bootup scripts, adjust the runlevels that will be offered, install the bootloader, reboot and fix all the issues that may appear in the host.

Projects #5 and #6 helps the student to understand what is needed to successfully boot a small (in terms of storage footprint) version of GNU/Linux. However, to further reduce the disk space needed, several actions can be taken, from stripping executables to build an entire system from scratch, with tailored

flags for our particular system architecture. This topics will be covered in the next project.

Project #7: Space reduction: The strip command, BusyBox, and Buildroot

In this project the students will try different solutions to reduce the disk space needed for their RFS. The first one is to reduce the size of executables and library files using the strip command. The second one is to compile and install BusyBox, a single executable that is capable of acting as many different Unix commands, depending on the name used in its invocation. The last one is to compile a Buildroot distribution, an entire package that combines BusyBox with a small GlibC-like library called μlibC, and generates an entire RFS ready to be installed in the system.

Project #8: Cross-compiling a RFS for the Raspberry Pi

The last project is to use the accumulated knowledge of the previous projects to cross-compile and install an entire RFS from scratch for the Raspberry Pi system. The RFS will then be augmented with the cross-compiled kernel and modules generated in Project #2.

Scores

Finally, a note about the scores of this course. The final score has two parts, "practice" and "theory", with weights of 65% and 35% respectively. Projects #1, #2, #4 and #5 account for 10% of the "practice" part of the final score, while the remaining projects account for 15% each.

The remaining 35% of the final score (the "theory" part) is evenly divided into three tests, covering the projects developed, with the aim of testing the gain in terms of knowledge of each student. The first test covers projects #1 to #4; the second one, projects #5 and #6, and the third one projects #7 and #8. Since practice is the best way to learn, students usually pass these tests with little effort.

4 Conclusions

This course on Embedded Operating Systems has been taught during two consecutive years. Our experience allows us to draw the following observations.

- **The Raspberry Pi are tough.** We have had around 20 students in the first year, and 35 in the following year. The Raspberry Pi kits given to our students were returned in perfect conditions. This is somewhat surprising, since they are given to the students without a protecting case, just with the cheap plastic boxes that are part of the basic kit. The only exception was one that was received from the distributor with a condenser completely desoldered. The student that the Raspberry was assigned to discovered the situation, asked us about the polarity of the condenser, soldered himself and the Raspberry Pi booted with no further issues.

- **The course structure is scalable (to a certain extent).** The use of virtual machines allows the student to work at home. A side effect of this flexibility is that the number of students that effectively assist to the lab classes are about one third of the enrolled students, potentially allowing larger groups. This flexibility is perceived by the students as an advantage: The feedback received shows that students prefer to work in this way.
- **The Raspberry Pi is a versatile platform.** Besides being inexpensive, the Raspberry Pi allows not only to be used as a target platform for embedded operating systems, but as a valid testbed for any further development, from any software project to hardware I/O, thanks to their exposed GPIO ports. This fact adds value to the investment in Raspberry Pi kits as part of universities' equipment. This is particularly true for universities with tight budget restrictions.

Our future work includes to merge Projects #5 and #6, making room in the term schedule for a final project consisting in (a) building a simple hardware device with some leds and buttons, (b) attach it to the Raspberry Pi GPIO interface, and (c) write a small application to control it.

In summary, we are particularly happy with the results obtained so far with this experience. Any comment or question is welcome: Please contact the author for any further inquiry.

5 Acknowledgements

The author would like to thank the anonymous reviewers for their suggestions, and the Teaching Activities Support Program of the Universidad de Valladolid for its support to the purchase of the Raspberry Pi kits needed in this course.

References

[1] Raspberry Pi User Guide, Eben Upton and Gareth Halfacree, Wiley, 2nd edition, ISBN-13 978-1118795484.

HLStool: Una herramienta de Síntesis de Alto Nivel para el aprendizaje del estudiante en asignaturas de ATC

Alberto A. Del Barrio, Guillermo Botella

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática de la Universidad Complutense de Madrid
{abarriog, gbotella}@ucm.es

Resumen. La utilización de herramientas de Síntesis de Alto Nivel (SAN) es una práctica habitual en las empresas dedicadas al diseño de circuitos. El principal beneficio de estas herramientas se basa en la reducción del "tiempo de lanzamiento al mercado", ya que permiten evaluar múltiples soluciones en un tiempo reducido. En esta contribución, se propone el uso de una herramienta de síntesis altamente visual y amigable que contenga los algoritmos clásicos para enseñar al alumno las técnicas básicas de SAN, y adicionalmente familiarizarle con el método de trabajo de las compañías de diseño de circuitos. Con esta contribución se propone ofrecer una formación más completa a los futuros graduados que cursen asignaturas del área de la Arquitectura y Tecnología de Computadores

Palabras Clave: Síntesis de Alto Nivel (SAN), planificación, asignación, diseño de circuitos integrados, Grado en Informática, Plan de Estudios, EEES.

Abstract. The use of High-Level Synthesis (HLS) tools is a common practice in circuit design companies. The main benefit of these tools consists of diminishing time to market, as they provide multiple solutions to explore in a reduced amount of time. In this paper we propose the use of a highly visual synthesis tool, which implements the classic algorithms to teach students the basic HLS concepts and to get them closer to the companies' methodology. In this way, Computer Architecture and Technology related subjects will offer a more complete programme to the future graduates.

Keywords: High-Level Synthesis, scheduling, binding, Integrated Circuit Design, Computer Science Degree, Syllabus, EEES.

1 Introducción

Actualmente, la automatización del proceso de diseño (*aka*. Diseño Automático) se ha convertido en una tarea esencial, debido a la cada vez mayor complejidad de los diseños y al menor tiempo destinado a ellos por parte de las compañías. Sin embargo,

las titulaciones actuales en las ramas relacionadas con la Informática no ponen suficiente énfasis en esta área.

La Tabla I muestra un ejemplo de los contenidos y competencias que pueden encontrarse en 3 asignaturas del área de la Arquitectura y Tecnología de Computadores (ATC) en las titulaciones de Grado en Informática, Grado en Ingeniería de Computadores y Máster en Ingeniería Informática ofertados por la Universidad Complutense de Madrid (UCM) en el curso académico 2013/2014 [1]. La asignatura Tecnología y Organización de Computadores (TOC) se encuentra ubicada en el 2º curso de ambos grados, Diseño Automático de Sistemas (DAS) puede encontrarse como asignatura optativa en ambos grados también, mientras que la asignatura Sistemas Empotrados Distribuidos (SED) se ubica en el 1er curso del Máster de Ingeniería Informática. Tal y como puede observarse, en todas las asignaturas se enfoca la enseñanza de sistemas de acuerdo a un esquema algorítmico, es decir, el objetivo es aprender a diseñar *IP core*s [2] o subsecuentemente los *kernels* que los componen con mayor o menor granularidad. Dichos *kernels* normalmente consisten en funciones que requieren de una alta capacidad de cálculo, como los que ofrece la suite de *PolyBench* [3].

Sin embargo, el diseño que realizan los alumnos en las asignaturas previamente mencionadas es plenamente manual, mediante un lenguaje de descripción hardware de alto nivel como VHDL [4,5] o adicionalmente con el editor de esquemáticos de la herramienta Xilinx ISE [6]. Mediante esta filosofía de diseño manual, -usando VHDL- los estudiantes deciden el diseño que aparecerá en la implementación final. Desgraciadamente este diseño está sujeto a errores frecuentes. Sin embargo, si se usa el editor de esquemáticos, se subsanan muchos de los errores que aparecen en la metodología anterior, pero se sacrifica el control sobre el diseño final. Asimismo, como se ha mencionado previamente, hoy en día las compañías utilizan herramientas de Síntesis de Alto Nivel (SAN) [7] que permiten producir circuitos correctos por construcción y en un tiempo muy reducido, lo que les proporciona enormes beneficios [8].

En este artículo se pretende dar una alternativa para resolver los problemas mencionados por medio de la utilización de una herramienta de SAN muy visual y amigable, de tal forma que los alumnos puedan evaluar múltiples soluciones en un tiempo reducido, sin perder al mismo tiempo la perspectiva real de la implementación del circuito. De esta manera, los estudiantes aprenderán una metodología más cercana a la forma de trabajar en las empresas que diseñan circuitos. Además, la visualización de los resultados redundará en una mejor comprensión de los algoritmos de planificación y asignación utilizados.

El resto del artículo está organizado de la siguiente manera: la Sección 2 dará una visión global sobre el uso de las herramientas de SAN, mientras que la Sección 3 describe los detalles de la nuestra herramienta propuesta -HLStool-. La Sección 4 presentará un ejemplo ilustrativo para comprender su potencial aplicación y finalmente la Sección 5 expondrá nuestras conclusiones.

Tabla 1. Contenidos y Competencias adquiridas en 3 asignaturas ofertadas por la UCM en el curso 2013/2014 en los grados de Ingeniería de Computadores e Ingeniería Informática.

Asignatura	Contenidos	Competencias
Tecnología y Organización de Computadores	Circuitos Aritméticos, Sistemas Algorítmicos, Lenguajes de Descipción de HW	Conocimiento de la Estructura e Interconexión de los Sistemas Informáticos
Diseño Automático de Sistemas	Diseño Automático de Sistemas, Síntesis sobre FPGAs	Conocimiento de la Estructura y Arquitectura de un Sistema y Componentes que los conforman
Sistemas Empotrados Distribuidos	Componentes de Sistemas Empotrados	Capacidad para Diseñar Sistemas Empotrados y Ubicuos

2 Herramientas de Síntesis de Alto Nivel

Desde finales de los años 80 han aparecido múltiples compañías ofreciendo herramientas de SAN. Actualmente Synopsys con la herramienta *Synphony* [9], Cadence con *C-to-Silicon* [12], Calypto con *Catapult C* [11] y Xilinx con *Vivado* [10] son las compañías que dominan el mercado. Además de las soluciones comerciales, desde el sector académico se han desarrollado herramientas muy completas como *SPARK* [13], *xPilot* [14], *ROCCC* [15] o *LegUp* [16]. De hecho, *xPilot*, desarrollada en la Universidad de California en Los Ángeles (UCLA), es el núcleo de lo que hoy es *Vivado*. En conclusión, tal y como corroboran los datos de la figura 1, la SAN es un mercado que está en creciente auge y expansión. Por tanto, es importante proporcionar alumnado una formación básica en el contexto del Diseño Automático.

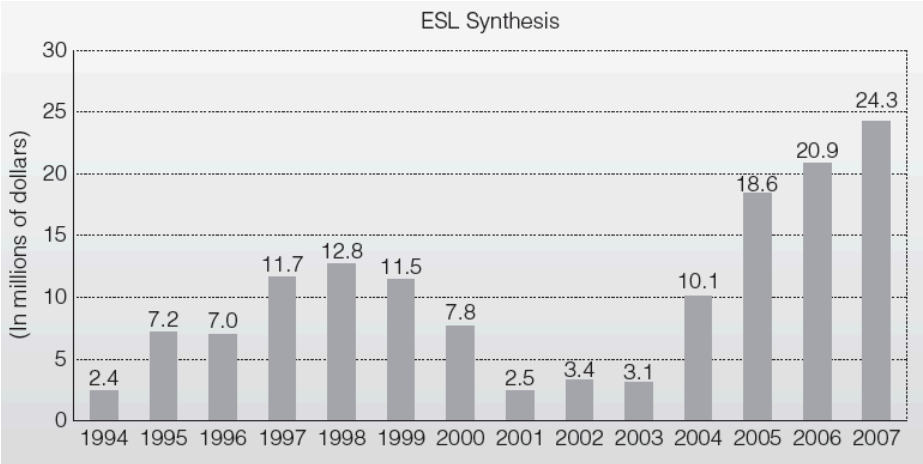

Figura 1. Venta de herramientas de Síntesis de Alto Nivel [17]

Quedando fuera de este artículo detalles más profundos de síntesis automática de alto nivel, podemos no obstante, abordar una breve descripción de la topología general de una herramienta SAN, que parte de un lenguaje de alto nivel, que desde comienzo de este siglo suele ser C [9-11] o variaciones como SystemC [12]. A continuación, transforma dicha descripción de alto nivel en un formato intermedio por medio de un *front-end* como el que ofrece LLVM [18]. En el caso de LLVM la representación sigue el formato AST (*Abstract Syntax Tree*). Posteriormente dicha información se procesa con algoritmos basados en grafos y se genera la descripción RTL que implementará el circuito.

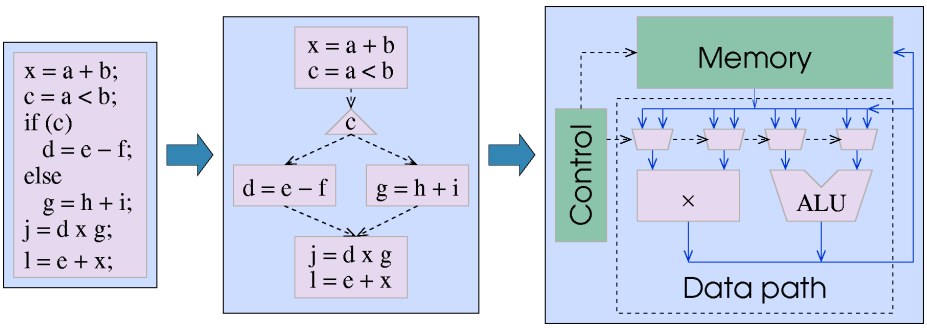

Figura 2. Flujo de Diseño en la Síntesis de Alto Nivel [11] desde C hasta RTL.

El flujo completo de diseño puede observarse en la Figura 2. Una vez obtenida la información de la descripción de entrada, las tareas realizadas por las herramientas de síntesis antes de generar la descripción RTL básicamente son 4:

1) Particionamiento (*partitioning*), o división del problema en subproblemas tratables.
2) Asignación de un número y tipo de recursos (*allocation*) suficientes para implementar el circuito.
3) Planificación (*scheduling*) de las operaciones en ciclos o control-steps (*csteps*).
4) Asignación (*binding*) de cada operación a instancias específicas de cada recurso.

Nuestra herramienta (HLStool) parte directamente de una representación en forma de grafo, y se centra fundamentalmente en las tareas 3 y 4. El objetivo es aportar una serie de interfaces gráficas que permitan al alumno comprender rápidamente que pequeñas modificaciones de las restricciones impuestas por el diseñador, pueden producir circuitos muy diferentes. Además, la visualización de los resultados permitirá comprender el funcionamiento de los distintos algoritmos de planificación y asignación utilizados.

3 HLStool

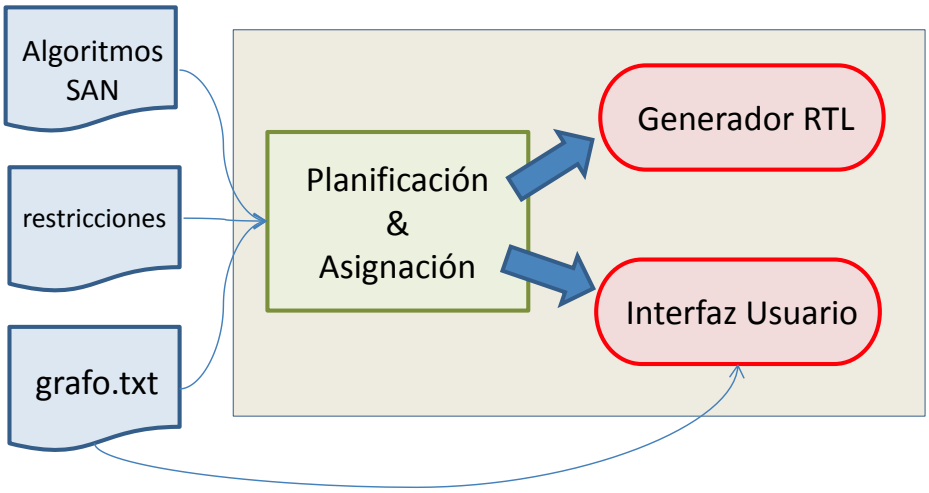

Figura 3. Estructura de la herramienta de síntesis propuesta (HLStool)

La estructura de HLStool puede observarse en la Figura 3. Inicialmente, se recibe un grafo a través de un fichero de texto. Dicho grafo puede mostrarse mediante un panel (*Dataflow Graph Panel*, o *DFG Panel*), aportando información acerca de las operaciones y sus relaciones. Además de ello, la herramienta recibe una configuración inicial que determinará qué algoritmos de planificación y asignación deben ejecutarse, y una serie de restricciones que especificarán detalles como la latencia del circuito o el número, tipo y latencia de los recursos utilizados (sumadores, multiplicadores, etc.). En otras palabras, la anteriormente mencionada tarea de *allocation* se realiza por medio de una restricción de usuario.

Nuestra herramienta implementa diversos algoritmos clásicos de planificación [7]:

1) As Soon As Possible (*ASAP*). Este algoritmo planifica las operaciones tan pronto como sea posible, es decir, en cuanto los operandos de entrada estén listos.
2) As Late As Possible (*ALAP*). Dada una latencia de circuito fijada por el usuario, el algoritmo planifica las operaciones tan tarde como sea posible.
3) Planificación basada en listas con prioridad. En cada *cstep* las operaciones disponibles se añaden a una lista de operaciones candidatas ordenadas por prioridad. De entre la lista de candidatas se seleccionarán las que se adecuen a las restricciones de usuario. Por ejemplo, si hay 4 sumas candidatas y tan solo 2 sumadores disponibles, se planificarán las 2 sumas de mayor prioridad.
4) Force Directed Algorithm (*FDS*) [19]. Dada una latencia de circuito, el algoritmo basado en fuerzas distribuye las operaciones de la forma más homogénea posible, para reducir el número de recursos necesarios para implementar el circuito.

Respecto a los algoritmos de asignación de operaciones, HLStool implementa las siguientes opciones [7]:

1) Algoritmos voraces para la asignación de operadores [7]. Dado un número de operadores de un tipo, y una planificación, cada operación se asigna al primer operador disponible.
2) Algoritmo *Left-Edge* o del Lado Izquierdo [7] para la asignación de registros. Dada una planificación, dicho algoritmo calcula el número mínimo de registros necesario para almacenar todas las variables, además de asignarlos.
3) Política *Least Recently Used* (LRU) para la asignación de registros. Para cada variable, esta política asigna el registro menos recientemente usado. En el contexto de la SAN clásica, tendría un efecto parecido al algoritmo *Left-Edge*, pero con la aparición de operadores de latencia variable [21,22] y la consecuente introducción de algoritmos de planificación dinámica para rutas de datos [20,21], puede tener un efecto apreciable sobre el rendimiento.

El resultado de ambas tareas se muestra por medio de un panel (*Operations Panel*), representando las operaciones con óvalos y los registros con rectángulos. Los colores determinarán a qué recurso en concreto está asignada cada operación. Un ejemplo de estas interfaces gráficas se describe en la siguiente sección. Además de la información visual, nuestra herramienta ofrece la funcionalidad de crear una descripción RTL generando sendas ruta de datos y controlador especificados en VHDL.

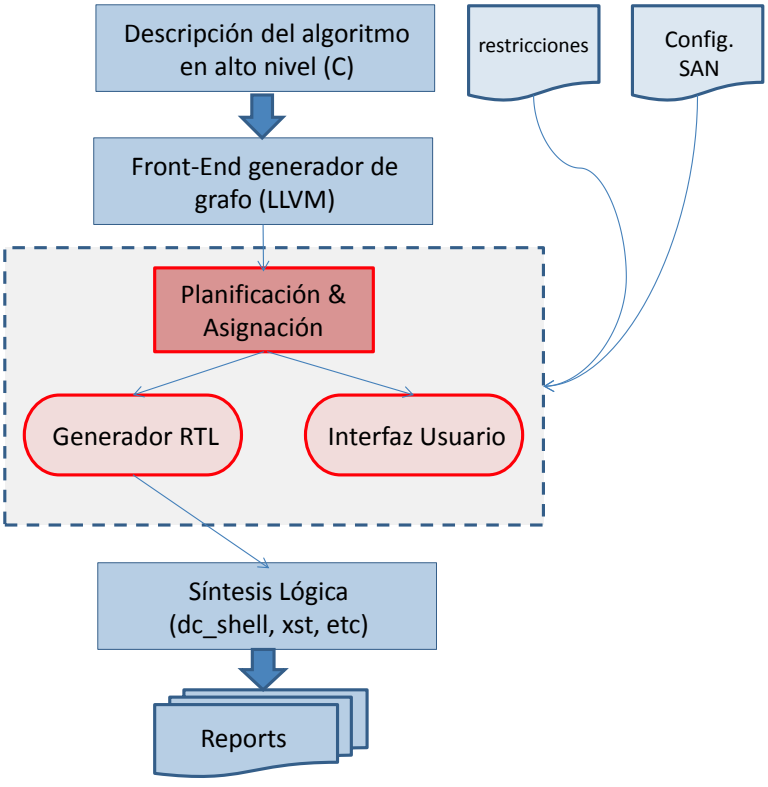

Figura 4. Flujo completo de diseño

Finalmente, la figura 4 muestra cómo se integraría HLStool en el flujo completo de diseño. Tan solo sería preciso proporcionar el grafo de entrada, que en nuestro caso es un fichero .txt. Para ello, bastaría con utilizar un frontend como *Clang*, proporcionado por LLVM [18]. *Clang* es capaz de construir un árbol en formato AST partiendo de un lenguaje de alto nivel como C. Por tanto, el AST nos proporcionaría el grafo. Por último, la descripción RTL generada por HLStool podría utilizarse para obtener datos de tiempo, área, consumo, etc. por medio de una herramienta de síntesis lógica.

4 Ejemplo Ilustrativo

En esta sección describiremos gráficamente los resultados correspondientes a utilizar la herramienta sobre el filtro Least Mean-Square (*LMS*) [23] aplicando el algoritmo de planificación basado en listas y el algoritmo Left-Edge para la asignación de registros. El código del LMS en C se encuentra en la figura 5 a). El panel del Dataflow Graph se encuentra en la Figura 5 b). Tal y como puede observarse, en este caso se trata de una instancia del código de la figura 5 a) en la que *length*=4.

a)

```
X_expected = 0;
for(i = 0; i < length; i++)
    X_expected += k[i]*x[i];
E = (X - X_expected)*Beta;
for(i = 0; i < length; i++)
    k[i] += E*k[i];
```

b)

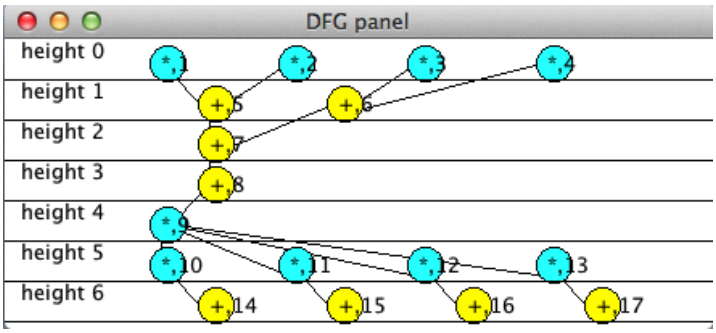

Figura 5. a) Código C del LMS filter, b) Dataflow Graph Panel para el filtro LMS [23]

Como puede observarse, cada operación aparece identificada por un *id* numérico, y el tipo de la misma viene determinado por un color: azul para las multiplicaciones y amarillo para las sumas. En la Figura 6 se muestra el resultado de la síntesis

utilizando como restricciones de entrada 4 multiplicadores de 4 ciclos de latencia, y 4 sumadores de 1 ciclo de latencia.

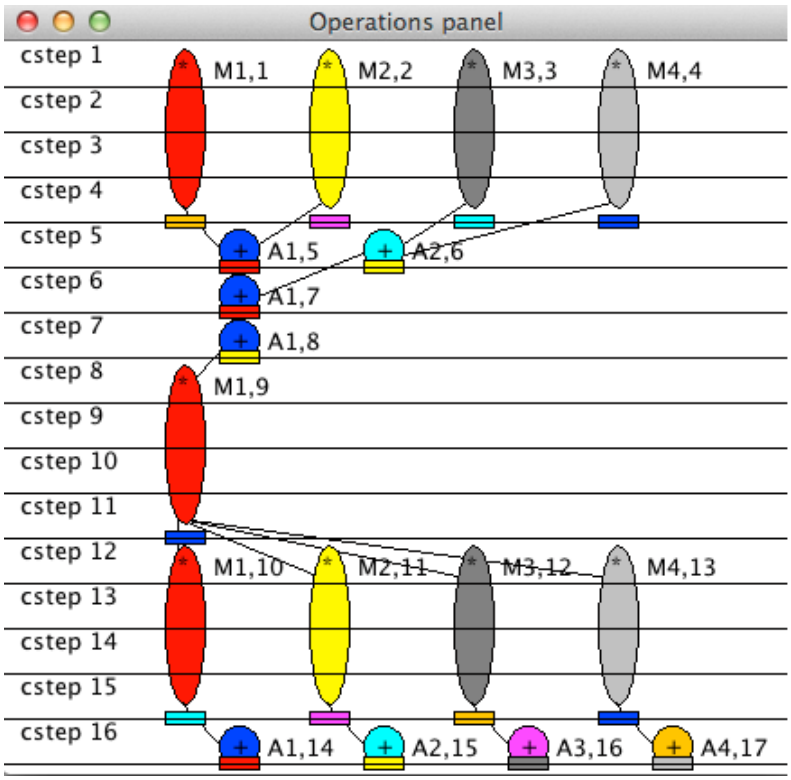

Figura 6. Panel de operaciones para el LMS [23], utilizando 4 multiplicadores de 4 ciclos de latencia y 4 sumadores de 1 ciclo de latencia.

El resultado producido por la herramienta sería una planificación de 16 ciclos de latencia y con la asignación mostrada por los colores de los recursos. Además, en el caso de los operadores dicha información viene complementada por una etiqueta identificativa del operador en el cual está asignada la operación. Por ejemplo, en el multiplicador M1, de color rojo, se han asignado las operaciones 1, 9 y 10. Si ahora se cambiasen las restricciones de entrada, el estudiante podría ver fácilmente cómo varía el circuito sin necesidad de generar la especificación RTL.

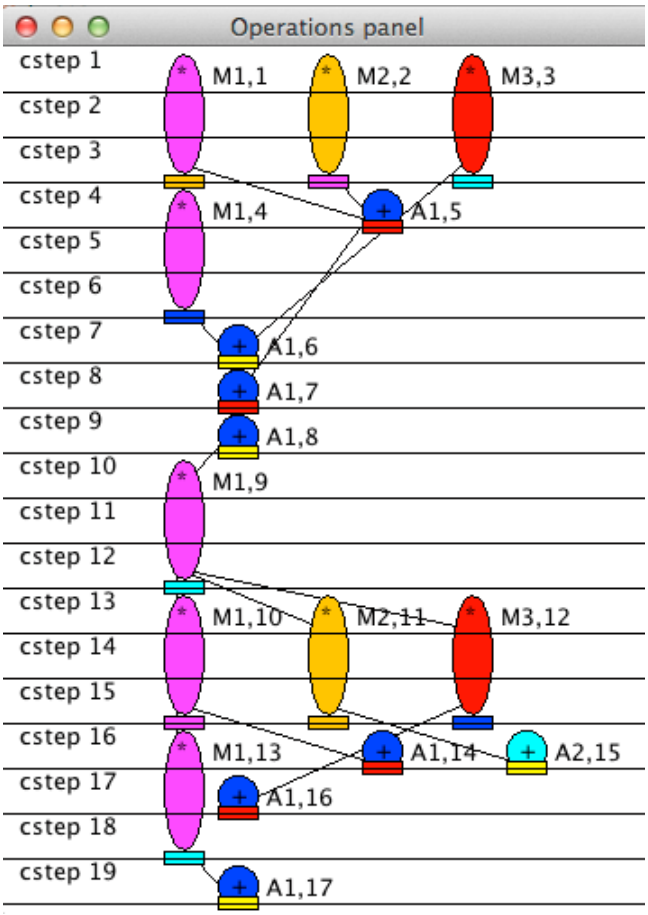

Figura 7. Panel de operaciones para el LMS [23], utilizando 3 multiplicadores de 3 ciclos de latencia y 2 sumadores de 1 ciclo de latencia.

La Figura 7 muestra el panel de operaciones tras cambiar las restricciones de entrada y utilizar 3 multiplicadores de 3 ciclos de latencia y 2 sumadores de 1 ciclo de latencia. En este caso, la latencia del circuito aumenta a 19 ciclos. Si quisiéramos resultados de área o potencia, generaremos la descripción RTL y utilizaremos una herramienta de síntesis lógica, como pueda ser *Design Compiler* de Synopsys [9] o *XST* de Xilinx [6].

5 Conclusiones

En este artículo se han descrito los potenciales beneficios del uso de una herramienta de Síntesis de Alto Nivel denominada HLStool, en asignaturas del área de Arquitectura y Tecnología de Computadores. En concreto, con la herramienta

propuesta, es muy sencillo evaluar el impacto que tendrían los cambios en las restricciones de entrada impuestas por el diseñador, por medio de una serie de interfaces gráficas que representan las operaciones, operadores y registros que aparecen en el circuito implementado. Todo esto sin la necesidad de realizar la síntesis lógica del circuito. Adicionalmente, el módulo de generación RTL permite obtener resultados de síntesis con cualquier herramienta comercial, lo que complementa la información de alto nivel proporcionada por la interfaz gráfica.

El uso de esta herramienta está pensada con un triple objetivo docente, por un lado formará de manera más completa a los estudiantes, por otro lado ayudará a que se familiaricen, en un tiempo reducido, a cómo trabajan las empresas relacionadas con el diseño de circuitos. Y adicionalmente, los alumnos aprenderán el funcionamiento de los algoritmos básicos usados en Síntesis de Alto Nivel.

Referencias

1. Facultad de Informática de la Universidad Complutense de Madrid: información docente, http://informatica.ucm.es/informatica/informacion-docente
2. Xilinx Inc.: IP cores, http://www.xilinx.com/products/intellectual-property/
3. PolyBench, http://www.cse.ohio-state.edu/~pouchet/software/polybench/
4. VHDL, http://www.vhdl.org/
5. N. Zainalabedin, VHDL:Modular Design and Synthesis of Cores and Systems, McGraw-Hill, 3ª ed., 2007.
6. Xilinx Inc.: ISE Design Suite, http://www.xilinx.com/products/design-tools/ise-design-suite/
7. Giovanni De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1ª ed., 1994.
8. D. McMillen, R. Camposano, D. Hill y T.W. Williams, An industrial view of Electronic Design Automation, IEEE Trans. On Computer-Aided Design, 2000, pp. 1428-1448.
9. Synphony, http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/default.aspx
10. Vivado, http://www.xilinx.com/products/design-tools/vivado/
11. Catapult C, http://calypto.com/en/products/catapult/overview/
12. C-to-Silicon, http://www.cadence.com/products/sd/silicon_compiler/pages/default.aspx
13. SPARK, http://mesl.ucsd.edu/spark/methodology.shtml
14. xPilot, http://cadlab.cs.ucla.edu/soc/
15. ROCCC, http://www.jacquardcomputing.com/roccc/
16. LegUp, http://legup.eecg.utoronto.ca/
17. G. Martin y G. Smith, High-Level Synthesis: Past, Present, and Future, IEEE Design & Test of Computers, 2009, pp. 18-25.
18. LLVM, http://llvm.org/
19. P.G. Paulin y J.P. Knight, Force-Directed Scheduling in Automatic Data Path Synthesis, Design Automation Conference, 1987, pp. 195-202.
20. A.A. Del Barrio, S.O. Memik, M.C. Molina., J.M. Mendías y R. Hermida, "A Distributed Controller for Managing Speculative Functional Units in High Level Synthesis", IEEE Trans. On Computed-Aided Design, 2011, pp. 350-363.
21. A.A. Del Barrio, R. Hermida, S.O. Memik, J.M. Mendías y M.C. Molina, "Multispeculative Addition Applied to Datapath Synthesis", IEEE Trans. On Computer-Aided Design, 2012, pp. 1817-1830.
22. A.A. Del Barrio, R. Hermida y S.O. Memik, "Exploring the Energy Efficiency of Multispeculative Adders", IEEE International Conference on Computer Design, 2013, pp. 309-315.

23. E. Musoll y J. Cortadella, “Scheduling and resource binding for low power”, International Symposium on System Synthesis, 1995, pp. 104-109.

Introducción de metodologías de Aprendizaje Basado en Problemas mediante tecnologías multimedia

Juan M. Orduña, Rafael Sebastian, and Ricardo Olanda

Departamento de Informática, Universidad de Valencia, Spain
{juan.orduna,rafael.sebastian,ricardo.olanda}@uv.es

Resumen. La propuesta de trabajar por competencias en lugar de hacerlo como se hacía tradicionalmente, por objetivos, ha hecho proliferar las propuestas y alternativas metodológicas para favorecer un cambio en la Educación Superior. Una de estas metodologías es el aprendizaje basado en problemas (ABP). El ABP es una metodología de aprendizaje en la cual el punto de partida es un problema construido por el profesor que permite al estudiante identificar necesidades para comprender mejor ese problema, identificar principios que sustentan el conocimiento y cumplir objetivos de aprendizaje relacionados con cada porción del contenido de la materia. En el caso de las tecnologías de la información y las comunicaciones es especialmente relevante, ya que permite desarrollar simultáneamente conocimientos teóricos y estrategias para resolver problemas prácticos en pequeños grupos, similares a los que se encuentran en la práctica profesional. La organización en grupos de trabajo permite además la creación de contenido de caracter longitudinal y transversal al grupo. El contenido transversal permite el uso de técnicas de aprendizaje colaborativo, que se pueden combinar con tecnologías multimedia en la web 2.0 para facilitar su difusión entre grupos y a la comunidad en general.
En este artículo se describe la experiencia docente en la asignatura de Arquitectura de Redes de Computadores impartida en los grados de Ingeniería Informática, Ingeniería Electrónica e Ingeniería Electrónica de Telecomunicaciones de la Universitat de Valencia. En ella, los alumnos de los tres grados han sido expuestos a la metodología de ABP y de aprendizaje colaborativo basado en tecnologías multimedia. Los contenidos de los grados se han adaptado adecuadamente de forma transversal permitiendo desarrollar los contenidos teórico prácticos. Los resultados de esta experiencia docente ha sido publicados en la web, así como los materiales generados por los grupos de alumnos. Todo el proceso ha sido supervisado por los profesores de la asignatura, obteniendo una retroalimentación continua acerca de su trabajo por medio de comentarios y evaluaciones realizados tanto por parte de los profesores como por alumnos externos al grupo de trabajo.

The introduction of skills-based learning instead of the more traditional goal-based learning has given rise to novel teaching proposals and methodological alterna tives that favor a change in the context of the Euro- pean

Higher Education Area. One of these methods is problem/project-based learning (PBL). PBL is a learning methodology in which the starting point is a problem constructed by the teacher, which allows students to identify needs to better understand the problem/situation, identify principles underpinning knowledge and meet learning objectives related to each content part. In the case of Information and Communications technologies it is especially relevant, since it allows the concurrent development of theoretical knowledge and strategies for solving problems in small groups, similar to those found in practice. This methodolgy can also be combined with the collaborative learning and the use of multimedia and web technologies for presenting the work done. It improves both the degree of motivation of the students and the degree of disseminations of the works. In this paper, we describe the teaching experience in the course of Computer Networks Architecture taught in undergraduate degrees of Computer Engineering, Telematics Engineering and Telecommunications Electronic Engineering from the University of Valencia. The students have been divided into working groups, and they have developed digital objects of learning that have been freely disseminated through Internet. This methodology has provided the students not only with collaborative learning within each group, but also with inter-group collaborative learning. Each group obtained feedback from the comments and evaluations of other students and the course teachers.

Keywords: Aprendizaje basado en problemas, aprendizaje colaborativo, tecnologías multimedia

Keywords: Problem-Based Learning, collaborative learning, multimedia technology

1 Motivación

Las metodologías docentes tradicionales basadas en la transmisión de conocimiento del docente al alumno no benefician el desarrollo de las competencias necesarias para el trabajo profesional de ingenieros en tecnologías de la información y las telecomunicaciones (TIC). Este tipo de trabajo suele desarrollarse a través de proyectos que presentan retos técnicos a equipos de ingenieros. Además, debido a la constante evolución de la tecnología, es necesario que los ingenieros TIC sean capaces de aprender de forma autónoma, y de aplicar conocimientos teóricos a la práctica.

Una técnica docente que se adapta a estos requisitos es el aprendizaje basado en problemas (ABP). Este método promueve en el alumno la responsabilidad de su propio aprendizaje, en lugar de que este sea un mero recipiente pasivo de información. Se fomenta la motivación del alumno para que aprenda de forma autónoma, y el profesor pasa de tener un rol de experto, a un rol de facilitador o tutor del aprendizaje. Con ello se consigue desarrollar habilidades para la evaluación crítica y la adquisición de nuevos conocimientos con un compromiso de aprendizaje de por vida.

El desarrollo de problemas de tamaño medio o grande facilita la posibilidad de llevarlos a cabo en grupos reducidos y a lo largo todo el curso. El ABP permite el planteamiento de problemas que puedan ser evaluados incrementalmente mediante documentos similares a los utilizados en proyectos reales (informes), sobre los que el profesor pueda ofrecer *feedback* que ayude a los alumnos a reflexionar y profundizar. En este marco, se consigue fomentar habilidades para las relaciones interpersonales y trabajo colaborativo. Por otra parte, el trabajo colaborativo se puede definir como actividades de aprendizaje expresamente diseñadas para grupos pequeños de alumnos que interactúan entre sí [2]. Las propuestas que han evaluado la efectividad del trabajo colaborativo muestran que los resultados son positivos para el proceso de aprendizaje [5, 10]. Son varias las técnicas existentes para aplicar esta metodología colaborativa. Algunos autores han hecho una comparativa de algunas de estas técnicas, como por ejemplo Hausmann [9], que se centra en 3 técnicas distintas: explicación directa, co -construcción y explicación por parte de otra persona. En base al estudio comparativo, se concluye que la explicación por parte de otra persona obtiene los mejores resultados en cuanto a cantidad de información aprendida, mientras que la co -construcción obtiene mejores resultados para la resolución de problemas.

En este artículo se describe la experiencia de unir la implantación y desarrollo de la metodología ABP con el aprendizaje colaborativo a través de la elaboración de una tarea longitudinal correspondiente a un proyecto, junto a tareas colaborativas transversales en forma de objetos digitales de aprendizaje. La división del trabajo en diferentes grupos se ha realizado siguiendo la técnica del puzzle de Aronson [1].

La metodología se ha aplicado a la asignatura de Arquitectura de Redes de Computadores, impartida en la titulaciones de Grado de Ingeniería Electronica de las Telecomunicaciones, Grado de Informática y Grado de Telemática de la Universidad de Valencia. ésta metodología ha permitido trabajar competencias específicas importantes tales como el trabajo en equipo o la comunicación oral. Los resultados obtenidos en las diferentes partes evaluables de la asignatura han demostrado una mayor implicación de los alumnos en los trabajos en grupo, si se compara con los resultados de carácter individual. Por otra parte, los objetos digitales de aprendizaje han permitido a los alumnos, además de adquirir los conocimientos propios derivados de su desarrollo, hacer uso de ellos como elementos constructivos básicos para la elaboración de unproyecto complejo que los englobe, sirviéndoles a su vez como recursos docentes disponibles en todo momento. Estos objetos estarán disponibles de forma libre para el resto de compañeros y para el público en general.

El resto de este artículo se divide en las siguientes secciones: en la sección 2 se revisa la metodologías del ABP y las diferentes técnicas de aprendizaje colaborativo. En la sección 3 de detalla el contexto y la realización de la experiencia. En la sección 4 se aportan los resultados extraídos de esta experiencia. Por último, en la sección 5 se presentan las conclusiones.

2 Estado del arte

2.1 Aprendizaje Basado en Problemas

El ABP es una metodología docente basada en el estudiante como protagonista de su propio aprendizaje. Es una estrategia educativa que hace descansar el proceso de aprendizaje del alumno sobre un problema construido por el profesor, similar a uno de la vida real, que sirve como vehículo, ya sea para la adquisición de un cuerpo integrado de conocimientos sobre la materia, o para la adquisición de competencias en la resolución de problemas prácticos [3, 13, 12]. En el ABP, los estudiantes se reunen en torno a un problema antes de recibir instrucción alguna sobre cómo resolverlo. El ABP representa toda una revolución respecto de los planes de estudios tradicionales donde los roles de profesor y alumno cambian. El papel del profesor en un aula de ABP es similar al de un guía o entrenador, es decir, el de facilitador, en lugar del poseedor de todo el conocimiento. Por ello, este enfoque requiere problemas o casos especialmente diseñados para motivar el aprendizaje de los aspectos más relevantes de la materia o disciplina de estudio. La colaboración y el aprendizaje cooperativo es esencial para el éxito del ABP [11]. El aprendizaje cooperativo de los estudiantes se realiza en pequeños grupos de estudiantes, y puede ser la parte más gratificante y productiva de su aprendizaje.

2.2 Aprendizaje Colaborativo Basado en Problemas

Con la aparición de las nuevas tecnologías se ha abierto un amplio abanico de técnicas para el diseño de tareas de trabajo colaborativo.

En el contexto de la docencia en Ingenierías existen también varios los trabajos que se han desarrollado para tratar el trabajo colaborativo. Uno de ellos se ha centrado en estudiar el efecto que tiene incorporar las redes sociales (en especial facebook) dentro del proceso de aprendizaje colaborativo [8]. Los resultados del trabajo muestran que se produce una mejora en la comunicación entre profesor y alumno. En otros trabajos se ha estudiado el aprendizaje colaborativo en Ingeniería Informática a través de herramientas Web 2.0, utilizando blogs, wikis y la plataforma virtual de la universidad Vaquerizo [14]. Los resultados muestran que la Web 2.0 es un medio de participación, comunicación y trabajo en equipo efectivo, pero el problema surge a la hora de evaluar a cada individuo del grupo por separado. Son varios los trabajos que mencionan YouTube como una herramienta Web 2.0 que se puede utilizar para aplicar el trabajo colaborativo [6, 7] de forma que permita la divulgación del contenido de aprendizaje desarrollado.

3 Aprendizaje Basado en Problemas apoyado en Objetos Digitales de Aprendizaje

Para aplicar la metodología de ABP en la asignatura de ARC, los alumnos organizados en grupos, deben llevar a cabo el diseño de una red de computadores que satisfaga las necesidades de comunicaciones de una empresa de gran tamaño.

En su desarrollo, se realiza un estudio detallado de cada una de las etapas de desarrollo e implantación de una red de computadores: definición del problema, diseño físico, diseño lógico e implantación de aplicaciones de red.

Para ayudar a los grupos en el desarrollo del proyecto, los propios alumnos elaborarán un conjunto de objetos digitales de aprendizaje transversales que profundizarán en cada uno de los puntos teóricos de diseño a tratar. Además el profesor les entregará un cuaderno de trabajo, donde se recogen los requisitos mínimos que debe incluir un proyecto, las diferentes fases del mismo, los roles que pueden desempeñar los alumnos, un calendario con las fechas límite para cada una de las fases del proyecto y los criterios de evaluación que se van a utilizar para valorar el mismo. La mecánica empleada para esta metodología es la siguiente:

- El profesor definirá las características del proyecto y de los objetos digitales de aprendizaje transversales a realizar, y proporcionará a los alumnos un cuaderno de trabajo para ayudarles a organizarse en el desarrollo del mismo.
- Para la constitución de los grupos se utilizará la técnica del puzzle de Aronson. El proceso de creación de grupos se detalla en la sección 3.1.
- Cada grupo de expertos elaborará el objeto de aprendizaje que le corresponde según cada fase del proyecto y lo envirá al profesor en formato de video presentaciones 3.2.
- Todos los videos elaborados se colgarán en un canal de YouTube público de la asignatura. Los videos estarán disponibles antes de que los alumnos comiencen la realización del proyecto de diseño.
- Mensualmente los alumnos, dentro de sus grupos nodriza, resolverán el proyecto de diseño para cada uno de los niveles (físico, lógico y aplicación), elaborando un documento escrito para cada uno de ellos.
- Cada grupo nodriza realizará una exposición oral de la solución propuesta para cada nivel del proyecto delante del profesor y de los otros grupos de compañeros.
- Cada grupo nodriza presentará su trabajo definitivo a través de un documento escrito y junto a un video describiendo el mismo o un poster, en el cual se resumirán las ideas fundamentales del proyecto.
- La evaluación de la resolución final del proyecto será evaluada por el profesor de la asignatura, por otros grupos de alumnos y por profesores externos a la asignatura, obteniendo una valoración que se complementará con la valoración del documento escrito del proyecto realizada por el profesor de la asignatura.

El desarrollo tanto de los objetos digitales de aprendizaje como del proyecto por parte de los alumnos debe realizarse de forma mixta, fuera del aula (la elaboración de los objetos de aprendizaje, el desarrollo del documento escrito y la preparación de las exposiciones orales del proyecto), para lo cual se emplearán las horas de trabajo fuera del aula especificadas en la guía docente, y en la propia aula (exposiciones orales y evaluación por compañeros), dentro de las horas de problemas contempladas en la planificación de la asignatura. De esta forma, la carga de trabajo del alumno se adecuará a la especificada en la guía docente.

3.1 Formación de los grupos

Como se ha comentado, se va a emplear la técnica del puzzle de Aronson para constituir los diferentes grupos que han de desarrollar los objetos digitales de aprendizaje y el proyecto de diseño de una red de computadores. El objetivo es que en cada grupo de desarrollo del proyecto de diseño exista al menos un experto en cada uno de los niveles de desarrollo del mismo. Los alumnos formarán grupos de 4 o 5 integrantes para la realización del proyecto de diseño de la red de computadores. Estos grupos se denominan grupos nodriza. Los alumnos han de desarrollar tres objetos digitales de aprendizaje asociados a los diferentes niveles de diseño de una red de computadores: físico, lógico y de aplicación. Para ello se crean diferentes grupos de expertos que están compuestos por uno o varios alumnos de cada uno de los grupos nodriza. Se ha de asegurar que cada grupo nodriza tenga al menos un miembro en cada uno de esos grupos de expertos. Es responsabilidad de los expertos, una vez hayan elaborado el objeto digital de aprendizaje correspondiente, explicar y compartir los conocimientos adquiridos dentro del entorno del grupo de expertos al resto de miembros del grupo nodriza, para que todos adquieran las competencias adecuadas en cada uno de los niveles de diseño.

3.2 Objeto digital de aprendizaje transversal

Los objetos digitales de aprendizaje que van a realizar los alumnos son videos multimedia. La base de estos videos multimedia está formada por transparencias con una plantilla predeterminada que son explicadas mediante la grabación de la voz en off de uno de los miembros del grupo. El vídeo se deberá centrar de forma teórico-práctica en la explicación de uno solo de los niveles de diseño de la red de computadores. Para su publicitación se usa una página web dentro del canal de videos YouTube púbblico. Adicionalmente a la realización de estos videos multimedia, los alumnos pueden elaborar de forma opcional un video multimedia que incluya la presentación final de todo su proyecto.

4 Resultados

La aplicación de la metodología de ABP en la asignatura de ARC se llevo a cabo a lo largo de cinco sesiones presenciales de dos horas en los espacios dedicados a seminarios/problemas. Adicionalmente, los estudiantes reportaron un total de entre seis a diez horas de dedicación por miembro del grupo para completar cada una de las fases del proyecto. Estas horas incluían tanto reuniones de todo el grupo (30%) como trabajo individual (70%). Todos los miembros de cada grupo fueron al menos una vez expertos de su grupo y realizaron una presentación oral al resto de la clase, lo cual ayudó a mejorar las competencias de comunicación oral y presentaciones frente a un público.

La figura 1 muestra, a modo de ejemplo, las puntuaciones globales obtenidas por los cinco proyectos realizados en el grado de Ingeniería Informática. Las

puntuaciones reflejan la calificación media de todas las sesiones destinadas a presentar las distintas partes del proyecto (especificaciones de usuario, diseño físico, diseño lógico, etc.) obtenidas por cada grupo de alumnos que realizaba un proyecto concreto. Para cada grupo hay 2 valores: la puntuación media otorgada por el profesor y la puntuación media otorgada por el resto de alumnos de la clase. La puntuación del resto de alumnos se realizó también de forma grupal; los miembros de cada grupo consensuaban una nota para el resto de grupos en cada sesión, en base a las presentaciones realizadas por estos y en base a las respuestas y discusiones surgidas después de cada presentación.

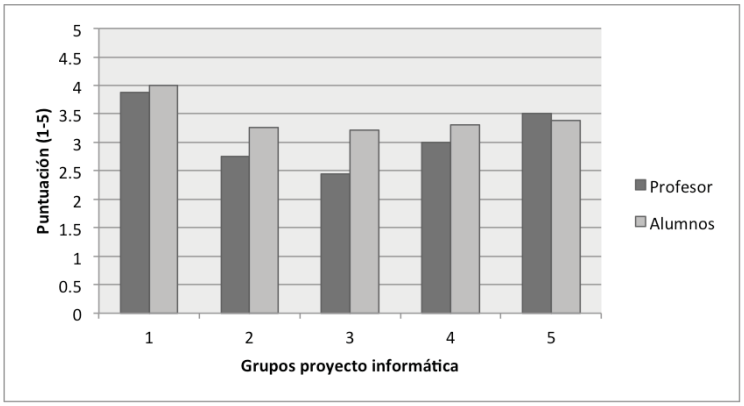

Fig. 1. Puntuaciones obtenidas en cinco proyectos en el grado de informática.

Los resultados de la figura 1 muestran que tanto para el grupo 1 como para el grupo 5 las puntuaciones otorgadas por el profesor y por el resto de los alumnos son muy parecidas, sin diferencias significativas. Sin embargo, para los grupos 2, 3,y 4 se observa que las puntuaciones otorgadas por el profesor son significativamente inferiores a las otorgadas por el resto de alumnos. Es de destacar que estas diferencias se deben en su gran mayoría a que los grupos presentaban trabajos donde algunos de los aspectos evaluados (p. ej. claridad/precisión/corrección de las respuestas) eran claramente deficientes, bien por ausencia o bien por incorrección de los contenidos. Si bien estas deficiencias quedaban explicitadas durante la discusión, los alumnos tendían a no reducir la nota de sus compañeros.

Por otro lado, para evaluar el uso de los objetos digitales de aprendizaje elaborados, se ha utilizado como métrica el número de visualizaciones que se han realizado de cada uno ellos. Estos vídeos se expusieron en una página de www.youtube.es/ARCuv2013. Además, también hemos analizado los resultados de la encuesta anónima acerca del uso de esta metodología rellenada por los alumnos.

En la figura 2 se muestra el número de visitas que ha recibido cada uno de los videos multimedia realizados por los alumnos. En esa figura se han agrupado por columnas los diferentes grupos de expertos que han realizado el mismo tipo de

video multimedia. El apartado de diseño del nivel físico (D.F.), se ha subdivido en tres, haciendo referencia al cableado de cobre, al de fibra y a los diferentes dispositivos de red.

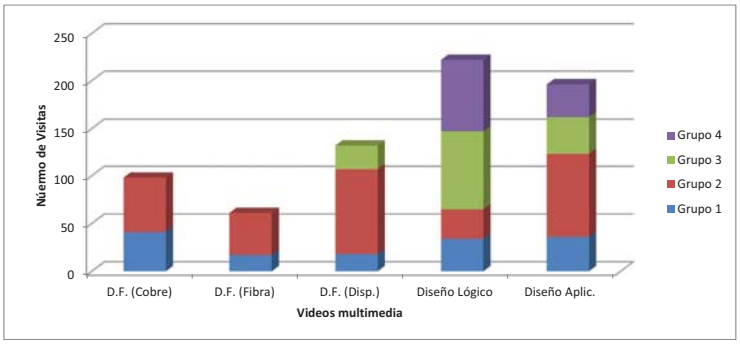

Fig. 2. Número de visitas recibido por cada uno de los videos multimedia

En la figura se observa que, para el nivel de diseño físico, a excepción de los videos multimedia referidos al cableado de fibra óptica (61 visitas en total para los dos videos), el número de visitas es superior al número de alumnos (que es 65), por lo que en media, podríamos decir que todos los alumnos han visto al menos una vez, un video multimedia de cada una de estas categorías.

El número global de visitas para el diseño físico sería de 291, lo que implica una media superior a 4 visitas por alumno a algún video de esta categoría. Para el diseño lógico el número total de visitas es de 222 y para el de aplicación de 196, lo cual supone una media de visitas por alumno superior a 3 en ambos casos. En cuanto a cada video particular, el número de visitas varía entre 17 y 89, lo que podría considerarse como un índice de calidad del mismo, ya que si le resulta útil al alumno, lo volverá a visualizar.

Finalmente, se realizó una encuesta anónima que constaba de las siguientes preguntas:

- He visualizado los videos de los diferentes niveles de desarrollo del proyecto.
- La visualización de estos videos para la realización del proyecto me ha resultado de gran utilidad.
- La realización del video me ha ayudado a comprender mejor ese nivel de desarrollo del proyecto.
- La técnica del puzzle de Aronson me ha parecido interesante.

Las respuestas posibles a estas preguntas son, para la primera pregunta, sí o no. Para el resto de preguntas, las respuestas abarcan una escala de Likert de 5 puntos, en función de lo de acuerdo que esté el alumno con la afirmación (1 = nada, 5 = totalmente de acuerdo).

Todos los alumnos respondieron afirmativamente a la primera pregunta, lo que supone que han hecho uso del material que se ha realizado. Los resultados

de las restantes preguntas (numeradas de 2 a 4) se muestran en la figura 3, donde se ha calculado la media de las respuestas de los alumnos (siguiendo las recomendaciones de trabajos previos sobre el análisis de las Likert [4]). En esa figura se puede observar que, en general, los alumnos ven útil la realización de estos videos. Cabe destacar que los alumnos indican que han aprendido más realizando el video (media en torno a 4) que viéndolo (media en torno a 3.7). La técnica del puzzle de Aronson, también la ven interesante, a pesar de obtener una valoración más baja (en torno al 3.3).

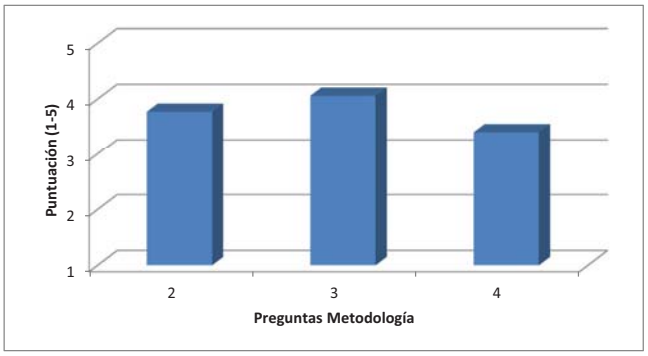

Fig. 3. Encuestas anónimas sobre la metodología docente

5 Conclusiones

En este trabajo hemos mostrado una experiencia de aplicación de aprendizaje basado en problemas unida a un aprendizaje colaborativo mediante la elaboración de objetos digitales de aprendizaje. Los alumnos han realizado diversos videos multimedia que se han convertido en objetos digitales de aprendizaje disponibles en un canal de YouTube. El acceso a estos videos es libre tanto para los alumnos de la asignatura como para el público en general. La aplicación de la metodología ABP ha demostrado ser beneficiosa para los alumnos, ya que ha permitido trabajar competencias específicas importantes tales como el trabajo en equipo o la comunicación oral. Los resultados obtenidos en las diferentes partes evaluables de la asignatura han demostrado una mayor implicación de los alumnos en los trabajos en grupo, si se compara con los resultados de carácter individual.

Por otra parte, las estadísticas de visitas reflejan que se ha hecho uso de los objetos digitales (vídeos) desarrollados, ayudando a los alumnos en la elaboración de su proyecto de diseño de una red de computadores. El uso de estos videos se hace patente a la hora de evaluar los proyectos, donde se observa una mejora de la calidad de los mismos respecto a cursos anteriores. Finalmente, los propios alumnos manifiestan que consideran útil esta metodología, ya que les

ha permitido comprender mejor los conceptos necesarios para el desarrollo del proyecto.

References

1. Aronson, E., Patnoe, S.: The jigsaw classroom: building cooperation in the classroom. Longman (1997), http://books.google.es/books?id=L0KfAAAAMAAJ
2. Barkley, E.F.: Técnicas de aprendizaje colaborativo: manual para el profesorado universitario. Ediciones Morata (2007)
3. Barrows, H., Tamblyn, R.: Problem-based Learning: An Approach to Medical Education. Springer Publishing Company (1980)
4. Blaikie, N.: Analyzing quantitative data: From description to explanation. Sage Publications (2003)
5. Dillenbourg, P., Baker, M., Blaye, A., O'Malley, C.: The evolution of research on collaborative learning. In: (Eds), E.S..P.R. (ed.) Learning in Humans and Machine: Towards an interdisciplinary learning science., pp. 189–211 (1995)
6. Esteve, F.: Bolonia y las TIC: de la docencia 1.0 al aprendizaje 2.0. La Cuestión Universitaria 5 (2009)
7. García, F., Portillo, J., Romo, J., Benito, M.: Nativos digitales y modelos de aprendizaje. In: Benito, M., Romo, J., Portillo, J. (eds.) SPDECE. CEUR Workshop Proceedings, vol. 318. CEUR-WS.org (2007)
8. Garrigós, I., Mazón, J.N., Saquete, E., Puchol, M.: La influencia de las redes sociales en el aprendizaje colaborativo. In: XVI Jornadas de Enseñanza Universitaria de la Informática. pp. 531–534. JENUI (2010)
9. Hausmann, R., Chi, M.T., Roy, M.: Learning from collaborative problem solving: An analysis of three hypothesized mechanisms. 26nd annual conference of the Cognitive Science society pp. 547–552 (2004)
10. Hennessy, S., Murphy, P.: The potential for collaborative problem solving in design and technology. International Journal of Technology and Design Education pp. 1–36 (1999)
11. Molina, J., Garcia, A., Pedraz, A., Anton, M.: Aprendizaje basado en problemas: una alternativa al metodo tradicional. Revista de la Red Estatal de Docencia Universitaria 3(2), 79–85 (2003)
12. Morales, P., Landa, V.: Aprendizaje basado en problemas. Theoria 13, 145–157 (2004)
13. Savery, J.: Overview of problem-based learning: Definitions and distinctions. The Interdisciplinay Journal of problem-Based Learning 1, 9–29 (2006)
14. Vaquerizo, B., Renedo, E., Valero, M.: Aprendizaje colaborativo en grupo: Herramientas web 2.0. In: XVI Jornadas de Enseñanza Universitaria de la Informática. pp. 447–450. JENUI (2009)

Propuesta docente para la asignatura Organización de Computadoras - Lic. en Ciencias de la Computación

Claudio Zanellato, Javier Balladini

Facultad de Informática, Universidad Nacional del Comahue.
Neuquén, Argentina
{claudio.zanellato, javier.balladini}@fi.uncoma.edu.ar

Resumen Este artículo presenta las experiencias en la asignatura "Organización De Computadoras" correspondiente al grado en Licenciatura en Ciencias de la Computación. La asignatura estudia principalmente la frontera software-hardware de las computadoras, su estructura básica, funcionamiento y programación en lenguaje ensamblador. El objetivo de este trabajo es mostrar la propuesta didáctica empleada, que intenta solucionar los inconvenientes presentados por la metodología previamente utilizada: ausencia de relación entre la programación de alto y bajo nivel que producía problemas al alumno para comprender otras asignaturas, y dificultad para detectar las causas de los errores que los alumnos cometían en los exámenes. La solución se basa en el diseño de actividades de enseñanza y aprendizaje que promueven la comparación de ambos niveles de programación, y una novedosa metodología de evaluación con exámenes que intentan recrear un ambiente de trabajo profesional con múltiple información a disposición. Los resultados indican una mejoría en la calidad del aprendizaje alcanzado por los alumnos.
Palabras Claves: Estructura de Computadoras, Organización de Computadoras, Aprendizaje constructivo, Metodología de evaluación.

Abstract This paper presents the experiences of teaching "Computer Organization" course for the Bachelor degree in Computer Science. The subject focuses mainly at software-hardware boundary of computers, their basic structure, internal operation, and assembly language programming. The objective of this work is to show the academic methodology, which attempts to solve the problems presented by the methodology used in previous course instances: absence of relationship between the high level programming and low level programming that cause problems to the students to understand other subjects, and difficulty in detecting the causes of the errors made by students in exams. The solution is based on the design of teaching and learning activities that promote comparing both programming levels, and a novel evaluation methodology through exams that attempt to recreate a professional work environment with multiple information available. The results show an improvement of the learning quality achieved by students.
Keywords: Structure of Computers, Computer Organization, Constructive Learning, Evaluation Methodology.

1. Introducción

La Facultad de Informática de la Universidad Nacional del Comahue está ubicada en la ciudad de Neuquén, Argentina. Una de las carreras de esta facultad es la Licenciatura en Ciencias de la Computación, que incluye en el cuarto semestre la asignatura denominada "Organización de computadoras" (ODC). Esta asignatura es impartida por el Departamento de Ingeniería de Computadoras, y su objetivo (de la asignatura), según la modificación del año 2010 del plan de estudios [1], es lograr que el alumno:

- Desarrolle la comprensión del diseño y construcción de un sistema de cómputo.
- Sea capaz de centrar la atención en la frontera software-hardware, explorando los niveles del hardware conectados a este punto de contacto.
- Comprenda el concepto de programa almacenado, la representación de las instrucciones en la memoria de una computadora y los pasos para su ejecución.
- Incorpore temas de diseño básicos y elementos en técnicas digitales.
- Introduzca los conceptos de programación en lenguaje ensamblador.

Previamente, en esta asignatura, se enseñaba lenguaje C en un módulo complementario, cuyo objetivo era proveer al alumno de conocimientos básicos del lenguaje para la realización de trabajos prácticos de asignaturas consecuentes como Sistemas Operativos, y Redes y Teleprocesamiento. El módulo complementario tenía una relación prácticamente nula con los contenidos de la asignatura, y normalmente era impartido por un profesor diferente al del módulo principal. Esta metodología desaprovechaba oportunidades para reafirmar conceptos teóricos por falta de sincronización entre ambos módulos, e impedía a los alumnos establecer relaciones entre la programación de alto nivel y la de bajo nivel (lenguaje ensamblador). Esta falta de relación no solo dificulta el aprendizaje sino que también produce, en general, que los alumnos adquieran un conocimiento totalmente abstracto de la programación de alto nivel. Esto es perjudicial para comprender problemas de rendimiento de las aplicaciones, que se estudia en asignaturas posteriores como Arquitectura de Computadoras y Sistemas Paralelos (asignatura electiva).

La metodología de evaluación utilizada anteriormente en esta asignatura (para ambos módulos: principal y complementario), consistía en exámenes escritos en papel donde el alumno no podía contar con ningún otro material que no sea el manual del lenguaje ensamblador utilizado. Este sistema de evaluación impedía al alumno depurar sus programas en una computadora y verificar que su ejecución era correcta, y además, impedía a los profesores, que corregían los exámenes, poder detectar si una equivocación del alumno en la resolución de un ejercicio era por simple distracción o por algún fallo conceptual.

El objetivo de este trabajo es mostrar la propuesta didáctica empleada en la asignatura ODC que intenta solucionar la problemática planteada con la propuesta metodológica previa de la asignatura, en relación a la división de la asignatura en dos módulos, principal y complementario (lenguaje C), y mejorar el

sistema de evaluación con respecto a la detección del origen de los errores cometidos por los alumnos en los exámenes.

El presente trabajo se organiza de la siguiente manera. Primero, se presenta el plan de estudios de la carrera y sus incumbencias profesionales, y se fundamenta la ubicación de la asignatura ODC en el plan. En segundo lugar, se expone la propuesta metodológica de la asignatura ODC, planteando los contenidos mínimos y extendidos del programa, los recursos necesarios para impartir las clases, la cantidad y distribución de las horas requeridas de clases y fuera de clases, la metodología utilizada para las clases teóricas y prácticas, una descripción de las actividades prácticas y herramientas de software utilizadas, y la metodología de evaluación y forma de acreditación de la asignatura. En tercer lugar, se detallan los resultados alcanzados empleando la metodología propuesta. Por último, se presentan las conclusiones del trabajo.

2. Plan de estudios, modalidad de acreditación, y fundamentación de la asignatura

La carrera Licenciatura en Ciencias de la Computación, dictada por la Facultad de Informática, está actualmente regida por las ordenanzas del Consejo Superior: 1004/98, 075/2010, 0647/2010 y 0235/2011. Según ellas, el egresado de la carrera debe ser un profesional capacitado para: "Actuar profesionalmente tanto en industrias como en organismos nacionales y privados de todo el país", "Planificar, dirigir y auditar Proyectos de Desarrollo de Software de cualquier escala", "Integrar y aplicar los conocimientos científicos del área", "Diseñar, desarrollar y mantener programas básicos y de aplicación (software)", "Evaluar y poner en funcionamiento el software ya desarrollado", "Efectuar estudios técnico–computacionales de proyectos que involucren uso de computadoras" y "Promover las aplicaciones de la informática a nuevas áreas".

El alumno logra la **acreditación de una asignatura** de tres formas diferentes. Una opción es matriculándose para cursar la asignatura, y superar su evaluación, estipulada por la cátedra (el equipo académico que imparte la asignatura), con una calificación mínima de 4. Una calificación podría comprender cualquier valor entre 0 y 10, y cada cátedra determina qué porcentaje de los exámenes debe estar correcto para alcanzar la calificación mínima de 4 (normalmente ronda el 70 %). Si el alumno no alcanza ese porcentaje para alcanzar la calificación de 4, aún podría obtener una **acreditación parcial** y temporal (2 años de duración) de la asignatura si superase otro porcentaje menor al anterior (por ejemplo: 50 %), también establecido por la cátedra. Esta acreditación parcial podría permitir al alumno cubrir requisitos para matricularse en otra asignatura, y también permite el acceso a la acreditación total de la asignatura mediante nuevas evaluaciones.

Las restantes dos opciones para obtener la acreditación (total) de la asignatura son: (1) presentándose directamente a una evaluación completa de la asignatura (modalidad que denominamos examen libre), y (2) mediante la acep-

tación por equivalencia de contenidos de una asignatura acreditada por otra institución universitaria.

La figura 1 muestra el plan de estudios de la carrera, donde para cada asignatura se indican los requisitos para matricularse, que se indican como "Cursada" cuando se requiere tener la acreditación parcial de asignaturas, y como "Aprobada" cuando se requiere tener la acreditación total de asignaturas. La asignatura ODC, está indicada con el número 11, y para matricularse el alumno debe cumplir con el requisito de haber "cursado" las asignaturas de Matemática Discreta, y Estructuras de Datos y Algoritmos, y "aprobada" la asignatura Elementos de Programación; estas relaciones pueden observarse en la figura 2. A su vez, ODC es requisito directo para poder matricularse en dos asignaturas que dicta el Departamento de Ingeniería de Computadoras: Arquitectura de Computadoras, y Sistemas Operativos.

	Primer Cuatrimestre				Segundo Cuatrimestre		
#	Nombre	Cursada	Aprobada	#	Nombre	Cursada	Aprobada
PRIMER AÑO							
1	ELEMENTOS DE ALGEBRA			3	ANALISIS MATEMATICO I		
2	RESOLUCION DE PROBLEMAS Y ALGORITMOS			4	MATEMATICA DISCRETA	1	
				5	ELEMENTOS DE PROGRAMACION	1 – 2	
SEGUNDO AÑO							
6	FUNDAMENTOS DE CS DE LA COMPUTACION	4 – 5		9	ANALISIS MATEMATICO II		3
7	ESTRUCTURA DE DATOS Y ALGORITMOS	3 – 5	2	10	PROGRAMACION ORIENTADA A OBJETOS	7	5
8	INGLES TECNICO			11	ORGANIZACIÓN DE COMPUTADORAS	4 – 7	5
TERCER AÑO							
12	ARQUITECTURA DE COMPUTADORAS	6 – 11	8	15	PROBABILIDAD Y ESTADISTICA	9	8
13	ANALISIS Y DISEÑO DE SISTEMAS	10	7 – 8	16	SISTEMAS OPERATIVOS	12	11
14	LOGICA PARA CS DE LA COMPUTACIÓN	7	6 – 8	17	TEORIA Y DISEÑO DE BASES DE DATOS	13-14	
CUARTO AÑO							
18	DESARROLLO DE SOFTWARE	15-17	13	21	INTELIGENCIA ARTIFICIAL		14
19	LENGUAJES DE PROGRAMACION	16-17	14	22	ADMINISTRACION Y GESTION DE PROYECTOS	18	
20	REDES Y TELEPROCESAMIENTO	16		23	COMPILADORES E INTERPRETES	19	
QUINTO AÑO							
24	ALGORITMOS Y COMPLEJIDAD	16		26	TESIS LICENCIATURA		
25	OPTATIVA I *			27	OPTATIVA II *		

Figura 1. Plan de estudios de la Licenciatura en Ciencias de la Computación

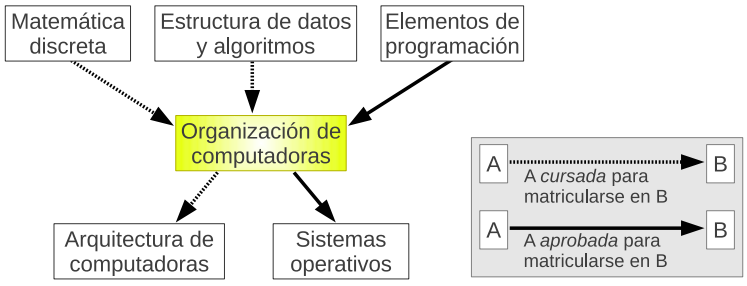

Figura 2. Interrelación de la asignatura objeto y afines

ODC cumple una función de suma importancia en la formación de los alumnos ya que es la primera asignatura de la carrera en la que los alumnos toman

contacto con el área de conocimiento propio del Departamento de Ingeniería de Computadoras. La consecución de los objetivos de la asignatura, que se detallaron en la sección 1 (Introducción), son fundamentales para lograr la construcción de conocimiento básico en el que se apoyan las restantes asignaturas que imparte este departamento (Arquitectura de computadoras, Sistemas operativos, Redes y teleprocesamiento, y algunas otras asignaturas de libre elección), y otras que imparten otros departamentos, como es el caso de Compiladores e Intérpretes, y Lenguajes de Programación.

3. Propuesta metodológica

3.1. Plan de la asignatura

Los **contenidos mínimos** según el plan de estudio son:

- Lenguajes, niveles y máquinas virtuales. Interconexiones. Terminología.
- Nivel de lógica digital. Sistemas combinacionales y secuenciales.
- Sistemas y aritmética de números. Sistema posicional de números. Conversión. Suma y resta de números no decimales. Representación de números negativos. Representación de números en punto flotante. Errores. Códigos de caracteres.
- Unidad Central de Proceso. Diferentes tipos de arquitecturas. Modos de direccionamiento. Memoria. Ciclo de instrucción. Tipos de instrucciones.
- Dispositivos de entrada/salida. Clasificación. Características y descripción de funcionamiento de periféricos convencionales.
- Subprogramas. Transferencia de datos. Subprogramas recursivos.
- Lenguaje ensamblador. Vinculación. Carga.
- Entrada/Salida. Drivers de E/S. Interrupciones. Prioridades, acceso directo a memoria. Procesos de interrupción.

Los **contenidos extendidos**, desarrollados por la cátedra, se presentan a continuación divididos en unidades temáticas:

UNIDAD I: Estructura de una Computadora. Introducción. Lenguajes, niveles y máquinas virtuales. Máquinas multiniveles actuales. Estructuras funcionales básicas. Evolución y generación. Unidad Central de Procesamiento. Unidad de control. Unidad aritmética-lógica. Memoria. Interconexiones. Terminología.

UNIDAD II: Sistemas Numéricos Posicionales. Eficiencia de almacenamiento. Tipos de representación de números. Bases: binaria, octal y hexadecimal. Conversión entre bases. Operaciones aritméticas de números binarios. Representación de números negativos. Signo magnitud. Complemento a la base. Complemento a la base disminuida. En exceso. Ventajas y desventajas de cada una. Rango. Redondeo. Truncar. Extensión de la precisión de los números binarios. Números fraccionarios. Aproximación de las representaciones fraccionarías. Conversión entre fracciones decimales y binarias. Punto flotante. Formato IEEE 754. Conversión. Información no numérica. Códigos ASCII y Unicode.

UNIDAD III: La Unidad Central de Proceso. Concepto de programa almacenado: instrucciones y datos. Principios de abstracción usados para construir sistemas como capas. Compilación e interpretación a través de las capas del sistema. Memoria. Estructura y Organización. Ordenamiento de los bytes. Códigos y detección de errores. Direccionamiento simbólico. Formato de las instrucciones. Registros de las computadoras, generales, de banderas. Procesadores de tres, dos, una y cero dirección. Ciclo de instrucción. Tipos de instrucciones. Movimiento de datos. Bifurcación. Procesadores con registros de propósito general. Procesadores RISC. La arquitectura MIPS. Grupo de instrucciones. Traducción de sentencias de lenguaje C en lenguaje ensamblador MIPS. Traducción de sentencias de lenguaje ensamblador MIPS en instrucciones de lenguaje máquina. Contenido de la memoria y registros durante la ejecución de un programa sencillo. Modos de direccionamiento. Objetivos. Modos de direccionamiento directos e indirectos. Inmediato, directo, registro, base, relativo. Código independiente de la posición

UNIDAD IV: Dispositivos de entrada salida. Métricas. Latencia. Productividad. Ancho de banda. Clasificación de los dispositivos de entrada-salida. Características y descripción de funcionamiento de periféricos convencionales. Terminales. Módems. Ratones. Impresoras. Unidades de disco y cintas magnéticos. Unidades ópticas. Discos magnéticos. Latencia. Tiempo de búsqueda. Tiempo de rotación. Velocidad de transferencia. Densidad de almacenamiento. Buses de Interconexión. Tipos de buses. Arbitración del bus. Métricas para evaluar un bus.

UNIDAD V: Sub-programas. Modelos de programación para implementar sub-programas. Instrucciones de invocación y retomo de sub-programas en procesador MIPS. Instrucciones de invocación y retomo de sub-programas en otros procesadores: Intel, Motorola. Comparación. Ventajas de cada uno. Mecanismos de hardware asociados. Parámetros, transferencia de datos. Pasaje de parámetros en registros y posiciones de memoria. Convención de registros. Procedimientos anidados. Área de parámetros. Asignación estática y dinámica de memoria. Pasaje de parámetros por la pila. Bloque de activación. Sub-programas recursivos.

UNIDAD VI: Lenguaje ensamblador. Vinculación. Carga. Características de un lenguaje de bajo nivel. Lenguaje y programa ensamblador. Formato de instrucciones. Instrucciones, pseudo-instrucciones, directivas. Símbolos locales y globales. Proceso de traducción a lenguaje máquina. Tabla de símbolos. Organización. Ensamblador de dos pasadas. Tiempo y operaciones en tiempo de ensamblado, carga y ejecución. Ensambladores y cargadores reubicables. Ensamblado de sub-programas independientes. Archivo objeto. Formato. Vinculación. Seudo-operaciones para símbolos externos y puntos de entrada. Técnicas de vinculación. Tabla de símbolos para vinculación de sub-programas ensamblados en forma independiente.

UNIDAD VII: Entrada-Salida. Programación directa de E/S con CPU. Direccionamiento de unidades de E/S. Programación de E/S. Puertos aislados y mapeo de memoria. Protocolos de E/S. Registros de interfases para datos y para control y estado. Simulación de MIPS. Operaciones de entrada. Ope-

raciones de salida. Drivers de E/S. Polling. Costo del Polling. Interrupciones. Estructura básica. Condiciones generales. Sistemas simples con interrupciones. Registros en MIPS para soportar interrupciones, para habilitación e inhibición de interrupciones. Prioridades. Identificación. Vectores de interrupción. Interrupciones internas. Excepciones. Llamadas al sistema. Acceso directo a memoria. Programación de un canal de DMA.

UNIDAD VIII: El nivel de lógica digital. Sistemas combinacionales y secuenciales. Sistemas combinacionales. Descripción de una función lógica. Tabla de verdad. Ecuación booleana. Forma canónica con términos mínimos y máximos. Simplificación de funciones lógicas. Diagramas de Veitch. Teoremas de De Morgan. Lógica AND, OR, NAND, NOR. Implementación de funciones lógicas. Circuitos físicos. Lógica positiva y negativa. Equivalencia de las funciones lógicas y los circuitos físicos. Circuitos AND, OR, NAND, NOR. Niveles integración. Circuitos integrados. Integración de bajo nivel SSI. Cantidad mínima de circuitos integrados para implementar una función lógica. Integración de mediano nivel MSI. Ecuaciones lógicas de multiplexores y decodificadores. Multiplexores y decodificadores para la implementación de funciones lógicas. Integración de alto nivel LSI. Circuitos combinacionales completos. RAM, ROM, EPROM. Arreglos lógicos programables PLA. Implementación de funciones lógicas con circuitos combinacionales LSI. Sistemas secuenciales. Autómata de Mealy. Autómata de Moore. Sistemas secuenciales sincrónicos. Integración de bajo nivel SSI. Flip-flops. Tipos de diagramas de estado y tablas de verdad de los flip- flops D, R-S, T, J-K. Implementación de contadores. Contador de Moebius. Registros. Buses. Sistemas secuenciales sincrónicos de control.

Los **recursos físicos** utilizados para impartir las clases son: un aula con proyector y pizarra, un laboratorio de computadoras, y una plataforma de educación a distancia. El laboratorio está conformado por computadoras personales, con sistema operativo GNU/Linux, el compilador de lenguaje C denominado GCC[2], depurador GDB[3], y simuladores de la arquitectura MIPS[4]: SPIM[5] y MARS[6]. La plataforma de educación a distancia utilizada se denomina PEDCO (Plataforma de Educación a Distancia del COmahue)[7], basada en moodle[8], que los alumnos pueden acceder para descargar el material de la asignatura (apuntes, diapositivas de clases, trabajos prácticos, etc.), utilizar el foro, e informarse de comunicaciones de la cátedra.

Con respecto a los **recursos humanos**, la cátedra está conformada por un profesor responsable de la asignatura, un asistente encargado de la realización de trabajos prácticos, y un ayudante de clases prácticas.

El alumno logra la **acreditación de la asignatura** si realiza bien por lo menos el 70 % de de cada uno de los temas evaluados en los dos exámenes prácticos (o instancias de recuperación), un examen teórico, y un informe de un trabajo realizado (en grupo de dos personas) por el alumno. Si el alumno no alcanza ese 70 %, pero supera el 50 %, obtiene la acreditación parcial de la asignatura.

La **bibliografía** básica recomendada es la siguiente:

- David A. Patterson, John L. Hennessy. Estructura y diseño de computadores: La interfaz hardware/software. 2da edición. 2011. Editorial Reverté S.A. ISBN-10: 8429126201. ISBN-13: 9788429126204.
- Andrew S. Tanenbaum, Todd Austin. Structured Computer Organization. 6th edition. 2012. Prentice Hall. ISBN-10: 0132916525. ISBN-13: 9780132916-523.
- Brian W. Kerninghan, Dennis M. Ritchie. El Lenguaje de Programación C. 2da edición. 1991. Pearson Educación. ISBN-10: 9688802050, ISBN-13: 9789688802052.

La bibliografía complementaria es:

- John F. Wakerly. Microcomputer Architecture and Programming. 1st edition. 1981. John Wiley & Sons. ISBN-10: 0471052329. ISBN-13: 978047105-2326.
- William Stallings. Computer Organization and Architecture: Designing for Performance. 9th revised edition. 2012. Pearson Education. ISBN-10: 027376-9197. ISBN-13: 9780273769194.

La **carga horaria** del cursado de la asignatura es de 160 horas en total, distribuidas en 16 semanas. Cada semana involucra 10 horas de dedicación, repartidas en 4 horas de clases teóricas, 4 horas de clases prácticas (en laboratorio), y 2 horas de consultas fuera de clase.

3.2. Metodología de clases teóricas y prácticas

Desde esta asignatura se espera mejorar ciertas competencias y habilidades de los alumnos. En principio, se pretende que el estudiante adquiera los *conocimientos específicos* de la asignatura, es decir, los conceptos teóricos y su aplicación para resolver problemas que se le presenten.

Como complemento a estos conocimientos específicos, el egresado de la carrera deberá contar con diversos recursos metodológicos que le permitan desempeñarse exitosamente en su profesión (según incumbencias profesionales descriptas en la sección 2), y por lo tanto se planea que en esta asignatura el estudiante mejore las *capacidades metodológicas* de:

1. Analizar y sintetizar la información.
2. Organizar y planificar sus actividades de aprendizaje.
3. Resolver problemas y tomar decisiones.
4. Trabajar en equipo, compartiendo los conocimientos y sabiendo comunicarlos.
5. Aprendizaje autónomo.
6. Búsqueda y valoración de la información.
7. Comprensión de textos técnicos escritos en idioma inglés.

Para adquirir los conocimientos específicos y las capacidades metodológicas detalladas, se desarrollan diferentes actividades en dos tipos de clases: teóricas y

prácticas. Las clases teóricas intentan mejorar las *capacidades metodológicas*: 1, 2, 6 y 7. Las clases prácticas, se enfocan en las *capacidades metodológicas* 2, 3, 4, 5 y 7.

Las clases se imparten dos días a la semana, y la planificación de cada día es la siguiente: se dedican 2 horas de clases teóricas seguidas de 2 horas de clases prácticas, con un intervalo de descanso de 15 minutos entre la clase teórica y práctica. En contraposición a una planificación que dedique un día para clases teóricas y un día para clases prácticas, este esquema permite al alumno: un mayor dinamismo en sus tareas diarias, avanzar en sus conocimientos teóricos y prácticos de manera progresiva para aumentar la asimilación de los mismos, leer bibliografía recomendada por el profesor, y consultar dudas en una clase próxima en el tiempo (evitando una diferencia de 7 días).

El primer día de clases del curso se le hace entrega al estudiante de un cronograma que indica las actividades (clases o evaluación) a desarrollar durante cada día del curso. Asimismo, se entrega el "Plan de la asignatura" descripto en el apartado anterior. La cantidad media de alumnos matriculados por curso es de aproximadamente 35, aunque la asistencia a clases es un poco menor, alrededor de 22 alumnos. Esto se debe a que la asistencia a clases no es obligatoria, para permitir el estudio a alumnos que trabajan (en muchos casos, la baja condición económica impide que puedan dedicarse de manera completa al estudio y tengan que compatibilizarlo con tareas laborales; cabe mencionar que la educación en universidades públicas, como ésta, es libre y gratuita).

Las clases teóricas son del tipo expositivas, se utiliza un proyector para mostrar diapositivas y la pizarra para resolver algunos detalles y preguntas de los alumnos. Se abordan los temas desde un punto de vista general planteando los problemas propios de la organización y funcionamiento de las computadoras, y se complementa explicando diferentes soluciones aportadas por distintas arquitecturas, puntualizando ventajas y desventajas de cada una.

Las clases prácticas tienen el objetivo de intentar aplicar la mayoría de los conocimientos teóricos a la resolución de casos y problemas concretos, permitiendo al alumno alcanzar una comprensión más acabada de los temas de la asignatura, así como la adquisición de las competencias.

Las múltiples actividades de las clases prácticas están enmarcadas dentro de lo que denominamos "trabajos prácticos". Cada trabajo práctico permite al alumno construir conocimiento indispensable para la resolución del siguiente trabajo práctico. Las distintas actividades de un trabajo práctico construyen conocimiento base para la siguiente actividad del mismo trabajo práctico. A su vez, los ejercicios que plantea cada actividad son graduales y construyen conocimiento base para los siguientes ejercicios de la actividad. De esta manera, se va construyendo a partir de problemas y conceptos simples, otros con mayor grado de complejidad.

Al comenzar cada práctico, se realiza una exposición explicando el objetivo del práctico, los temas que se van a estudiar, se explica el uso de las herramientas a utilizar, y se mencionan posibles dificultades con que los alumnos pueden encontrarse. Durante las clases prácticas, los alumnos son guiados por los docen-

tes para alcanzar sus objetivos. Al concluirse el tiempo estipulado de resolución de cada trabajo práctico, se resuelven ciertos ejercicios fundamentales y abarcadores que están incluidos en el trabajo práctico, y aquellos que los alumnos soliciten porque han encontrado dificultades. La resolución de los ejercicios se realiza con la atención de la totalidad de los alumnos presentes, utilizando la pizarra, la computadora y el proyector, y promoviendo la participación de los alumnos para llegar a la solución.

3.3. Actividades prácticas y herramientas de software utilizadas

La elección de la arquitectura MIPS como base para la enseñanza del funcionamiento de una computadora y la programación en lenguaje de bajo nivel, o ensamblador, se debe a la simpleza de la arquitectura, la buena bibliografía que existe y la posibilidad de utilizar el simulador en diferentes arquitecturas y sistemas operativos (permitiendo que los alumnos continúen las prácticas fuera de clases). Como contrapartida, los alumnos no trabajan sobre una arquitectura física real, aunque esto se complementa con los ejercicios en el lenguaje C de alto nivel, que si se realizan sobre la máquina real. Las actividades prácticas buscan, siempre que sea posible, promover al alumno a relacionar la programación de alto y bajo nivel. Esta relación intenta evitar principalmente que los alumnos adquieran un conocimiento abstracto de la programación de alto nivel, que podría dificultar la comprensión de problemas de rendimiento de las aplicaciones (que se estudia en asignaturas de años más avanzados).

Las actividades prácticas se realizan en grupos de dos personas, y cada grupo utiliza una computadora del laboratorio. Las 64 horas dedicadas a actividades prácticas se distribuyen en un total de 7 trabajos prácticos. En cada trabajo práctico se indica la duración $a+e$, donde a representa la cantidad de clases que el alumno debe dedicar para su resolución, y e representa la cantidad de clases utilizadas por el profesor para resolver ejercicios fundamentales y abarcadores del mismo (metodología descripta en el apartado anterior); en el último trabajo práctico las horas son solo de dedicación del alumno (no existe e). Los trabajos prácticos se describen a continuación:

1. Componentes de un sistema de computación. El objetivo es comprender la relación entre los componentes de un sistema de computación: programas de aplicación, sistema operativo y hardware. Duración: 3+1 clases. Las actividades abarcan los siguientes temas:
   a) Introducción al entorno GNU/Linux y el intérprete de comandos.
   b) Introducción al lenguaje de programación C: ciclo de compilación, uso de GCC, estructura de un programa, depuración utilizando GDB.
   c) Introducción a los simuladores de MIPS.
   d) Código en ejecución: concepto de proceso, comandos para administrar procesos en GNU/ Linux.
2. Sistemas de numeración y representación digital de datos. Su objetivo es comprender el sistema de numeración posicional, la conversión entre sistemas de distintas bases, la representación binaria de números enteros y de coma

flotante, la suma y resta de números enteros en binario, y la representación binaria de texto. Duración: 3+1 clases. Las actividades abarcan los siguientes temas:

   a) Sistemas de numeración.
   b) Representación y aritmética de números enteros.
   c) Representación de números de coma flotante estándar IEEE-754.
   d) Representación de caracteres UTF-8.
   e) Tipos de datos en el lenguaje C, arreglos, estructuras y uniones, conversión de tipos.
   f) Tipos de datos en MIPS.

3. Instrucciones, registros y memoria. El objetivo es estudiar la codificación de instrucciones, la utilidad de los registros, los modos de direccionamiento y la organización y gestión (básica) de memoria de los programas en ejecución. Duración: 4+1 clases. Las actividades abarcan los temas:

   a) Modos de direccionamiento.
   b) Organización de la memoria: gestión de memoria en C y MIPS, orden de bytes, alineación de datos en memoria.
   c) Registros.
   d) Representación de instrucciones: código de instrucción, direcciones de memoria, formato de instrucción.

4. Conjunto de instrucciones. Su objetivo es conocer los diferentes tipos de instrucciones disponibles en las arquitecturas, y familiarizarse con el uso de las mismas en el desarrollo de programas. Duración: 4+1 clases. Las actividades involucran programación en MIPS y lenguaje C.
5. Subrutinas. El objetivo es comprender cómo funcionan las subrutinas y las distintas maneras de pasar argumentos, relacionando el uso de subrutinas en lenguajes de alto nivel con implementaciones en lenguaje ensamblador. Duración: 4+1 clases. Las actividades involucran programación de subrutinas en MIPS y lenguaje C.
6. Dispositivos de entrada y salida, interrupciones, excepciones y llamadas al sistema. El objetivo es comprender, mediante la programación en lenguaje ensamblador, las técnicas de comunicación con los dispositivos de entrada y salida, y el tratamiento de las interrupciones, excepciones y llamadas al sistema. Duración: 4+1 clases. Las actividades abarcan los siguientes temas:

   a) Dispositivos de entrada y salida en MIPS.
   b) Llamadas al sistema en MIPS.
   c) Interrupciones y excepciones en MIPS.

7. Circuitos combinacionales y secuenciales. El objetivo es comprender el funcionamiento de los circuitos digitales combinacionales y secuenciales. Duración: 4 clases. Al finalizar la cuarta clase, cada grupo de dos alumnos deberá entregar un informe con la resolución de todos los ejercicios. Las actividades incluyen fundamentalmente temas de compuertas lógicas, multiplexores, y *flip-flops*.

3.4. Evaluación

En la plataforma PEDCO, se habilita un test para cada práctico, que cada alumno puede opcionalmente realizar y le permite auto-evaluarse antes de realizar el examen correspondiente. Esta evaluación no se califica, y el alumno puede realizar múltiples veces el mismo test.

Cada uno de los dos exámenes de práctica se realizan en el laboratorio de computadoras, son exámenes individuales, destinando una computadora por alumno. Los exámenes consisten en la resolución de una serie de ejercicios de programación, en un tiempo determinado, y como recursos pueden tener cualquier tipo de información en papel o digital (apuntes, resolución de trabajos prácticos, diapositivas de teoría, libros), y acceso total a la red Internet mediante la computadora. Este sistema lo hemos denominado "evaluación a mundo abierto", debido a que los estudiantes disponen de acceso a Internet durante el examen, siendo la intención recrear posibles situaciones de su futura vida profesional, a excepción de que no se permite ningún tipo de interacción entre los estudiantes. Este sistema de evaluación, en donde el alumno puede utilizar todas las herramientas de programación, depuración, y manuales disponibles, hace que el alumno que tiene los conceptos claros pueda encontrar la solución de cada ejercicio de los exámenes; y así los errores se pueden relacionar directamente con fallos conceptuales.

En el examen teórico se pide desarrollar ciertos temas estudiados en las clases teóricas, que son seleccionados al azar. El examen se escribe en papel (sin computadora), y el alumno no tiene acceso a ningún material de lectura.

Los exámenes de práctica tienen sus correspondientes instancias de recuperación, no así el examen teórico y la evaluación del informe entregado (resolución del último trabajo práctico).

4. Resultados

En este apartado se indican los resultados cuantitativos y cualitativos relacionados a los alumnos matriculados que logran o no acreditar la asignatura. En lo siguiente, consideramos que un alumno "aprueba" un examen o la asignatura cuando realiza correctamente el 50 % de cada tema evaluado, es decir, el porcentaje exigido para la acreditación "parcial" de la asignatura.

La metodología de evaluación utilizada para la parte práctica de la asignatura podría hacer pensar que se ofrecen demasiadas facilidades a los alumnos para aprobar los exámenes, a causa del acceso a una enorme cantidad de información (desde apuntes, libros, programas anteriormente realizados, hasta acceso total a Internet), y la dificultad para evitar interacciones entre los alumnos mediante comunicaciones por Internet. Sin embargo, un porcentaje de aprobación de exámenes que se encuentra en alrededor del 60 % de los alumnos matriculados, nos permite confirmar que los exámenes no resultan simples para todos.

Respecto a la precisión del sistema de evaluación, y tomando como muestra estadística a los alumnos que asisten a clases, existe una muy fuerte correlación entre el resultado de los exámenes y el concepto que la cátedra construyó de cada

alumno en base al rendimiento observado durante las clases (entre otros: participación y trabajo activo, avance en la resolución de los trabajos prácticos de acuerdo al cronograma). Más del 95 % de los alumnos que asisten regularmente a las clases y realizan las actividades propuestas (resuelven todos los trabajos prácticos, realizan los test de autoevaluación y consultan dudas), aprueban los exámenes de práctica realizando correctamente más del 80 % de cada tema evaluado.

Casi la totalidad de los alumnos que aprueban los exámenes de práctica son aquellos que asisten regularmente a clases y cumplen con las actividades propuestas. De los alumnos que no asisten regularmente a clases, solo unos pocos logran aprobar estos exámenes. Los motivos de que un 40 % de los alumnos matriculados no logren aprobar la asignatura, en general, son prácticamente los opuestos a los del porcentaje de aprobados. No asisten regularmente a las clases tanto teóricas como prácticas, no resuelven los trabajos prácticos y, normalmente, solo asisten a clases de consultas. Observada esta situación, se les aconseja a todos los alumnos, seguir en lo posible la metodología propuesta. Cabe destacar que la cátedra solo se ocupa de analizar posibles causas académicas sobre el rendimientos de los alumnos en la asignatura, mientras que otros tipos de causas son analizadas por diferentes áreas de la institución (principalmente el área de Tutorías y Becas)[1].

Normalmente, un 90 % de los alumnos que aprueban los exámenes prácticos también aprueban el examen teórico. Además, casi la totalidad de ellos alcanzan el nivel de acreditación total de la asignatura.

5. Conclusiones

En este trabajo se ha presentado la metodología de enseñanza y aprendizaje utilizada por el equipo de cátedra de la asignatura Organización de Computadoras, correspondiente al grado de Licenciatura en Ciencias de la Computación de la Universidad Nacional del Comahue, Argentina. La metodología soluciona los inconvenientes presentados por la metodología utilizada anteriormente en la asignatura: ausencia de relación entre la programación de alto y bajo nivel, y dificultad para detectar las causas de los errores que los alumnos cometían en los exámenes. Por un lado, se han adaptado las clases teóricas y prácticas, y se han desarrollado nuevas guías de trabajos prácticos y actividades que fomentan al alumno a establecer permanentes comparaciones de la programación de alto y bajo nivel. Por otro lado, un sistema de evaluación en computadora y "a mundo abierto" (incluso con acceso libre a Internet), permite al alumno disponer de toda información y herramientas necesarias para encontrar las soluciones a los ejercicios de los exámenes, y comprobar si funcionan adecuadamente; situación que

[1] Las causas más frecuentes que afectan el rendimiento de los alumnos son: incompatibilidad con los trabajos (muchos alumnos, en esta etapa de sus vidas, tienen actividades laborales parciales o completas), problemas de formación en etapas educativas anteriores, y desmotivación por el estudio.

intenta aproximarse al ambiente de trabajo real de un profesional. Esta metodología de evaluación permite, a los profesores, relacionar los errores cometidos por los alumnos, en los exámenes, con fallos conceptuales y no simples distracciones. Así, el proceso de evaluación logra un resultado más justo, y permite determinar con mayor precisión posibles falencias en la enseñanza.

Los resultados de aplicar la metodología propuesta indican una mejoría en la calidad del aprendizaje alcanzado por los alumnos. Esta realidad no solo se fundamenta en los índices obtenidos para esta asignatura, sino también en el seguimiento del desempeño de los alumnos en las materias siguientes. Para la cátedra, los resultados significan un acierto en la metodología, tanto en la forma de impartir las clases como en la forma de evaluación. Sin embargo, es razón actual de estudio, de la cátedra y de la institución, lograr mejorías en la cantidad de alumnos aprobados cuando los estudiantes no pueden asistir (por distintas circunstancias) a clases.

Referencias

1. Plan de la carrera Lic. en Cs. de la Computación: (Accedido en Abril de 2014) http://faiweb.uncoma.edu.ar/images/ordenanzas/Carreras/ord_0075_2010_23.pdf.
2. GCC, the GNU Compiler Collection: (Accedido en Abril de 2014) http://gcc.gnu.org/.
3. GDB: The GNU Project Debugger: (Accedido en Abril de 2014) http://www.sourceware.org/gdb/.
4. David A. Patterson, J.L.H.: Estructura y diseño de computadores: La interfaz hardware/software. 2da edn. Editorial Reverté, S.A. (2011) ISBN-10: 8429126201, ISBN-13: 9788429126204.
5. SPIM: A MIPS32 Simulator: (Accedido en Abril de 2014) http://spimsimulator.sourceforge.net/.
6. MARS (MIPS Assembler and Runtime Simulator): (Accedido en Abril de 2014) http://courses.missouristate.edu/kenvollmar/mars/.
7. Plataforma de Educacion a Distancia, Universidad Nacional del Comahue: (Accedido en Abril de 2014) http://pedco.uncoma.edu.ar/.
8. Moodle - Open-source learning platform: (Accedido en Abril de 2014) https://moodle.org/.

Una experiencia en la enseñanza de la programación para la permanencia de los alumnos de Ingeniería Electrónica

Norma Arellano, María Verónica Rosas, Mariela E. Zuñiga, Jacqueline Fernandez , Roberto Guerrero
Universidad Nacional de San Luis,
Ejército de los Andes 950
Tel: 02664 420823,
San Luis, Argentina
{nmare, mvrosas, mezuniga, jmfer, rag}@unsl.edu.ar

Resumen La programación es una disciplina de las Ciencias de la Computación con diversas y variadas aplicaciones. Aprender a programar problemas computacionales representa un profundo desafío intelectual. La verdadera dificultad no reside en expresar la solución del problema en términos de instrucciones elementales de un lenguaje de programación específico, sino en la resolución del problema propiamente dicho. El proceso de encontrar una solución adecuada al mismo provoca en el alumno un conflicto cognitivo por no dominar un sistema de estrategias que le permitan afrontar la situación de manera satisfactoria. Del análisis de la realidad en el aula que se viene presentando con los alumnos que ingresan a la carrera de Ingeniería, se ha detectado que el rendimiento académico de los estudiantes es cada vez menos satisfactorio agravado por la complejidad intrínseca de los contenidos de la asignatura. En este trabajo se presenta la experiencia realizada durante el dictado del primer curso de enseñanza de la programación que reciben los alumnos de primer año en la carrera de Ingeniería Electrónica de la Facultad de Ciencias Físico-Matemáticas y Naturales, de la Universidad Nacional de San Luis, Argentina.

Palabras clave: Resolución de Problemas, Pensamiento Computacional, TICs, Enseñanza y Aprendizaje de la Programación, B-Learning

Abstract: Programming is a computer science discipline that has many applications in the real world. Learning to program computational problems represents an intellectual challenge. The main difficulty lies not in expressing the solution in terms of elementary instructions, but in solving the problem itself. To find an appropriate solution for a problem causes with the student a cognitive conflict due of they do not dominate a system of strategies in order to respond satisfactorily. The analysis of what happen in the classroom today with the first level students to the Engineering, exhibit that the academic performance of students is becoming less satisfactory added to the complexity of the subject content. This paper presents the experience made during the first programming course the students receive in the first year of electronic

engineering, Faculty of Physics and Mathematics, and from the National University of San Luis, Argentina.

Keywords: Problem Solving, Computational Thinking, ICTs, Learning and Teaching Programming, B-Learning

1. Introducción

Aprender a programar computadoras involucra un proceso en el que la persona pone en juego una gran variedad de habilidades y capacidades. En el proceso de aprender a programar, es dable esperar que los alumnos adquieran habilidades y desarrollen capacidades fundamentales en la resolución de problemas. No se está simplemente aprendiendo a programar sino que al mismo tiempo se está programando para aprender; pues, además de incorporar conceptos computacionales, simultáneamente se está aprendiendo estrategias para solucionar problemas. Habilidades que no sólo son de utilidad para los expertos en computación sino, también, para muchas de las actividades cotidianas de la vida de las personas, sin distinción de edad, origen, intereses u ocupación [6, 8].

Existen diferentes modelos de resolución, Polya, Mason, Burton, Stacey, Bransford, Guzmán entre otros, que establecen cómo actuar frente a los problemas. Los modelos de Mason, Burton y Stacey, analizan los comportamientos de los sujetos reales involucrados en la resolución de problemas y sus características propias (reacciones, sensaciones, emociones, altibajos, retrocesos o inspiraciones) considerando positivos los atascos y errores que pudiesen surgir. Bransford y Guzmán proponen hábitos mentales útiles mientras que el método heurístico definido por el matemático Polya en 1957, describe el comportamiento de quien resuelve el problema al recorrer secuencialmente determinadas fases fundamentales pasando de una a otra sólo cuando se ha concluido con la anterior. El modelo de Polya define un marco conceptual que consiste de cuatro etapas fundamentales: entender el problema donde se identifican los datos dados y se definen las incógnitas, trazar un plan donde se establece la relación entre los datos y las incógnitas y se buscan patrones, ejecutar el plan de la posible solución donde se comprueba si son los pasos correctos y finalmente revisarlo, etapa de la visión retrospectiva donde es muy importante observar qué fue lo que se hizo, verificar el resultado y el razonamiento seguido. Siendo el análisis el elemento fundamental del proceso de resolución. Esta estrategia permite transformar el problema en una expresión más sencilla que se sepa resolver. Esta metodología puede pensarse como el instrumento heurístico que permite descubrir, interrelacionar y desarrollar el pensamiento crítico y reflexivo, la creatividad y capacidad de inventiva con el pensamiento computacional necesario para implementar la transformación [1, 15].

En la era digital el pensamiento computacional es una habilidad que se encuentra al alcance de todos. Es un proceso de solución de problemas que se caracteriza por organizar los datos de manera lógica para su análisis, representar datos mediante abstracciones, formular soluciones a problemas computacionales y automatizarlas algorítmicamente. El pensamiento computacional implica un pensamiento recursivo y un procesamiento paralelo [13,15]. Las abstracciones son las herramientas mentales

de la computación. El poder de las herramientas mentales aumenta con el poder de las herramientas metálicas, la computadora. Computación, en consecuencia, es la automatización de las abstracciones [16].

Los avances tecnológicos han ampliado significativamente la capacidad de resolución de los problemas y, por lo tanto, los estudiantes no sólo necesitan aprender sino practicar nuevas habilidades como la del pensamiento computacional que les permitirá aprovechar el potencial generado por los rápidos avances en las TICs [4, 12]. Es necesario conocer las potencialidades de estas herramientas para hacer un uso adecuado de las mismas.

En este contexto se encuentra el Blended learning o aprendizaje que combina la enseñanza presencial con la enseñanza no presencial. Modelo híbrido donde se deja de lado la "enseñanza asistida por computadora" para incorporar el "aprendizaje basado en la computadora" y es incorporado como complemento de la modalidad presencial para definir estrategias que permitan mejorar los resultados académicos de los alumnos [17, 18].

La universidad es consciente de su rol en la sociedad del conocimiento, donde se ha pasado de un paradigma concentrado en la enseñanza a un paradigma concentrado en el aprendizaje apoyado en la construcción colaborativa del conocimiento. Es pertinente entonces renovar los grados de innovación con el objetivo de favorecer la reducción del nivel de deserción, mejorar el desempeño académico de todos los estudiantes y establecer bases robustas que le faciliten su permanencia y egreso del sistema universitario [7].

2. Contexto e identificación de la problemática.

Cada año estudiantes provenientes de distintos lugares del país y de diferentes instituciones educativas, ingresan a la UNSL. Es necesario que los docentes de asignaturas del primer año en carreras universitarias deban contribuir con la adaptación de los estudiantes a la vida universitaria. Proceso que supone la apropiación de nuevas responsabilidades, normativas y hábitos; todo un desafío para muchos ingresantes, constituyéndose en un factor que incide en su rendimiento académico.

El presente trabajo presenta la experiencia llevada a cabo con los alumnos de la asignatura Fundamentos de la Informática de la carrera Ingeniería Electrónica con Orientación en Señales digitales de la Facultad de Ciencias Físico, Matemáticas y Naturales (FCFMyN). Su dictado es responsabilidad de docentes del Dpto. de Informática de dicha facultad y representa el primer curso de programación que reciben los alumnos de esta carrera. La asignatura se organiza en seis horas semanales de clases teórico-prácticas hasta completar, según el plan de la carrera, un total de 105 horas. Inicialmente las clases teórico-prácticas obligatorias completaban un crédito horario de 90 horas completándose el crédito total con clases de consultas que no resultaban motivadoras para los alumnos y, por lo tanto, no eran aprovechadas.

Del análisis de los resultados académicos obtenidos en los últimos períodos, fue posible identificar algunos aspectos críticos de los alumnos que ingresan año a año y que pudieron influir de forma negativa en su desempeño:

- Las "habilidades académicas" que caracterizan a la mayoría de los alumnos y que se relacionan principalmente con la falta de hábito de estudio, problemas con la comprensión de textos y dificultad en la resolución de problemas.
- La falta de motivación o poco interés de los alumnos. Potenciado por el hecho de que están acostumbrados a trabajar y contar con diferentes recursos tecnológicos que, si bien son utilizados, no son considerados el eje principal.
- Bajo umbral de tolerancia al fracaso.

Éstos y otros factores, desde la perspectiva de los alumnos ingresantes, quedan desfavorablemente expuestos en el alto porcentaje de deserción y el bajo rendimiento obtenido.

Así fue que, teniendo en cuenta la problemática detectada y considerando las características propias de los alumnos ingresantes en su calidad de "nativos digitales" y a su manejo cuasi innato de la tecnología se decide el estudio y definición de diferentes propuestas metodológicas que contribuyan a mejorar los procesos educativos involucrados en la enseñanza de la programación y el análisis de la factibilidad de su puesta en marcha. El proceso se desarrolló en forma gradual. En una primera etapa se realizó la incorporación de prácticas de laboratorios complementarias con la intención de motivar a los alumnos y en una segunda etapa se integró el uso del campus virtual definiendo un nuevo canal de comunicación y cuyo potencial provee, además, la posibilidad de definir actividades de aprendizajes significativos que complementan el trabajo que se venía realizando hasta el momento.[2] [10].

3. Justificación de las herramientas utilizadas

Para diseñar las prácticas de laboratorio de programación fue necesario seleccionar cuidadosamente el software con el cual se iba a trabajar. Las distintas herramientas de software que se utilizaron se fueron incorporando en base a su complejidad de uso y en correspondencia con las prácticas de aula, dando soporte a las mismas.

Se incorpora el uso del Lenguaje TIMBA (Terribly Imbecile Machine for Boring Algorithms) que fuera ideado con fines educativos por un grupo de docentes de la UNSL, dirigido por el Ing. Hugo Ryckeboer. Dicha herramienta fue desarrollada en respuesta a la necesidad de contar con un pseudo-lenguaje simple que permite introducir al alumno en la noción de algoritmo y los conceptos básicos necesarios para la construcción de los mismos basado en el paradigma de la programación estructurada. TIMBA es un lenguaje que permite la definición de algoritmos combinando las tres estructuras de control básicas y que consisten, en definitiva, de una secuencia de órdenes a un ejecutor, denominado UCP, capaz de comprender un conjunto reducido de acciones primitivas y de manipular pilas de cartas [14, 19].

La aplicación Dia (Diagram Editor) es una herramienta muy potente y fácil de aprender que permite la creación de forma sencilla de numerosos tipos de diagramas: UML, de flujo, de red, y cronogramas. La herramienta Dia fue empleada como

complemento y facilitador del proceso de diseño y definición del algoritmo final a través de la realización de diagramas de flujo simples [20].

Para iniciar a los alumnos en la construcción de programas o algoritmos computacionales similares a los lenguajes formales de programación, sin tener que lidiar con las particularidades estrictas de sintaxis, se incorpora un software multiplataforma de distribución libre y gratuito denominado PSeInt (Pseudo code Interpreter). Con la inclusión de esta herramienta se continúa con la profundización del uso de las estructuras de control agregando complejidad a los conceptos ya aprendidos al utilizar TIMBA. Se introducen los conceptos de variables, expresiones, manejo de estructuras de datos y la implementación de la Modularización [21].

La simpleza de sintaxis de esta herramienta le facilita al alumno la tarea de escribir algoritmos en un pseudo-lenguaje similar al lenguaje natural haciendo hincapié en la apropiación significativa de conceptos básicos vinculados a la programación. Presenta un conjunto de ayudas y asistencias, además de brindar algunas herramientas adicionales que le ayuden al alumno a encontrar errores y comprender la lógica de los algoritmos como un paso previo a programar en un entorno integrado de programación más complejo y lenguajes de programación formales como Pascal, C o Java [21].

Para añadir el modelo Blended Learning, se incorporó la utilización de la plataforma virtual Moodle que permite definir un canal más de comunicación e intercambio con los alumnos, generando un laboratorio virtual que logra un ambiente común e integrado de trabajo. Fundamentalmente, con la incorporación de esta herramienta se desea, en una primera etapa, promover el trabajo colaborativo de los alumnos con sus pares, sacar ventaja de la independencia de tiempo y espacio, y definir actividades que resulten más motivadoras como así también propuestas de autoevaluación que orienten al alumnos en su proceso de aprendizaje [22].

4. Experiencia

La asignatura Fundamentos de la Informática de la carrera Ingeniería Electrónica con Orientación en Señales digitales corresponde, según el plan de estudios, al segundo cuatrimestre del primer año. Para fomentar la permanencia de aquellos alumnos que no alcanzan los objetivos en un cuatrimestre y reducir el índice de deserción de los mismos, la FCFMyN desarrolla diferentes actividades, como por ejemplo repetir el dictado de la materia en el primer cuatrimestre del año siguiente.

A partir de los contenidos mínimos que la materia debe cubrir, los conceptos que en ella se dictan se organizan en tres ejes fundamentales: introducción a la informática, lógica proposicional y, finalmente, resolución de problemas y definición de algoritmos, el cual es el eje fundamental de la materia. Este eje se basa principalmente en el desarrollo del modelo descriptivo definido por G. Polya el cual establece las necesidades del alumno para aprender a resolver problemas. Es menester definir soluciones siguiendo un enfoque lógico y algorítmico que permita enfrentar al alumno con la problemática de analizar y resolver problemas de carácter general y problemas computacionales, y la transformación de estos últimos a programas que puedan ser resueltos por una computadora. Para desarrollar las cuatro fases de este

modelo se organizó a los contenidos y sus respectivas actividades prácticas en un esquema en forma de espiral positiva, es decir el método de resolución de problemas se abordó en dos momentos sucesivos, incluyendo dos herramientas distintas, siendo la segunda instancia de mayor complejidad y profundidad que la primera. La etapa de Comprender el problema, requiere de un proceso inicial que promueva en el alumno actividades vinculadas a la resolución de problemas y cuya solución, en principio, puede ser expresada de una manera flexible poniendo en práctica la técnica del Refinamiento Sucesivo y utilizando el lenguaje natural. Una vez que se han precisado los límites del problema, determinado los datos, incógnitas y sus relaciones se continúa con la etapa de Desarrollo de un plan que redundará en la definición del algoritmo. Es en esta etapa donde se incluye en la práctica el uso del Lenguaje Timba con el objetivo de favorecer al alumno en la tarea de aprender a caracterizar el comportamiento del flujo de control de un algoritmo. Una vez determinado el mismo se continúa con la codificación respetando la rigurosidad sintáctica propia del lenguaje. Se obtienen así programas codificados en Lenguaje Timba los cuales posibilitan continuar con las etapas de Ejecución del plan y de Examinar la solución definida para evaluar la eficiencia y eficacia de la misma.

Se continúa luego con la aplicación de cada una de las fases de este modelo pero esta vez incluyendo la práctica con PSeInt cuya potencialidad permite la introducción de conceptos fundamentales para la programación como manejo de variables, estructuración de datos, modularización, entre otros.

Esta experiencia se implementó en dos etapas durante el año 2013.

En la primera etapa, desarrollada durante el primer cuatrimestre, se propuso la incorporación de un horario para laboratorio de programación de modalidad opcional, donde se utilizaron aplicaciones que permiten al alumno verificar y depurar los programas resueltos en las clases prácticas utilizando papel y lápiz. La práctica desarrollada incluyó el uso de la herramienta TIMBA, se realizaron diagramas de flujo con la aplicación Dia y para afianzar los conceptos fundamentales de la algoritmia computacional y acercar al alumno al uso de un entorno de desarrollo integrado de programación se utilizó el software PSeInt.

En la segunda etapa, se continuó con el horario de laboratorio para programación pero, a diferencia de la anterior, con asistencia obligatoria en reemplazo de las clases de consultas que no se aprovechaban. Se continuó con el uso de las herramientas TIMBA y PSeInt, incorporando su editor de diagramas de flujo. Además, con la intención de generar un espacio común de trabajo y promover un espíritu colaborativo en el proceso de enseñanza aprendizaje se incorporó la utilización de la plataforma virtual Moodle favoreciendo la comunicación, el seguimiento y la retroalimentación personalizada.

5. Análisis de los resultados

Los resultados observados durante esta experiencia surgen de la percepción de los docentes en relación con las formas de aprendizaje de los estudiantes y su desempeño académico considerando diferentes elementos como la interacción, el liderazgo y las respectivas prácticas dentro de las actividades colaborativas. Además, se llevó a cabo

una encuesta a los alumnos que asistieron a la asignatura con el objetivo de conocer, desde su punto de vista, los propios hábitos de estudio y la influencia que la nueva estrategia pudo haber tenido en los respectivos procesos de aprendizaje.

5.1. Desde la perspectiva docente

Se considera como positiva a la experiencia en diferentes sentidos:

- Los medios utilizados permitieron una nueva forma de comunicación e intercambio, sustentado en el hecho que los alumnos, en su mayoría, no sólo cuentan con una serie de recursos tecnológicos para la comunicación sino que, además la utilización de los mismos es más frecuente que la de los medios tradicionalmente propuestos por la universidad.
- Al redefinir la propuesta de enseñanza, a partir de la disponibilidad tecnológica y en busca de mejorar la comunicación con los alumnos se favoreció el logro de una participación más activa y predispuesta por parte de los mismos, si bien, cabe destacar que este incremento en la participación se produjo en forma paulatina y con diferentes niveles de entusiasmo. Un aspecto a mejorar es la propuesta de actividades tendientes a promover el trabajo colaborativo entre pares ya que predominó la comunicación alumno-docente y prácticamente no existió la comunicación alumno-alumno.
- Con la incorporación del aula virtual y de sus herramientas se logró un registro digital del progreso individual de cada alumno, la retroalimentación grupal e individual sobre cada actividad evaluada y la especificación de los criterios de evaluación en las diferentes etapas, todo esto influyendo de manera positiva en el ejercicio de la práctica docente.
- La utilización del aula virtual permitió proyectar una nueva estrategia de trabajo que modificó el estilo habitual del equipo docente, logrando una comunicación más activa, integrada y participativa.

5.2. Desde la perspectiva del alumno

Para evaluar la experiencia se elaboró una encuesta individual con el propósito de conocer la opinión de los alumnos con respecto a la nueva estrategia planteada en lo que se refiere a la utilización de la plataforma virtual. La encuesta fue respondida de manera anónima y voluntaria.

En la Figura 1 se observan los resultados de las respuestas de alumnos a la consulta para determinar los conocimientos previos en relación al manejo de herramientas virtuales. El 67% de los encuestados manifiestan estar familiarizados con el uso de la plataforma virtual ya que se utiliza en otras asignaturas previas.

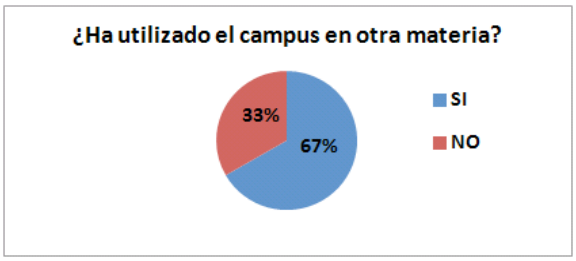

Figura 1

En relación a la frecuencia con la que el alumno accedía a la plataforma, en la Figura 2 es posible observar que el 50% de los alumnos reconocen haber accedido frecuentemente al campus evidenciando que la gran mayoría incorporaron al aula virtual como un recurso importante de consulta. En la Figura 3 se observa que la mayoría reconoce a la incorporación del campus como una continuidad de la práctica de aula que beneficie al proceso de aprendizaje y la minoría lo considera una sobrecarga de dedicación.

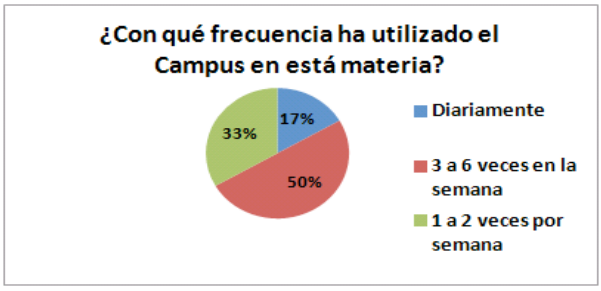

Figura 2

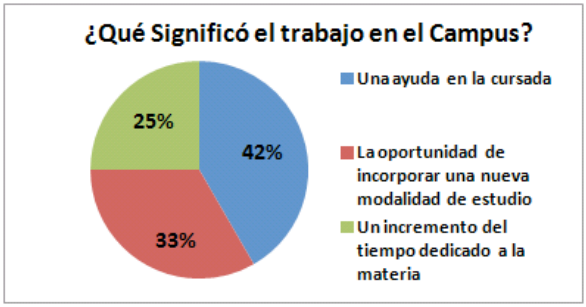

Figura 3

En la Figura 4 se observa que el 92% de los alumnos reconocen como beneficioso la incorporación del uso del aula virtual siendo casi insignificante el porcentaje de alumnos que declaran no tener una opinión formada .

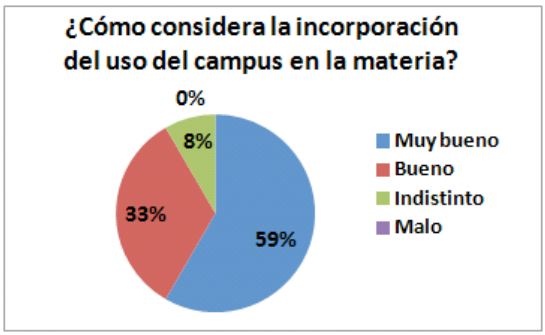

Figura 4

La Figura 5 muestra que la mayoría de los alumnos han accedido al campus con el propósito de cumplir con los requerimientos que fueron realizados por los docentes, en segundo lugar y en un porcentaje significativo, le interesó la retroalimentación del docente al alumno. Un grupo pequeño utilizó al campus como un canal para obtener el material de estudio y un porcentaje casi nulo incorporó el uso de los foros con lo que resulta evidente que un aspecto a mejorar es el promover actividades que favorezcan el trabajo colaborativo.

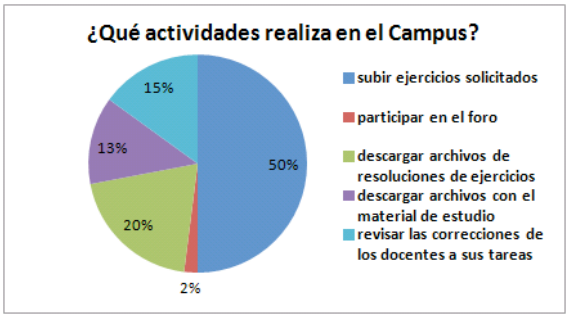

Figura 5

En una reflexión personal solicitada a cada alumno, en la misma encuesta, pudieron ampliar su opinión acerca de la asignatura y a la vez destacar otros beneficios que surgen del uso del campus como el contar con todo el material de estudio en un mismo lugar, la posibilidad de comunicación a través de foros y chat con docentes y compañeros y, una mejor organización del tiempo dedicado a la asignatura. Un porcentaje menor de alumnos planteó la dificultad en el uso adecuado del Campus debido a problemas de conexión a Internet.

6. Conclusiones

Si bien, la intención de llevar a cabo esta experiencia fue abordar una problemática identificada en una carrera específica y en un contexto determinado, es importante enfatizar que esta realidad se reitera en carreras similares en diferentes universidades.

Por lo tanto, se considera significativo innovar en las propuestas de enseñanza en busca de una solución factible que permita intervenir positivamente.

De la experiencia llevada a cabo por este equipo de trabajo se pueden destacar:

- La integración de recursos tecnológicos en el proceso de enseñanza-aprendizaje de la programación tuvo un impacto positivo como estrategia para afianzar la comunicación entre docentes y alumnos, y como facilitador del desarrollo de las capacidades y competencias necesarias en alumnos que se inician en la práctica de programar.
- La formación del pensamiento computacional en la resolución de problemas representa una estrategia potencialmente significativa para fomentar en los alumnos distintas habilidades que promuevan el desarrollo del pensamiento crítico y potenciar su creatividad.

El desafío a partir de esta nueva experiencia podría plantearse en dos grandes ejes:

- Creación y promoción de acciones que favorezcan el trabajo colaborativo para propiciar procesos de enseñanza-aprendizaje más significativos. El proceso de implementar esta nueva estrategia requerirá principalmente la adaptación de la tecnología a esta nueva propuesta y una capacitación continua de los docentes en el rol de tutor.
- Integración de herramientas de desarrollo al campus, generando un laboratorio virtual en la plataforma y logrando un ambiente común e integrado de trabajo. En este sentido, se está trabajando en la incorporación del módulo de programación VPL (Virtual Programming Lab), software de código abierto que permite la gestión de prácticas de programación en el entorno Moodle posibilitando, de esta manera, la integración de herramientas de programación al campus [22,23].

Referencias

1. Polya, George, Cómo plantear y Resolver problemas - Editorial Trillas – ISBN 968-24-0064-3 (1965).
2. Adell, J. Los estudiantes universitarios en la era digital: la visión del profesor. La Cuestión Universitaria, (2011), pp 97-100.
3. Brookshear, J. Introducción a las Ciencias de la Computación, Wilmington Delaware (U.S.A.): Addison-Wesley Iberoamerican, S.A., (1995)
4. Cukierman, U. Las TICs en la Educación de Ingeniería de las Nuevas Generaciones. Información y Comunicación para la Sociedad del Conocimiento, (2009). Córdoba, Argentina.
5. Gries D. The Science of programming. Springer-Verlag. (1981).
6. Helminen, J., Ihantola, P., Karavirta, V., & Malmi, L. How Do Students Solve Parsons Programming Problems?. An Analysis of Interaction Traces. Proceedings of the Eighth Annual International Computing, (2012), pp. 119-126

7. Lovos, E., Gonzalez, A. et al. Estrategias de enseñanza colaborativa para un curso de Programación de primer año de la Lic. en Sistemas. CACIC XVIII (2012).

8. Mac Gaul, M., López, M., Del Olmo, P. Resolución de problemas computacionales: Analisis del proceso de aprendizaje.TE&ET, (2008).

9. Negroponte, N. Ser digital. (1995), Buenos Aires, Argentina: Atlántida.

10. Prensky, M. Digital natives, digital immigrants. On the Horizon, (2001).

11. Prudkin, A. (30 de Junio de 2010). Educ.ar. Obtenido de http://portal.educ.ar/debates/sociedad/cultura-digital/manuel-castells-en-argentina-c.php

12. Rozenhauz, J., Cukierman, U., & Santángelo, H. Tecnología Educativa: Recursos, modelos y metodologías. (2009). Buenos Aires: Pearson.

13. Stager, G. (13 de Enero de 2004). En pro de los computadores. http://www.eduteka.org/ProComputadores.php

14. Szpiniak, A., Rojo, G. Enseñanza de la programación. TE&ET: Revista Iberoamericana. (2006)

15. Wing, J. M. Computational Thinking. Communications of the ACM, 49(3), (2006), pp 33-35.

16. Wing, J. M. Computational Thinking and Thinking about Computing - *Philosophical transactions. Series A,* Mathematical*, physical, and engineering sciences* 366 (1881) (2008) p. 3717-3725.

17. Stimolo, M. I., Caro, N. P. B-Learning: Implementación de Recursos de Internet en la Enseñanza de Estadística en la Facultad de Cs. Económicas - UNC - VI Congreso de Tecnología en Educación y Educación en Tecnología. TE&ET (2011).

18. González Mariño, J. C. B-Learning utilizando software libre, una alternativa viable en Educación Superior. *Revista Complutense de Educación* 17 (1) (2006) p. 121-133.

19. Lenguaje **TIMBA** (*Terribly Imbecile Machine for Boring Algorithms*) – Informe Técnico – UNSL.

20. Diagram Editor: http://dia-installer.de/index.html.es

21. Herramienta PSeInt: http://pseint.sourceforge.net

22. Plataforma Moodle: http://moodle.org

23. Virtual Programming Lab for Moodle: http://vpl.dis.ulpgc.es

Avances en el desarrollo de un Repositorio para Recursos Educativos Abiertos

Silvia Vanesa Torres, María Soledad Zangla, Marcela Cristina Chiarani
UNSL, Ejercito de los Andes 950
San Luis, Capital
{svtorres, szangla, mchiarani}@unsl.edu.ar

Resumen. En este artículo se presenta el avance en el desarrollo de un repositorio de recursos educativos abiertos, creado con el objetivo de compartir experiencias y materiales educativos. En la actualidad uno de los enfoques, ampliamente aceptado, en la aplicación de las TIC en educación se basa en un nuevo paradigma de diseño de materiales de aprendizaje, poniendo énfasis en la reutilización de contenidos. Por consecuencia, surge la necesidad de almacenar, buscar, recuperar, consultar y descargar estos materiales de aprendizaje. Desde el 2009, el Centro de Informática Educativa (CIE), lleva a cabo la producción de Materiales Educativos Digitales (MED), a partir de una perspectiva interdisciplinaria con docentes de diferentes niveles educativos de la provincia de San Luis. El principal propósito es lograr que los materiales sean utilizados en las aulas. Considerando la idea de código abierto y el movimiento de acceso libre al conocimiento, desde el proyecto de investigación, PROICO N°30212 "Herramientas Informáticas Avanzadas para Gestión de Contenido digitales para Educación" de la Facultad de Ciencias Físico Matemáticas y Naturales de la Universidad Nacional de San Luis, se tiene como objetivo investigar, modificar y desarrollar herramientas de software libre que permitan crear MED. Se busca con ello propiciar un marco de colaboración para el desarrollo de recursos educativos, a modo de potenciar la generación y reusabilidad de contenidos digitales.
Palabras Claves: MED, Repositorio, código abierto

Abstract. This paper presents the progress in the development of a repository of open educational resources, created with the aim of sharing experiences and educational materials. At current approach, widely accepted in the application of ICT in education is based on a new design paradigm for learning materials , emphasizing the reuse of content . Consequently , the need to store, search, retrieve , view, and download these learning materials. Since 2009, the Center for Computing Education (ICE) conducts the production of Digital Materials Education (MED) , from an interdisciplinary perspective with teachers from different educational levels in the province of San Luis . A main purpose is to use the materials in classrooms. Taking in mind the idea of open source movement and free access to knowledge , the research project No. 30212 PROICO "Advanced Computer Tools for Managing Digital Content for Education " , Faculty of Physics and Mathematics, and the National University San Luis , the objective is to research, develop and modify free software tools that create MED . With the intention of fostering a collaborative framework for the development of educational resources to enhance the generation mode and reusability of digital contents.
Keywords: MED, Repositorio, open source,

1. Introducción

Uno de los enfoques en la aplicación de las Tecnologías de la Información y la Comunicación (TIC) en educación, ampliamente aceptado, se basa en un nuevo paradigma de diseño de actividades de aprendizaje, poniendo énfasis en la reutilización de contenidos. Como explica Sicilia [4], al principio de la década de los 90 se popularizó la divulgación de contenidos educativos a través de Internet. Desde proyecto PROICO N°30212 se investiga, modifica y desarrolla herramientas de software libre que permitan crear materiales educativos digitales. Con esto se busca favorecer un marco de colaboración para el desarrollo de las actividades académicas en los diferentes ámbitos educativos, con el fin de potenciar la generación y reusabilidad de contenidos digitales.

De lo antes dicho, surge la necesidad de desarrollar un repositorio on line para proveer el acceso a estos MED. Los repositorios se convirtieron en uno de los pilares del e-learning, hasta lograr la proclamación de la ley 26899 Repositorios digitales institucionales de acceso abierto. *"Básicamente se pueden entender como: almacenes digitales en los que se recogen aportaciones individuales de los miembros de una comunidad argentina para ser compartidos y evaluados entre todos ellos*."

1.1 Recursos Educativos Abiertos

En términos sencillos, se puede pensar que un Recurso Educativo Abierto (REA) es un material educativo, en cualquier formato (texto, imagen, audio, video, etc.) que ha sido desarrollado utilizando herramientas de software de uso libre y que su autor publica de forma abierta, es decir brindando las libertades de utilización, modificación y libre distribución; pero la definición formal abarca más que esto, la Fundación Hewlett define los REA como "*recursos destinados a la enseñanza, el aprendizaje y la investigación de dominio público o que han sido liberados bajo un esquema de licenciamiento que protege la propiedad intelectual y permite su uso de forma pública y gratuita o permite la generación de obras derivadas por otros. Los Recursos Educativos Abiertos se identifican como cursos completos, materiales de cursos, módulos, libros, video, exámenes, software y cualquier otra herramienta, materiales o técnicas empleadas para dar soporte al acceso de conocimiento. (Atkins et al., 2007, p. 4)*"

Desde el año 2002 la UNESCO se convirtió en la organización anfitriona de la discusión internacional en torno a esta iniciativa, cuando en el "Foro sobre Impacto de los Cursos Abiertos para Educación Superior en los países en desarrollo" se adoptó la sigla OER (del inglés Open Educational Resouces) y cuya traducción al español fue REA (Recursos Educativos Abiertos). Para ello, la UNESCO con la generosa contribución de la Fundación Flora y William Hewlett, mantiene un foro internacional de discusión con el fin de servir como un laboratorio de ideas, una central de recolección e intercambio de información, un impulsor de estándares y un catalizador de la cooperación internacional.

Teniendo como referencia la Declaración de la UNESCO en el Congreso Mundial sobre Recursos Educativos Abiertos (REA) celebrada en París en 2012, se ha decidido

impulsar la promoción, producción, distribución y uso en Iberoamérica de licencias Creative Commons y REA. Para ello, entre otras acciones, elaborará una propuesta de licencia Creative Commons, específica para su uso en toda Iberoamérica, para la producción y uso de Recursos Educativos Abiertos.

Con respecto a esto, en Argentina se sancionó el proyecto de ley "Creación de Repositorios Digitales Abiertos de Ciencia y Tecnología". El Sistema Nacional de Repositorios Digitales (SNRD) es una iniciativa del Ministerio de Ciencia, Tecnología e Innovación Productiva conjuntamente con el Consejo Interinstitucional de Ciencia y Tecnología (CICyT) a través de sus representantes en el Consejo Asesor de la Biblioteca Electrónica de Ciencia y Tecnología.

El SNRD tiene como propósito conformar una red interoperable de repositorios digitales en ciencia y tecnología, a partir del establecimiento de políticas, estándares y protocolos comunes a todos los integrantes del Sistema. A través de su sitio web es posible acceder a las publicaciones científico-técnicas depositadas en los Repositorios argentinos adheridos al Sistema Nacional de Repositorios Digitales.

Los objetos digitales disponibles, pueden ser accedidos en forma gratuita, leídos, descargados, copiados, distribuidos, impresos, buscados o enlazados y utilizados con propósitos legítimos ligados a la investigación científica, a la educación o a la gestión de políticas públicas, sin otras barreras económicas, legales o técnicas que las que suponga Internet en sí misma. La única condición, para la reproducción y distribución de las obras, es la obligación de otorgar a los autores el control sobre la integridad de su trabajo y el derecho a ser adecuadamente reconocidos y citados.

Los MED desarrollado por nosotros cumplen los requisitos de ser recursos destinados para la enseñanza, el aprendizaje y la investigación, están liberados bajo un esquema de licenciamiento que protege la propiedad intelectual y permite su uso de forma pública, gratuita y permite la generación de obras derivadas por otros. Por lo tanto, los MED se transforman en REA y el repositorio es lo que alojará finalmente.

El presente trabajo está organizado de la siguiente manera: comienza con una introducción al tema, seguida de la conceptualización de software libre. Continúa con repositorios de código libre. Luego, se detalla el avance en el modelo de repositorio que desarrolla nuestro proyecto. Finalizamos este documento con la conclusión del mismo.

2. ¿Porque elegir software libre?

El software libre proporciona la independencia a los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Dentro de la corriente de software denominada "Software Libre" se encuentran varias aplicaciones para instalar un repositorio.

Nuestro proyecto de investigación se encuadra dentro de las premisas del software libre como lo expresa en su libro digital Richard M. Stallman:

El adjetivo «libre» en el software libre hace referencia a la libertad del usuario para ejecutar, modificar y redistribuir software. Es por ello que creemos que los usuarios de

ordenadores deberían tener libertad para cambiar y redistribuir el software que utilizan. Esto contribuye al saber humano, al contrario que el software propietario. Por este motivo, desde esta investigación se pretende fomentar el software libre. Tal como lo promueve la ley 26.899 que fomenta a científicos y académicos a publicar sus obras. [1]

El mismo autor aporta una definición, difundida ampliamente en Internet, que aclara cual es la libertad que tienen los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Allí también, menciona las cuatro libertades para los usuarios del software:

De acuerdo con tal definición, el software es "libre" si garantiza las siguientes libertades:

Libertad 0: ejecutar el programa con cualquier propósito

Libertad 1: estudiar y modificar el programa

Libertad 2: copiar el programa de manera que se pueda ayudar al vecino o a cualquiera.

Libertad 3: mejorar el programa, y hacer públicas las mejoras.

Como se puede ver, las libertades 1 y 3 obligan a que se tenga acceso al código fuente. La libertad 2 hace referencia a la libertad de modificar y redistribuir el software libremente licenciado bajo algún tipo de licencia de software libre que beneficie a la comunidad.

Hay que aclarar que un determinado programa sea de Software libre no implica en ningún momento que este sea o deba ser gratuito (freeware). Es perfectamente compatible el que se trate de un software libre y se cobre por servicios brindados. En cuanto a la 2ª premisa (la libertad de distribuir copias, con lo que puede ayudar a otros), está siempre está supeditada a los acuerdos de licencia de dicho programa (aunque se trate de programas en régimen de freeware).

Para las instituciones educativas públicas es indiscutible la conveniencia de trabajar con material de estas características.

3. Repositorios de código abierto

Las motivaciones que llevan el uso y desarrollo del software de código abierto son diversas, van desde razones éticas y filosóficas a cuestiones prácticas. A continuación se detalla algunas de las ventajas prácticas.

Una de las principales ventajas de los modelos de código abierto es el hecho de que el software de código abierto está disponible gratuitamente a un bajo costo, si bien esta característica no es exclusiva de software de fuente abierta, y varios productos de software propietario están a disposición en forma similar. Lo que realmente distingue el software de código abierto del software disponible sin costo es la combinación de los efectos que causan las características, tales como: el acceso al código fuente, el derecho a modificar el mismo, entre otras. Todas ellas en conjunto producen un efecto sinérgico que es la causa de las verdaderas ventajas del modelo de código abierto.

La disponibilidad del código fuente y el derecho a modificar permite el ajuste ilimitado y la mejora de un producto de software. También hace posible la adaptación del código a un nuevo hardware, para adaptarlo a las condiciones cambiantes, y para llegar a una

comprensión detallada de cómo funciona el sistema. Esta es la razón por la que muchos expertos están llegando a la conclusión de que en realidad para extender la vida útil de una aplicación, debe estar disponible el código fuente. La disponibilidad del código fuente hace que sea mucho más fácil de aislar errores y solucionarlos.

El número de comunidades virtuales que han surgido relativas al desarrollo de repositorios y REA demuestran un gran interés en el ámbito académico. Los siguientes proyectos se encuentran trabajando con la perspectiva del Software libre: Aloha II[10], Pool[11], Eduplone[12], eRib[6], Planet[8], DOOR[5]. Entre los repositorios desarrollados de primera generación se encontró Careo [12], uno de los más conocidos que estuvo disponible de este modo.

Luego de la evaluación realizada, por el proyecto de investigación, sobre Repositorios existentes de código abierto [15] se procuró seleccionar uno que se adecuara a nuestras necesidades, por tal motivo, se optó por DOOR. La razón fundamental de esta selección es que está desarrollado en PHP y MySql, compatible con la plataforma utilizada en el proyecto para sus producciones. Además, consideramos importante que DOOR está desarrollado siguiendo el estándar internacional IMS Metadata 1.2.1 y Content Package 1.1.3. En base a las modificaciones realizadas sobre el repositorio DOOR, se dio origen al Repositorio de Objetos de Informática ROI. En este último se alojarán todos los Recurso Educativos Abiertos.

4. Modificaciones al Repositorio

Readecuar el repositorio DOOR para que aloje los REA, tiene como objeto que puedan ser accedidos en forma gratuita, buscados, visualizados, descargados, copiados, distribuidos, impresos, o enlazados y utilizados. Con propósitos ligados a la investigación científica, a la educación, sin barreras económicas y legales. La única condición, para la reproducción y distribución de las obras, es la obligación de otorgar a los autores el control sobre la integridad de su trabajo y el derecho a ser adecuadamente reconocidos y citados.

A continuación se detallan los cambios que realizamos al repositorio, como así también la optimización de los servicios que ofrece (buscar, cargar, descargar, etc. recursos educativos abiertos). En DOOR se pude acceder con diferentes roles, tales como: invitado (lector), administrador o usuario a cargar un REA (autor). Las modificaciones son:

1) Se llevaron a cabo modificaciones visuales para readecuarla al proyecto de investigación al cual pertenece.

2) En DOOR es posible cargar un recurso (un solo archivo a la vez), se realizaron las modificaciones necesarias para poder agregar un REA al repositorio. Por lo que los recursos de aprendizaje se pueden agregar en forma de paquetes que incluyen la descripción del objeto y el contenido o bien incluir solamente los metadatos junto con la url del objeto.

3) Se generó un árbol con categorías, para que los REA que van siendo incorporados al repositorio estén organizador a través de ellas. Esta categorización

puede ser modificada por aquel usuario que posea el rol de administrador. Se observa en la figura 1.

4) A cada nuevo recurso que se incorpore al repositorio, se le podrá agregar el tipo de licencia "creative common" que el autor requiera. Además, cuáles serán los destinatarios del recurso, según el autor. Se realizaron modificaciones en la base de datos para que se visualice el tipo de licencia.

Figura 1: Visualización de categorías

En la pantalla principal del repositorio se puede observar que el acceso se realiza indicando usuario y contraseña (figura 2). Si el usuario no se ha registrado puede solicitar al administrador una cuenta para acceder. Los tipos de usuarios que se pueden asignar son los siguientes: lector, autor y administrador. Además, se generó un nuevo usuario denominado "invitado" el cual puede acceder sin que sea necesario ingresar clave y usuario.

En la figura 2 se observa que el repositorio actualmente se encuentra on-line, alojado en el servidor del proyecto. Está a disposición de docentes e investigadores que quieran alojar sus REA.

Figura 2: Servidor principal que contiene on line el Repositorio CIE

Evidentemente la creciente utilización de repositorios educativos nos lleva a observar, que en los últimos años la práctica docente ha experimentado un vertiginoso progreso en el uso de los recursos de apoyo. Es así como se ha podido observar una inclusión de herramientas sustentadas en las TIC. En consecuencia, en la actualidad podemos observar que en muchas instituciones educativas del material impreso comienzan abordar al material digitalizado, de la consulta en libros a la navegación en Internet, entre otros. Los recursos educativos, producto del avance tecnológico, están ahora al alcance de los docentes y de los alumnos. La utilización de estos recursos en el aula constituye una herramienta fundamental para el desarrollo y enriquecimiento del proceso de enseñanza-aprendizaje del alumno. A partir de estas últimas fortalezas de la inclusión de recursos es de vital importancia que toda la comunidad educativa pueda acceder a los recursos educativos abiertos, a través del repositorio, que son construidos interdisciplinariamente entre docentes de diferentes instituciones educativas.

5. Conclusiones y tareas futuras

Todo el trabajo que se viene realizando permitió que actualmente se encuentre on - line la primera versión del repositorio, a cual se accede solicitando un usuario y clave. A futuro, una vez que los usuarios estén más concientizados en el uso de estos repositorios será de acceso abierto para que los usuarios puedan darse de alta. El rol seguirá siendo asignado por el administrador.

Además, se está enfocando las modificaciones del repositorio en la adecuación de la

meta etiquetado de los REA al estándar Dublin Core [17], debido a que los repositorios más utilizados a nivel mundial utilizan estos parámetros de clasificación.

Otra propuesta para avanzar, es lograr la evaluación de pares de trabajos alojados en el repositorio, esto daría jerarquía a los mismos. Para ello es necesario que existan evaluadores que trabajaran en las distintas categorías que tienen el árbol de clasificación y un coordinador que confirmara las evaluaciones.

Nuestro equipo, seguirá trabajando en busca de compartir experiencias y conocimientos relacionados con nuestro tema central de investigación para posibilitar espacios de conocimiento colectivos.

Referencias

1. Richard M. Stallman. El Movimiento Del Software Libre. http://www.fsf.org
2. Chiarani M., Leguizamon G., Pianucci I. (2006). Repositorio de Objetos de Aprendizaje para Carreras Informáticas. WICC 2006, Moron.
3. Software libre. Free software foundation. http://www.fsf.org/
4. Sicilia Urbán Miguel-Angel, Sánchez Alonso Salvador. (2006) Repositorios de objetos de aprendizaje. Information Engineering Research Unit. Universidad de Alcalá.
5. DOOR. http://door.elearninglab.org/website/index_ita.php
6. eRIB. Repositorio Edusource. http://edusource.licef.teluq.uquebec.ca/ese/fr/install_erib.htm
7. IEEE Learning Technology Standards Committee (2002) Learning Object Metadata (LOM), IEEE 1484.12.1-2002. http://www.um.es/ead/red/M2/
8. Planet. http://ants.etse.urv.es/planetdr
9. Repositorio institucional E-Print. http://www.eprints.org/
10. Aloha II. http://aloha.netera.ca
11. Pool. Portal para Edusplash. http://edusplash.net
12. Eduplone. http://www.eduplone.net
13. Repositorio digital DSpace. http://www.dspace.org/
14. Ponce Viviana, Chiarani Marcela, Pianucci Irma. ROI: Repositorios de Objetos de Aprendizaje Informáticos. CACIC 2007. Corrientes. ISBN 950-656-109-3.
15. Chiarani M., Pianucci I., Leguizamon G. (2006). Repositorio de Objetos de Aprendizaje para Carreras Informáticas. Publicado en el WICC. Morón Disponible en CD
16. Dublin Core. http://www.dublincore.org/projects/
17. Atkins D. , Brown J. , Hammond A. (2007). A Review of the Open Educational Resources (OER) Movement: Achievements, Challenges, and New Opportunities consultado el 6 de junio de 2013, disponible en: http://www.hewlett.org/uploads/files/ReviewoftheOERMovement.pdf

Relatos, Mapas Conceptuales y Arquitectura de Computadores

Julio Ortega, Mancia Anguita
Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada
{jortega, manguita}@ugr.es

Resumen. Este trabajo parte de la hipótesis de que la historia de una disciplina y, más concretamente, los relatos de episodios de su desarrollo contribuyen a conformar los mapas conceptuales que permiten fijar los conceptos esenciales, y a motivar a los estudiantes en el estudio de dicha disciplina. Aquí se proponen algunos episodios relacionados con la evolución de los computadores para incluir en un curso de Arquitectura de Computadores, junto con referencias que permitan acceder a información relativa a los mismos.

Palabras clave: Arquitectura de Computadores, Historia de los computadores, Mapas conceptuales, Motivación

Abstract. This paper stems from the hypothesis that the history of the development of a discipline and, more specifically, the storytelling of episodes of that development contribute to build conceptual maps that make it easy a persistent memorization of the essential concepts along with the motivation of the students towards these concepts. Here, we also propose a set of histories related with the computer evolution to be included in a Computer Architecture course.

Keywords: Computer architecture, Conceptual maps, History of computers, Motivation

1 Introducción

La inclusión de relatos acerca de la forma en que se han planteado problemas esenciales en el desarrollo de una disciplina, y de cómo han ido proponiéndose soluciones, puede incrementar la motivación para el estudio de dicha disciplina. En [1] se analiza la utilidad de la historia de la computación como herramienta pedagógica y la necesidad de incluir asignaturas de este tipo en los planes de estudios. La historia contribuye a que los estudiantes se hagan preguntas fuera de lo que es la dinámica de razonamiento propio de la asignatura y la contemplen dentro del contexto

del desarrollo global de la informática descubriendo relaciones con otras asignaturas de su currículo. Se contribuye así a construir mapas que interrelacionan elementos y facilitan su retención y a ampliar su comprensión del contexto y su pensamiento crítico. La historia puede mejorar la comprensión de las tendencias del desarrollo del hardware y el software al relacionar los computadores con su dimensión humana.

Otra de las razones más aparentes para conocer la historia del desarrollo de una tecnología es la de tener en cuenta los errores cometidos para no repetirlos, y fomentar la capacidad de construir a partir de las aportaciones que han tenido éxito en el pasado. También se puede poner de manifiesto la importancia de disponer de una tecnología suficientemente madura para poder plasmar ideas avanzadas, y la de reflejar las estrategias sociales que han dado lugar a nuevos avances como el papel de los pioneros y emprendedores, el de la cooperación y el trabajo en grupo, etc. En muchos casos, no se tiene una información completa de las respuestas que las empresas o las instituciones han dado a problemas o retos que se han ido planteando, pudiéndose fomentar en este caso la discusión en clase y la curiosidad.

La referencia a los protagonistas más relevantes en el desarrollo de los elementos que se han constituido en los conceptos propios de la disciplina contribuye a definir una comunidad con la que el estudiante puede empezar a identificarse. En esta red social va a encontrar personalidades que pueden definir modelos y trayectorias para su actividad como ingeniero. El acercamiento histórico a la disciplina permite interrelacionar conceptos técnicos y su incidencia social contribuyendo a ubicar el papel del ingeniero dentro de su entorno. Surge así una faceta de motivación digna de consideración, que puede contribuir de manera relevante a mejorar el acercamiento del estudiante a la disciplina en cuestión.

Partiendo de la utilidad de incluir referencias históricas en las asignaturas, no obstante es importante tener en cuenta la forma en que esto se lleva a cabo. Presentar una serie de hechos distribuidos en el tiempo sin más, sería contemplada por los estudiantes como un ejercicio más de memoria que tendría un efecto negativo. A este respecto consideramos acertado el uso del relato o la narración histórica, tal y como se propone en [2], donde se hace referencia a la importancia de esta estrategia dado que los relatos son intrínsecamente interesantes dada nuestra mayor disposición a recordar narraciones más que argumentos lógicos. La idea es aprovechar la tendencia a recordar historias para que los estudiantes también recuerden la información técnica relacionada con la disciplina involucrada en el relato.

Considerando las claves que proporcionan algunas de las teorías de la motivación más relevantes propuestas hasta el momento, se proponen algunas estrategias docentes basadas en el uso de relatos de la evolución de los computadores para motivar el estudio de las asignaturas de tecnología, estructura y arquitectura de computadores, en particular la asignatura de Arquitectura de Computadores, y mejorar la asimilación de los correspondientes mapas conceptuales. Si los estudiantes consideran los contenidos de estas asignaturas asequibles y útiles para su formación como Ingenieros Informáticos es más probable que la Ingeniería de Computadores sea contemplada por algunos como su profesión.

En este artículo se considera una asignatura de introducción a la Arquitectura de Computadores. En la Sección 2 se proporcionan los detalles relativos a los contenidos y competencias de la asignatura que se imparte en el Grado de Ingeniería Informática de la Universidad de Granada, así como un mapa conceptual para la misma, y en la

Sección 3 se considera el modelo de aprendizaje y la importancia de la motivación, junto con la utilidad de los relatos históricos. La Sección 4 presenta, a partir de referencias a una serie de artículos, relatos de la evolución de los computadores y se muestra su relación con los conceptos de la asignatura. Finalmente, la Sección 5 incluye las conclusiones del artículo y la Sección 6 proporciona las referencias.

2 Arquitectura de Computadores y sus mapas conceptuales

En [3] se describen los contenidos y las estrategias docentes de una asignatura de introducción a la Arquitectura de Computadores. Concretamente, se trata de una asignatura de grado que se imparte en el Grado en Ingeniería Informática de la Universidad de Granada. La Figura 1 muestra un mapa conceptual que relaciona los principales conceptos y contenidos que se abordan en dicha asignatura. Para construir dicho mapa se puede partir de uno de los modelos más frecuentemente utilizados para evaluar las prestaciones de un procesador (por ejemplo en [4]). Se trata de un modelo sencillo, y con un nivel de detalle suficiente para nuestro propósito:

$$T_{tarea} = NIxCPIxT_{ciclo} = NIx\left(\frac{CPI}{f}\right)$$

donde T_{tarea} es el tiempo que tarda en ejecutarse un programa, NI es el número de instrucciones máquina que lo componen, CPI es el número medio de ciclos por instrucción, y T_{ciclo} el periodo de reloj del procesador (inverso de la frecuencia, f). Así, el estudio de la arquitectura del computador debe referirse a todos aquellos aspectos que determinan los valores de NI, CPI, y f, y su interrelación. Así, NI depende del repertorio de instrucciones y del compilador; CPI viene determinado por el repertorio de instrucciones y la organización del computador; y f depende de las prestaciones que proporcione la tecnología y de la organización del computador. Como se ha indicado, la organización del computador afecta tanto al valor de CPI como a f, ocasionando que realmente la métrica importante sea el cociente CPI/f. Así, por ejemplo, se puede reducir el valor de CPI utilizando frecuencias menores.

El estudio de la organización del computador implica considerar las características de los distintos subsistemas que lo integran (procesador, memoria, dispositivos periféricos y de almacenamiento), tanto desde el nivel de arquitectura como desde el de microarquitectura; así como la interconexión de dichos subsistemas (jerarquía de buses), y los flujos mutuos de datos y control (comunicación y sincronización) que permiten el funcionamiento del computador. Por tanto, desde una perspectiva de ingeniero, el estudio de la Arquitectura del Computador debe incluir las interacciones y compromisos entre los distintos componentes del computador para conseguir el nivel de prestaciones requerido. Esto nos lleva a estudiar las distintas alternativas de mejora de prestaciones, teniendo en cuenta la relación entre las características de los elementos de una arquitectura y el nivel de prestaciones alcanzado, a través de distintos modelos.

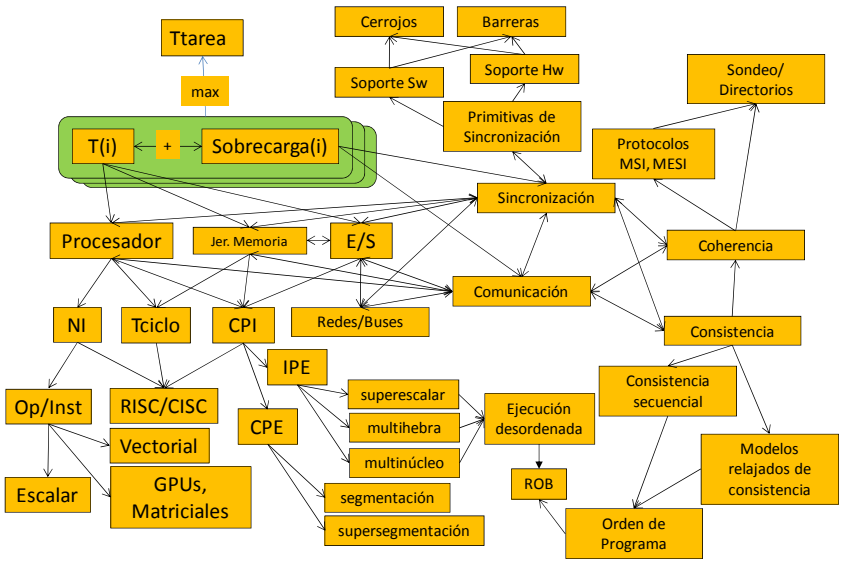

Figura 1. Un mapa conceptual para una asignatura de introducción a la Arquitectura de Computadores

La arquitectura traduce a capacidades reales y prestaciones las posibilidades que ofrece la tecnología de integración y fabricación de circuitos integrados. La forma en que esto se ha llevado a cabo a lo largo de la evolución de los computadores (tanto en el pasado como en un futuro previsible) se puede resumir en paralelismo y localidad. El paralelismo, aplicado a distintos niveles, permite reducir el tiempo por la realización simultánea de operaciones, y la localidad de los datos permite reducir las latencias de acceso a los mismos por parte del procesador, que evita accesos a niveles de la jerarquía de memoria más profundos y lentos. No obstante, tanto para implementar el paralelismo como para mejorar la localidad hay que utilizar recursos, sobre cuya distribución debe decidir el diseñador, en base a la comprensión de los efectos posibles en las prestaciones del sistema. Esta interesante interacción paralelismo-localidad se da en el computador a todas las escalas, tiene efectos en los elementos y problemas que se plantean (por ejemplo la coherencia de cache, o la necesidad de estructuras de comunicación entre procesadores, procesador y memoria, etc.), y permite exponer los conceptos esenciales al diseñar/evaluar/utilizar un computador. Por lo tanto, tienen un papel central en el estudio de la arquitectura del computador.

El paralelismo se puede aprovechar de formas diferentes (paralelismo funcional y paralelismo de datos) y a niveles muy distintos (paralelismo entre instrucciones, paralelismo entre hebras, entre procesos, etc.), y su estudio, junto con los problemas que plantea su aprovechamiento y las alternativas para resolverlos debe incluirse dentro de los contenidos de la materia. El paralelismo entre instrucciones (ILP) se ha aprovechado a través de procesadores segmentados en los que se han introducido mejoras como la posibilidad de enviar a ejecutar varias instrucciones al mismo tiempo, la ejecución desordenada, el procesamiento especulativo, etc. Se puede

reflejar esta situación en la expresión del tiempo de ejecución, si se introduce en ella el número medio de ciclos entre emisiones de instrucciones (CPE, Ciclos por cada Emisión), y el número medio de instrucciones que se emiten cada vez (IPE). Así, el valor de CPI sería igual a CPE/IPE. En un procesador segmentado (no superescalar), CPE=IPE=1, y CPI=1, en uno superescalar debería ocurrir que CPE=1 y IPE>1 (cada ciclo se emiten varias instrucciones), y en un procesador no segmentado se tendría que IPE=1 y CPI=CPE>1 (se emite una instrucción cada ciclo máquina constituido por varios ciclos de reloj).

$$T_{tarea} = NIx\left(\frac{CPE}{IPE}\right) xT_{ciclo} = NIx\left(\frac{CPE}{IPE}\right) * \left(\frac{1}{f}\right)$$

En los repertorios de instrucciones de tipo vectorial o SIMD, cada instrucción codifica varias operaciones. Por lo tanto, en lugar de tener que considerar NI como en una arquitectura escalar, donde al codificar cada instrucción una operación se tendría que NI = Noper (Noper es el número de operaciones a realizar en el programa), el número de instrucciones ejecutadas será Noper/Op_instr y se tendrá:

$$T_{tarea} = \left(\frac{Noper}{Op\_instr}\right) xCPIxT_{ciclo}$$

En el caso de un procesador vectorial Op_instr > 1 y el número de instrucciones es menor que el de operaciones, en un repertorio escalara, Op_instr =1 y, como se dijo NI=Noper.

Si se utilizan varios procesadores en el computador, y se distribuye el trabajo entre ellos, la expresión que se utilizó al comienzo de esta sección para el tiempo de ejecución de un programa debe modificarse. Ahora, al dividir el trabajo entre P procesadores, se tendría:

$$T_{tarea} = \max_{i=1,..,P}\left\{T_{tarea(i)} + O_i\right\}$$

donde $T_{tarea(i)}$ es el tiempo correspondiente a la ejecución de la parte de tarea asignada el procesador i-ésimo, y O_i es la sobrecarga (overhead) ocasionada básicamente, por los tiempos de comunicación entre los procesadores y los tiempos de espera para la sincronización. Las causas de esta sobrecarga (redes de conexión, modelos y costo de las comunicaciones, los problemas de coherencia, etc.), y las alternativas para reducir sus efectos deben incorporarse a los contenidos de la materia. Así pues, habría un segundo grupo de contenidos relacionados con el incremento de prestaciones de las arquitecturas.

3 Los relatos históricos y la motivación

En [5] se revisan distintas teorías de la motivación para justificar el diseño de estrategias docentes que incrementen el interés y el rendimiento de los estudiantes en una materia. En la Tabla 1 se resumen algunas de las estrategias que se discuten en

[5], relacionándolas con las distintas teorías de la motivación analizadas en dicho trabajo.

Tabla 1. Estrategias docentes según distintas teorías de la motivación

Teoría de la motivación	Estrategias docentes
Objetivos (ser competente, alcanzar los objetivos)	• Despertar la curiosidad planteando preguntas sobre aplicaciones prácticas de la materia que se irán respondiendo a lo largo de la asignatura • Evitar referencias que puedan convertir a las calificaciones en objetivos de aprendizaje
Atribuciones (causas internas, dinámicas, y controlables)	• Proponer relaciones de problemas de dificultad creciente, comenzando con problemas de resolución muy directa a partir de las leyes o expresiones matemáticas estudiadas en la asignatura • Evaluar el esfuerzo: proponer varias actividades voluntarias y tener en cuenta el número de actividades completadas.
Expectativa-Valor (metas alcanzables y relevantes)	• Mostrar que los conceptos que se abordan en la asignatura son importantes para ser profesionales competentes. • Realizar pruebas de evaluación sobre habilidades concretas o contenidos específicos tratados en clase con detalle.
Autodeterminación (autonomía, competencia, socialización)	• Plantear prácticas y ejercicios individuales que pongan de manifiesto la utilidad de los contenidos de la asignatura. • Promover el trabajo cooperativo en equipo • Describir posibilidades profesionales que abren las competencias alcanzadas con los contenidos de la asignatura

Así, desde el punto de vista de la motivación, los objetivos de las asignaturas (1) deben ser relevantes para la profesión, y (2) deben contemplarse como asequibles a través de un esfuerzo razonable. No es "bajar el listón" (ver los 10 mandamientos de la docencia según el profesor Yale Patt [6]), sino buscar que los estudiantes asimilen y sientan que controlan el aprendizaje de la asignatura, y sean capaces de abordar conceptos o problemas complejos que, en muchos casos, son los que les hacen ser conscientes de la utilidad de la asignatura para su futuro profesional, los que le convencen de que el esfuerzo realizado les han permitido avanzar en su conocimiento del computador y de aspectos de su profesión que no hubieran ni imaginado si no es por la asignatura estudiada.

Según un modelo coevolutivo de las transformaciones tecnológicas aplicado al ámbito del computador digital [7], en las innovaciones y los logros de los ingenieros y científicos (von Neumann, Eckert y Mauchly, Wilkes, Amdahl, Cray, etc.) y de las empresas (IBM, Burroughs, CDC, DEC, Cray, etc.) están enraizadas formas de trabajo y tecnologías previas, y son parte de un proceso evolutivo de cambio social y tecnológico que no está aislado de su contexto histórico. Lo nuevo no aparece aleatoriamente sino que se crea en organizaciones y sistemas sociales a partir de las capacidades tecnológicas disponibles, las ideas sobre la necesidad de ciertos productos, y las posibilidades de fabricación y uso de los mismos. Un ingeniero no explora todas las posibles soluciones a un problema sino que enfoca su búsqueda a partir de sus nociones acerca de los problemas relevantes y de las formas plausibles de abordarlos según lo que se denomina el régimen o paradigma tecnológico dominante, definido a partir del conocimiento científico existente, el trabajo práctico

del ingeniero, las tecnologías de producción, las características de los productos, y las costumbres, habilidades, e infraestructuras de los usuarios y de las instituciones. Los ingenieros desarrollan su actividad a través de proyectos que tendrán como resultado productos o procesos que competirán con otras alternativas en el momento futuro en que se comercialicen. Según esto, todo ingeniero debe tener presente un modelo de la dinámica de su campo, y para construirlo es importante conocer qué factores han determinado los cambios en el pasado.

Tabla 2. Referencias utilizadas para construir relatos históricos y su relación con los contenidos abordados en la asignatura Arquitectura de Computadores

Contenidos	Referencias para los relatos
Prestaciones del procesador	"A Tale of two processors: revisiting the RISC-CISC debate" C. Isen, L. John, E. John, 2009 [8] "RISC: Back to the future?" C. G. Bell [9]
Clasificación de las arquitecturas	"A Historical overview of Computer Architecture" R. E. Smith [10] "Creating the computer industry" E.W. Pugh; W. Aspray, 1996 [11]
Multicomputadores y supercomputadores	"The social limits of Speed: The development and use of supercomputers" B. Elzen; D. Mackenzie, 1994 [12]
Ley de Amdahl	"Computer Architecture and Amdahl's Law" G.M. Amdahl, 2013 [13]
Multiprocesadores y multicomputadores	"Ultracomputers a Teraflops before its time" G. Bell [14]
Paralelismo entre instrucciones	"The Pentium Chronicles: Introduction". R.P. Colwell, 2006 [15]

4 Algunos episodios relevantes en Arquitectura de Computadores

En esta sección se proponen varios artículos que pueden servir de ayuda para conformar mapas conceptuales en los temas de un programa de introducción a la Arquitectura de Computadores. Como se ha indicado en la Sección 2, una asignatura de este tipo comprende temas dedicados a la evaluación de prestaciones, la clasificación de las arquitecturas y la introducción a la programación paralela. En cuanto a las arquitecturas paralelas, se presentan los conceptos fundamentales del paralelismo ILP y el procesamiento multihebra: planificación de instrucciones, comunicación y sincronización, coherencia y consistencia de memoria, etc. En la Tabla 2 se recogen las referencias que pueden utilizarse para construir relatos relacionados con los conceptos fundamentales que se abordan. Se incluyen artículos con información acerca de la controversia RISC/CISC [8,9], y del momento en que una arquitectura CISC se implementó de forma eficiente a través de una microarquitectura superescalar (microarquitecturas P6 de Intel) [15]. También hay

artículos que describen cómo van surgiendo distintas arquitecturas a partir de planteamientos que priman la compatibilidad con diseños previos o la obtención de mejores prestaciones a cualquier precio. Así, se considera el nacimiento y desarrollo de las arquitecturas vectoriales [12] y de los multiprocesadores [14], y la controversia relativa al uso de varios procesadores o mejora de la capacidad del procesador que se produce a partir de la ley de Amdahl [13]. Textos frecuentemente utilizados como el libro de Arquitectura de Hennessy y Patterson [4] incluyen secciones que revisan el desarrollo histórico de los conceptos y las implementaciones que se abordan en sus distintos temas. Esas fuentes proporcionan información de la sucesión en el tiempo de las distintas arquitecturas y sus principales características. Lo que aquí se propone son una serie de referencias para construir relatos sobre el desarrollo ciertos episodios clave en el avance de las arquitecturas de computador incluyendo algunas anécdotas e información acerca del entorno en el que se van sucediendo los principales conceptos estudiados en la asignatura.

A continuación se resumen parte de los contenidos de los artículos que se proponen, y se indican algunas de las razones por las que resultan interesantes para la asignatura y pueden servir para motivar a los estudiantes.

A historical overview of Computer Architecture (R.E. Smith) [10]. Este extraordinario artículo presenta una historia de la arquitectura de computadores que incluye los orígenes de la disciplina. Se describen con cierto detalle los episodios correspondientes al desarrollo de los primeros computadores y se analiza cómo van surgiendo los diferentes tipos de máquinas, desde las que buscan mejorar las prestaciones ante todo (los supercomputadores) hasta los que introducen el coste como un elemento importante a la hora de evaluar el computador (los minicomputadores y los microcomputadores). No se limita a describir detalles técnicos de las máquinas, sino que se relata el proceso según el que se fueron introduciendo distintos elementos clave de la arquitectura, desde las mejoras en los circuitos aritméticos, hasta la incorporación de distintas tecnologías de memoria, la necesidad de lenguajes de programación de alto nivel, y la llegada de los microprocesadores, entre otros. Este trabajo puede utilizarse tanto en los temas de introducción en los que se presenta el concepto de arquitectura y se describen los distintos tipos de máquinas, como para poner ejemplos del proceso de diseño y fabricación de computadores de cada uno de ellos.

The social limits of Speed: The development and use of supercomputers (B. Elzen; D. Mackenzie) [12]. Proporciona detalles del desarrollo de los primeros supercomputadores comparando dos tendencias, representadas inicialmente por CDC y por IBM. Por un lado está la búsqueda de las máquinas más rápidas en cada momento, y por otro la necesidad de mantener la compatibilidad con diseños previos de la compañía. La primera opción es la defendida por Seymour Cray, uno de los ingenieros más populares y relevantes en el desarrollo de arquitecturas de computador innovadoras, que inició su carrera en CDC. La información que se proporciona en este artículo permite construir relatos acerca del desarrollo de máquinas como la CDC6600 y la CDC7600 y contraponer su filosofía de diseño a la seguida por IBM con sus System 360. El CDC6600 debía ser de 15 a 20 veces más rápido que su predecesor el CDC1604, y se partió de un diseño totalmente novedoso que buscaba la mejora deseada en dos fuentes: (1) el uso de circuitos fabricados por Fairchild

Semiconductor que eran cinco veces más rápidos que los utilizados en el CDC1604, y (2) el diseño de una arquitectura de computador novedosa con varias unidades funcionales que podían funcionar en paralelo y que sería responsable de un factor de mejora de tres o cuatro. El nuevo CDC6600 apareció en Octubre de 1964 y fue capaz de atraer gran parte del mercado antes dominado por los IBM7090 e IBM7094. Una alternativa de diseño diferente la representa el IBM System 360, que también apareció en 1964. Este computador utilizaba un repertorio de instrucciones más complejo que buscaba satisfacer los requisitos de aplicaciones tanto de carácter científico, como los de otras propias del ámbito comercial donde, por ejemplo, era necesario manejar grandes cantidades de datos sin tener que realizar operaciones matemáticas costosas sobre ellos. Con estas máquinas IBM introdujo la idea de separar las características de la arquitectura de la máquina que determinaban el nivel de prestaciones y el repertorio de instrucciones, que mantenía la compatibilidad entre máquinas diferentes. IBM fue capaz de doblar las prestaciones del CDC6600 en 1967, con el IBM System360/91, pero dos años después el CDC7600 dobló a su vez las del computador de IBM (el ciclo de reloj había pasado de 100 ns a 27.6 ns). Por tanto, los relatos del desarrollo de los CDC6600 y CDC7600 y del IBM System360 permiten poner de manifiesto, la existencia de distintas filosofías de diseño, las orientadas a mejorar la velocidad a cualquier precio frente a las que pretenden mantener la compatibilidad con los diseños previos. En ambos casos hay que tener en cuenta las capacidades que ofrece la tecnología disponible, y el uso que la arquitectura puede hacer de ella, con o sin cambios en el repertorio de instrucciones (según el mantenimiento de la compatibilidad sea o no un objetivo).

Otro relato importante que se puede encontrar en [12] es el del diseño del Cray-1 y los principales supercomputadores vectoriales que le siguieron. En 1972, S. Cray fundó Cray Research Inc. a partir de 500.000 dólares aportados por el propio Cray, otros 500000 aportados por CDC, y 1.500.000 aportados por otros 14 inversores (entre ellos, algunos amigos de S. Cray). Es importante resaltar que en el nombre de la empresa aparece la palabra “investigación”, poniendo de manifiesto su importancia en el desarrollo de máquinas cada vez más potentes.

Computer Architecture and Amdahl’s Law (G.M. Amdahl) [13]. El número de diciembre de 2013 de la revista IEEE Computer Magazine tiene como tema central los orígenes, el impacto, y la situación actual de las leyes de la computación. Concretamente se incluyen artículos relacionados con las leyes de Moore, Amdahl, Metcalfe, Grosch, y Makimoto. El artículo correspondiente a la ley de Amdahl esta escrito precisamente el autor de dicha ley, Gene M. Amdahl. Se trata de un trabajo muy útil e interesante porque ofrece una descripción de primera mano del sentido de dicha ley, y una información difícil de encontrar acerca de la evolución del trabajo de un ingeniero en el desarrollo de los primeros computadores. Tras graduarse en física en 1948, después del periodo correspondiente a la Segunda Guerra Mundial en el que había enseñado electrónica en la Marina, empezó investigando en física nuclear y a partir de ahí, se plantea la posibilidad de mejorar la forma en que se realizaban los cálculos necesarios para resolver los problemas que se planteaban en su línea de trabajo. Amdahl relata cómo va implicándose cada vez más en problemas relacionados con el diseño de computadores. En 1952 lee su tesis doctoral titulada “The Logical Design o an Intermediate Speed Digital Computer”, que tuvo que ser

leída en otra Universidad distinta dado que en la suya no había nadie capaz de evaluarla, y dio lugar al computador WISC (Wisconsin Integrally Synchronized Computer), construido en 1955.

Después participa en el diseño del IBM 709 (más concretamente en el diseño de los canales de E/S) aunque inicialmente su tarea en IBM era la simulación de redes neuronales artificiales y su aplicación al reconocimiento de caracteres. El artículo describe bastantes detalles acerca de su etapa como ingeniero en IBM, en particular su participación en el diseño de la familia IBM System 360 y de cómo tuvo acceso a bastantes datos acerca del uso de computadores de IBM, desde el 704 al 7094, y llegó a ciertas conclusiones respecto a la velocidad de cómputo que se podría conseguir en función del tamaño de la memoria en esas máquinas, y de una relación que establecía que por cada instrucción ejecutada se necesitaba 1 bit de E/S. Una de las secciones del artículo se dedica, como no puede ser de otra forma, a la ley de Amdahl que, según cuenta el propio G. Amdahl, surgió a partir una presentación y del artículo correspondiente para la Spring Joint Conference de 1967. Se trataba de comparar la potencia de un computador basado en un procesador de altas prestaciones con las de un computador paralelo, como era el ILLIAC IV, un computador SIMD presentado en la misma conferencia por D. Slotnick. Parece ser que el debate fue bastante intenso, y posiblemente, esto contribuyó a dar más publicidad a la correspondiente controversia. En [13], Amdahl se arrepiente de haber participado en una discusión que, en algunos momentos pareció referirse más a las personas que defendían cada alternativa que a la comparación de las propias alternativas. También habla de las propuestas que han mostrado los límites de aplicación de su ley, pero hace referencia a que, a través de ella se puso de manifiesto las exigencias que los multicomputadores deberían satisfacer respecto al aprovechamiento del paralelismo. De hecho termina indicando que Massively Parallel, una compañía dedicada a la fabricación de computadores paralelos, lo contrató para su comité de asesores.

The Pentium Chronicles: Introduction (R.P. Colwell) [15]. Este artículo del IEEE Computer es un resumen de la introducción del libro del mismo título publicado en 2006 por R. P. Colwell, responsable del equipo de diseño de la microarquitectura P6 de Intel. Esta ha sido una microarquitectura bastante relevante ya que permitió a Intel disponer de microprocesadores con características verdaderamente superescalares manteniendo su repertorio de instrucciones CISC compatible con los microprocesadores anteriores. La idea de incluir una etapa de decodificación que traduce las instrucciones CISC de los programas a un repertorio interno de microoperaciones con un formato RISC constituye una idea que parece bastante sencilla y directa pero que plantea sus dificultades desde el punto de vista de la implementación hardware. Esta misma tendencia, pero en el ámbito del software, es la que se ha aplicado en el desarrollo de arquitecturas novedosas posteriores como el Crusoe de Transmeta.

La propia descripción que hace Colwell del inicio del proyecto es reveladora, ya que pone de manifiesto el planteamiento de los objetivos de mejora de prestaciones a alcanzar (doblar las prestaciones del microprocesador anterior, el Pentium, incluso antes de que haya salido al mercado), y la organización del equipo de desarrollo. Se trata de un artículo ideal para incluir en los temas dedicados a los procesadores ILP, y

también se puede usar en aquellos en los que se hable de las alternativas RISC y CISC.

RISC: Back to the future? (C. G. Bell) [9]. La controversia que se produjo en la década de los 80 entre estas dos filosofías de diseño del repertorio de instrucciones máquina ilustra muy bien cómo, en ingeniería, razonamientos plausibles desde un punto de vista lógico pueden no verse corroborados por la realidad y hacen necesario un análisis cuantitativo y una evaluación experimental que refleje las características de las aplicaciones, tal y como se plantea en el libro de Arquitectura de Computadores de Hennessy y Patterson. En este artículo de Gordon Bell [9] la propuesta de las arquitecturas RISC se sitúa en el contexto del aumento de la capacidad de integración que hace posible diseños más complejos de microarquitecturas, y la inclusión caches en los microprocesadores, con lo que se puede reducir el tiempo de acceso a memoria. Además de proporcionar información acerca de los inicios de los RISC, también se ilustran los principios que caracterizan estos procesadores y se hace referencia a máquinas que pueden considerarse precursoras de esta filosofía de diseño.

En esta controversia RISC/CISC se ha puesto de manifiesto la importancia del compromiso entre el nivel de prestaciones que proporciona una alternativa y la compatibilidad con el software existente y cómo nuevas mejoras tecnológicas promueven microarquitecturas novedosas que anulan diferencias de prestaciones previas. En [8] se pone de manifiesto este proceso y cómo las discusiones sobre la superioridad de un repertorio RISC respecto a uno CISC fueron desapareciendo a lo largo de los 90, a la par que desaparecían muchos microprocesadores basados en esta filosofía, que ofrecían unos niveles de prestaciones considerables, mientras que los microprocesadores CISC copaban prácticamente todo el mercado, incluyendo el de los computadores de altas prestaciones. Concretamente, en [8] se hace referencia a un estudio de 1991, donde Bhandarkar y Clark [16] comparaban el VAX (repertorio CISC) y el MIPS (repertorio CISC), encontrando un factor de mejora del 2.7 para el MIPS respecto al VAX. Después se indica que, en un estudio de 1997 [17] en el que se comparaban el Alpha 21064 con el Intel Pentium, todavía se ponía de manifiesto la superioridad de la arquitectura RISC en algunos programas del SPEC CPU95 (entre un 5% y un 200% de mejora). Finalmente, en [8] se muestra como esa diferencia se redujo, comparando procesadores como el POWER5+ y el Xeon 5160. Dado que las técnicas más agresivas para extraer el paralelismo entre instrucciones se pueden implementar en microarquitecturas con repertorios CISC, la controversia RISC/CISC desapareció, como se pone de manifiesto en el mercado actual de microprocesadores.

Ultracomputers, a Teraflops before its time (G. Bell) [14]. Este trabajo proporciona una descripción detallada de la aparición de arquitecturas que buscan alcanzar los mayores niveles de prestaciones posibles en cada momento. Es interesante porque presenta las alternativas que se planteaban en ese momento para fabricar máquinas capaces de alcanzar el Teraflops (el artículo es de 1992, prevé que se alcanzará en 1995, pero hasta 1998 no se consiguió, con el ASCI Red según el TOP500), y se contempla la tendencia hacia máquinas paralelas con nodos basados en microprocesadores comerciales frente a las que utilizaban procesadores vectoriales como los computadores SX3 de NEC o los Cray C90, o los SIMD de Thinking Machines como el CM2. Así, este artículo es una fuente muy útil para generar discusiones sobre lo acertado de algunos de los juicios y previsiones del artículo, y se

puede comprobar hasta qué punto unas máquinas han sido más o menos exitosas. También se puede encontrar información acerca de precios, tamaños de las memorias, frecuencias de reloj, y velocidades pico de los principales computadores paralelos del momento, análisis de la escalabilidad de los distintos tipos de computadores. Se hace referencia a la mayoría de los computadores paralelos del momento, ya clasificados como multiprocesadores (de memoria compartida y memoria distribuida) y multicomputadores, y se ofrecen bastantes explicaciones relativas a la importancia de la jerarquía de memoria y comparaciones con otros computadores de altas prestaciones del momento. Se trata, pues, de un artículo muy adecuado para construir relatos de los temas que aborden el estudio de los multiprocesadores y los multicomputadores.

Creating the computer industry (E.W. Pugh; W. Aspray) [11]. En la innovación en el ámbito de los computadores, la industria ha contribuido de forma decisiva. Si una motivación fundamental para la construcción de los primeros computadores electromecánicos de los 40 fue su previsible utilidad en el desarrollo armamentístico imprescindible en el contexto de la segunda guerra mundial, pronto quedó clara las posibilidades de negocio que se abrían, aunque inicialmente ligadas a aplicaciones demandadas por el sector público. En este artículo de E.W. Pugh y W. Aspray [11] se pone de manifiesto cómo la interacción entre tecnología, fabricantes y clientes, determinó la evolución del computador desde los primeros dispositivos electromecánicos hasta los computadores modernos.

Un aspecto importante para el desarrollo de la industria de los computadores fue su desclasificación, a pesar de que estaban inicialmente ligados a la seguridad militar. Esto se debió, según el artículo, a que en realidad el ENIAC no pudo poner de manifiesto su utilidad militar en la segunda guerra mundial, y a que von Neumann expuso las cuestiones relacionadas con el diseño de computadores más en un ámbito académico ligado con los estudios neurológicos que en un entorno de ingeniería convencional. En realidad, el computador se veía como una calculadora de propósito general más que como una máquina para la guerra y había cierto interés por parte del ejército y de la Universidad de Pennsylvania de dar publicidad a los logros en electrónica que se habían alcanzado. En 1946, Eckert y Mauchly dejaron la Universidad de Pennsylvania para fundar una compañía dedicada a la fabricación y venta de computadores. El primero de ellos fue el UNIVAC, utilizado en 1951 por la oficina del Censo de los EE.UU., pero Eckert y Mauchly ya habían vendido su compañía a Remington Rand para conseguir el dinero y la experiencia empresarial que necesitaban. El UNIVAC apareció en la televisión para predecir los resultados de las elecciones presidenciales de 1952 y, durante un tiempo, llegaron a identificarse los términos UNIVAC y computador, que se contemplaba ya como una máquina electrónica capaz de realizar tareas de procesamiento de información entre las que las tareas de computación eran solo un subconjunto. El diseño del computador de programa almacenado fue el que hizo posible la transición de las tecnologías electromecánicas a las tecnologías electrónicas en los computadores: las instrucciones y los datos de los programas podían ser utilizados por los circuitos electrónicos con la rapidez necesaria a costes razonables. Otros elementos esenciales en el desarrollo del computador fueron las memorias de núcleo de ferrita, la tecnología de almacenamiento magnético, los circuitos de semiconductores, y los sistemas

operativos. A medida que empiezan a haber clientes en ámbitos de negocio distintos al de la defensa, la posición de los fabricantes como suministradores de máquinas en estos sectores se hizo decisiva. Así, IBM que tenía alrededor del 90% del mercado de los equipos de tarjetas perforadas fue capaz de absorber gran parte del mercado de los computadores, a diferencia de lo que ocurrió con Sperry Rand que, a pesar de su tamaño, tenía una menor participación en el negocio de los equipos de tarjetas perforadas, y una falta de liderazgo centralizado y fuerte. Igual sucedió a RCA y GE, mientras que empresas más pequeñas como CDC, DEC, Burroughs y NCR encontraron mejores enfoques. A comienzos de los 60 ya se puede considerar que había aparecido el computador moderno y que existía un volumen de clientes sensibles al coste del computador que promovió el desarrollo de productos más estandarizados. En este contexto se anuncia el IBM S/360 que no es capaz de satisfacer todos los requerimientos del mercado en un momento en que la mejora en coste y prestaciones de las tecnologías del computador y la aparición de una gran diversidad de aplicaciones complejas dieron lugar a que apareciesen nuevas oportunidades para los fabricantes de computadores (los supercomputadores). Este artículo ofrece bastante información de las empresas pioneras en la fabricación de computadores, y anécdotas acerca de la intervención de las empresas en el desarrollo de los computadores, y de la interacción entre tecnología y demanda de aplicaciones con el propio desarrollo de la arquitectura de computadores.

5 Conclusiones

La generación de mapas conceptuales para los contenidos de una asignatura permite integrar el conocimiento adquirido de forma interrelacionada, contribuyendo a mejorar la memorización de dichos conceptos. La capacidad para avanzar en la comprensión de una asignatura y la resolución de problemas planteados en la disciplina mejora la motivación hacia la misma. Por tanto, la utilización de relatos acerca del proceso histórico que ha generado el cuerpo de conocimientos de una asignatura contribuye a crear relaciones entre los conceptos y las circunstancias en las que se han desarrollado.

Además de aprovechar la facilidad para memorizar relatos, al poner de manifiesto la dimensión social aparejada al desarrollo de nuevas propuestas, se dota a los conceptos de una serie de características emotivas que contribuyen a la motivación, y con ello facilitan la asimilación de los conceptos. En relación con estos planteamientos, en este artículo se han propuesto y comentado una serie de artículos de los que se pueden extraer esos relatos, y se ha indicado su relación con los mapas conceptuales de la asignatura que, como se ha dicho, pueden contribuir a mejorar el aprendizaje de los estudiantes.

6 Referencias

1. Implagliazzo, J.; Lee, J.A.N.:”Using computing history to enhance teaching”. History of Computing in Education, pp. 165-176, 2004.

2. Papadimitriou, C.: “Mythematics: In Praise of Storytelling in the teaching of Computer Science and Math”. Invited Editorial, ACM inroads, Vol. 35, No.4, pp. 7-9. Diciembre, 2003.
3. Ortega, J.; Anguita, M.:”Arquitectura de Computadores en seis créditos ECTS”. Enseñanza y Aprendizaje de Ingeniería de Computadores, pp.13-25 (http://hdl.handle.net/10481/20528), 2012.
4. Hennessy, J.L.; Patterson, D. A.:”Computer Architecture. A Quantitative Approach”. Elsevier Science & Technology, 2011 (5ª Edición).
5. Ortega, J.; Anguita, M.; Damas, M.; González, J.:”Motivación para la Ingeniería de Computadores”. Enseñanza y Aprendizaje de Ingeniería de Computadores, pp.3-22, (http://hdl.handle.net/10481/26351), 2013.
6. Patt, Y.:”Ten Commandments for Good Teaching”. (http://users.ece.utexas.edu/~patt/Ten.commandments/), 2013.
7. Ende, J. van den; Kemp, R.:”Technologival transformations in history: how de computer regime grow out of existing computing regimes”. Research Policy, 28, pp.833-851, 1999.
8. Isen, C.; John, L.; John, E.:”A Tale of two processors: revisiting the RISC-CISC debate”. Springer-Verlag LCNS 5419, pp. 57-76, 2009.
9. Bell, C.G.:”RISC: Back to the Future?”. DATAMATION, pp.76-82. Junio, 1987.
10. Smith, R.E.:”A Historical overview of Computer Architecture”. Annals Hist. Computer, 10, pp. 277-303, 1989.
11. Pugh, E. W.; Aspray, W.: “Creating the Computer Industry”. IEEE Annals of the History of Computing, Vol. 18, No.2, pp. 7-17, 1996.
12. Elzen, B.; Mackenzie, D.: “The social limits of Speed: The development and use of Supercomputers”. IEEE Annals of the History of Computing, Vol. 16, No.1, pp. 46-61, 1994.
13. Amdahl, G.M.: “Computer Architecture and Amdahl`s Law”. IEEE Computer, pp.38-46. Diciembre, 2013.
14. Bell, G.:”Ultracomputers, a Teraflops before its time”. Communications of the ACM, Vol.35, No.8, pp.27-47. Agosto, 1992.
15. Colwell, R.P.: “The Pentium Chronicles: Introduction”. IEEE Computer, pp.49-54. Enero, 2006.
16. Bhandarkar, D.; Clark, D.W.:”Performance from architecture: comparing a RISC and a CISC with similar hardware organization”. Proc. of ASPLOS, pp. 310-319, 1991.
17. Bhandarkar, D.:”RISC versus CISC: a tale of two chips”. ACM SIGARCH Computer Architecture News, Vol. 25, No.1, pp. 1-12, 1997.

Arquitectura Orientada a Servicios, un enfoque basado en proyectos

Marcela Castro-León[1], Francesc Boixader[1], Dolores Rexachs[2], Emilio Luque[2]
[1]Escola Universitària d'Informàtica Tomàs Cerdà.
Sant Cugat del Vallès, Barcelona, España
{marcela.castro, francesc.boixader}@eug.es
[2]Universidad Autónoma de Barcelona.
Barcelona, España
{dolores.rexachs, emilio.luque}@uab.es

Resumen. En este artículo se presenta el enfoque metodológico que se utiliza en la asignatura de Arquitectura Orientada a Servicios (AOS) que se imparte como obligatoria en el tercer curso del Grado de Informática y Servicios en la Escola Universitària d'Informàtica Tomàs Cerdà. AOS, es innovadora en un nivel educativo de grado, y propone, en la parte teórica de la metodología, enseñar las propiedades y características de la AOS de un modo conceptual relacionándolo con la tecnología existente. De este modo, los alumnos disponen de los criterios necesarios para valorar la oferta de productos que existen en el mercado actual, y adquieren conocimientos que van más allá de una línea de productos en concreto. En la parte práctica, se ha optado por un enfoque de aprendizaje basado en proyectos soportado por video-tutoriales, se consigue que el alumno proponga y desarrolle un proyecto basado en servicios web en tan solo una semana de uso de laboratorio. El método permite que el alumno aprenda, en forma consolidada y aplicada, los conceptos de tecnologías de servicios web SOAP, ReST y bus de servicios.

Palabras Clave: Arquitectura Orientada a Servicios, Servicios Web, Bus de servicios, SOAP, ReST

Abstract. In this paper we present the methodological approach we are using in Service Oriented Architecture (SOA), which is taught as a mandatory subject during the third course in Informatics and Services degree at the *Escola Universitària d'Informàtica Tomàs Cerdà*. This subject, innovative at grade level, proposes, in the theoretical part of its methodology, to teach the SOA properties and characteristics in a conceptual way and in relation to the current technology. As a result, the students are provided with the required elements to assess the current SOA product supply and they acquire the SOA knowledge beyond of a concrete product line. In the practical part, we choose project-based collaborative learning model supported with video-tutorial, we achieve the pupils could propose and develop a web services based project during using only a laboratory week. This method allows the students to learn in a practical way the technology concepts of SOAP, ReST, and service bus.

Keywords: Service Oriented Architecture, web services, Enterprise service bus, ESB, SOAP, ReST

1 Introducción

El Grado de Informática y Servicios [1] que se imparte en la *Escola Universitària d'Informàtica Tomàs Cerdà* contempla una formación pluridisciplinar que abarca sectores tan diversos como el de servicios, el de las TIC o el financiero, y surge al detectar en el tejido económico de este país una demanda de profesionales en instituciones financieras y de seguros, gabinetes de asesoría científico-técnica, empresas con equipos interdisciplinarios de I+D y de transferencia de tecnología y, en el sector sanitario. Esta visión de acercar la informática al mundo de los servicios contiene un grado de originalidad, siendo esta propuesta pionera a nivel español.

Las AOS se han introducido en el mercado del software, por los beneficios relacionados que ofrecen a la hora de desarrollar aplicaciones distribuidas. El servicio es un componente de software asociado a un contrato o interfaz bien definida, y que se ejecuta en forma independiente del lenguaje en que fue creado y de la plataforma que lo provee o lo consume, utilizando protocolos estándares como HTML, XML o SOAP. Propone un acoplamiento orientado a mensajes con un mayor grado de autonomía que el de la computación tradicional de multiniveles, que facilita la reutilización, el diseño modular y por capas. El servicio se puede componer en servicios de mayor granularidad, con mayor nivel de abstracción hasta alcanzar una nivel adecuado para definir la gestión de procesos de una empresa y para aplicar seguridad de acceso y de confidencialidad de los datos. Además, esta arquitectura se adapta mejor a la nueva infraestructura, plataforma y servicios que ofrece *Cloud Computing*. El enfoque de AOS para los sistemas implica pensar y enseñar diferente considerando el objetivo de construir sistemas más efectivos [2].

Lawler y otros en [3] indican la importancia de introducir metodologías de enseñanza de AOS para apoyar el proceso de grandes inversiones en AOS que han realizado en los últimos años las empresas y que, sin embargo, falta un largo recorrido hasta lograr que las empresas sean *Service Oriented Enterprise* (SOE), es decir, que sus procesos de negocio estén completamente orientados a servicios.

La metodología de la asignatura AOS se basa, en la parte teórica, en transmitir al alumno las propiedades y características de AOS de modo conceptual y relacionarlo con la tecnología existente. De este modo, los alumnos disponen de los elementos necesarios para comprender las diferentes propuestas que existen en el mercado actual y, a su vez, adquieren conocimientos que van más allá de una línea de productos en concreto. En la parte práctica, se ha optado por un enfoque basado en proyectos utilizando herramientas de desarrollo con librerías y *wizards* altamente productivos y en video-tutoriales con los que el alumno puede desarrollar un proyecto basado en servicios web en una semana de práctica de laboratorio, trabajando en grupos de 2/3 personas. El método permite que el alumno aprenda en forma consolidada y aplicada los conceptos de tecnologías de servicios web SOAP, ReST y bus de servicios.

Este artículo presenta en la Sección 2, el contexto, el contenido y el enfoque general. En la Sección 3, se describen las competencias, los objetivos y el temario de la asignatura. En la Sección 4, se expone la metodología. En la Sección 5 se comentan otros enfoques de centros donde se estudia AOS. En la Sección 6 explicamos los resultados obtenidos y por último, en la Sección 7, enunciamos nuestras conclusiones.

2 La asignatura de Arquitectura Orientada a Servicios

2.1 Contexto

La Escola Universitària d'Informàtica Tomàs Cerdà, adscrita a la Universidad Autónoma de Barcelona, elaboró un plan de estudios para el Grado de Informática y Servicios [1] que se incorporó desde el curso 2004-05 al plan piloto de la Universidad Autónoma de Barcelona para la adaptación al Espacio Europeo de Educación Superior y que se empezó a impartir en el curso 2009-10.

El perfil profesional objetivo que se puede adquirir mediante este grado se ajusta al de un ingeniero informático, dado que integra competencias en tecnología e ingeniería, que son complementadas con una formación más específica en el área de la gestión y organización empresarial, así como conocimientos, tanto en ciencias de la computación, comunicaciones y tecnologías web y multimedia. Este profesional debe ser capaz de evaluar, diseñar e implementar los sistemas de información en los que la tecnología juega un papel clave, con el objeto de gestionar la información utilizada por las empresas de servicios en todas sus áreas de negocio, en particular, aquellas que ofrecen los servicios a través de Internet, y, para hacerlo de manera óptima, tienen en cuenta la experiencia del usuario interno y externo a la empresa.

La asignatura AOS se imparte en el tercer año del grado en Informática y Servicios y está incluida en uno de los cinco bloques de contenidos del grado, en concreto en el de Servicios. AOS es una de las dos asignaturas obligatorias de la primera materia del bloque.

AOS es una asignatura que no aparece, como tal, en los planes de estudios de Grado de Informática, lo que hace que represente un elemento diferencial del Grado de Informática y Servicios con respecto al resto de ingenierías informáticas. AOS representa un modelo de integración entre software y empresa, concepto que es considerado clave en el desarrollo de este grado.

2.2 Contenidos principales de la asignatura

Los contenidos básicos de la asignatura son los siguientes:

- Arquitectura orientada a servicios.
- Arquitecturas de negocio y arquitectura orientada a servicios (SOA).
- Fundamentos del modelo de referencia de la organización de estándares de sistemas de información (OASIS).
- Orquestación en la arquitectura orientada a servicios.
- Bus de servicios de empresa y procesador de eventos inteligente.
- Lenguajes de especificación de servicios multiusuario, seguros y reutilizables.
- Metodologías de desarrollo de aplicaciones de servicios.
- Despliegue, mantenimiento y test de sistemas de gestión de procesos de negocio (BPMS).

Podemos distinguir cuatro pilares en el temario. En primer lugar, el concepto de AOS como arquitectura de negocio y de software. Luego, la definición de estándares,

fundamentales para la comunicación de procesos en forma independiente de la tecnología y los lenguajes que los implementa. En tercer lugar, el bus de servicios, una de las piezas claves de AOS, es el que facilita la integración de los estándares con múltiples puntos de acceso y permite orquestar, procesar, registrar y controlar los diferentes eventos en la operación diaria de una empresa. Por último, la gestión de procesos de negocio, conocida por *Business Process Management* (BPM), es la cara visible de una empresa con AOS. BPM saca provecho a la arquitectura ofreciendo herramientas que permiten adaptar y desplegar cambios de los procesos de negocio en forma eficiente.

2.3 Un enfoque conceptual y práctico

El perfil del profesional del graduado en Ingeniería y servicios está pensado para ofrecer las nuevas habilidades tecnológicas que requieren las empresas del sector TIC del siglo XXI. Presenta una orientación profesional, y se alinea de la forma más efectiva posible con las necesidades del mercado laboral.

Por tanto, es necesario que el estudiante se familiarice con las tecnologías emergentes. No obstante, es necesario seleccionar aquellas que provienen de grupos de empresas reconocidas y/o que, pudiendo venir de grupos de *open-source*, utilicen estándares con un alto grado de aceptación por parte de la comunidad usuaria para dotar a los alumnos de habilidades no efímeras como pueden serlo algunas de las tecnologías emergentes que implementan AOS.

Durante la asignatura se enseña a diferenciar entre los conceptos teóricos relativos a la AOS y las propiedades de los productos de software disponibles en el mercado en la actualidad. Así entonces, el alumno desarrolla criterios que le permite reconocer con mayor facilidad las capacidades de un producto en concreto respecto al modelo teórico. Se pretende dotar a los alumnos de un conocimiento conceptual que tiene una mayor duración que el uso de una tecnología específica.

3 Diseño de la asignatura

3.1 Competencias

Del conjunto de competencias asociadas al grado de Informática y Servicios, la asignatura AOS selecciona las siguientes competencias específicas:

1. Evaluar sistemas hardware/software en función de un criterio de calidad determinado.
2. Adaptar la tecnología que sea realmente eficiente en su papel de intermediario en el proceso comunicativo entre las empresas de servicios y los usuarios de dichos sistemas.
3. Analizar, modelar y optimizar el conjunto de los procesos involucrados en los servicios, antes y después de su implementación.

4. Gestionar la colaboración de los diferentes "actores" involucrados en la definición, diseño, implementación y explotación de servicios, así como integrarlos con las distintas tecnologías y metodologías.

De entre las competencias transversales asociadas al grado, la de Gestionar (planificar) el tiempo y los recursos disponibles, es la que en mayor grado se trabaja, por parte del alumnado, en el contexto de AOS.

3.2 Objetivos

A continuación se describen los objetivos que cumplen los alumnos que cursan con éxito la asignatura AOS, a través de los cuales se desarrollan las competencias mencionadas.

- Definir la arquitectura orientada a servicios, sus componentes en cada una de las capas de software (plataforma SOA) y las relaciones entre ellos.
- Conocer las propiedades de las aplicaciones orientadas a servicios, sus beneficios y los desafíos de implantar SOA en una empresa.
- Definir las objetivos particulares y características de cada uno de los componentes de la arquitectura orientada a servicios.
- Analizar productos de mercado AOS *open-source* y comerciales.
- Conocer las tecnologías y estándares más utilizados en la arquitectura SOA.
- Diseñar y desarrollar aplicaciones basadas en servicios web basados en protocolo SOAP y en la arquitectura ReST.
- Utilizar un Bus de servicios web para implementar ruteo y enmascaramiento de servicios web.

3.3 Contenidos

El programa de contenidos de la asignatura contempla los siguientes temas:

Tema 1.- Introducción a la Arquitectura Orientada a Servicios.
- 1.1. Definición, objetivos, fundamentos.
- 1.2. Plataforma tecnológica de SOA
- 1.3. Propiedades de SOA

Tema 2.- Tecnología de servicios web.
- 2.1. Estándares de servicios web.
- 2.2. Modelo de servicios web basados en SOAP.
- 2.3. Modelo de servicios web *Representational State Transfer (ReST).*
- 2.4. Diferencias, ventajas y desventajas entre modelos de WS SOAP y ReST.

Tema 3.- Productos SOA
- 3.1 Productos SOA *Open Source* para cada componente SOA
- 3.2 Establecimiento de criterios de selección según los objetivos empresariales.
- 3.3 Estudio de productos SOA comerciales.

Tema 4.- Composición de servicios WEB

4.1 Definición, fundamentos de *Service Component Architecture*.
4.2 Principales elementos de *Service Component Definition Language (SCDL)*.
Tema 5.- Integración de aplicaciones *Enterprise Service Bus (ESB)*
5.1 Definición y fundamentos del bus de servicios
5.2 Principales funcionalidades.
5.3 Implementaciones *ESB* más utilizadas en el mercado actual.
Tema 6.- Procesamiento de flujos de eventos *Event Stream Processing (ESP)*
6.1 Definición y fundamentos de *ESP*.
6.2 Estudio de la implementación *Esper*
Tema 7.- Gestión de procesos de negocio *Business Process Manager* (BPM)
7.1 Orquestación de servicios con BPEL.
7.2 Principales lenguajes estándares BPEL, BPMN i JPDL.

Este programa está diseñado para ser desarrollado en un semestre académico que comprende 16 semanas de clases teóricas, a razón de 3 horas de clase semanales, y otras 3 semanas de trabajo práctico en el laboratorio, con un total de 18 horas, bajo la directa supervisión del profesorado de la asignatura.

4 Metodología docente

4.1 Clases teóricas

La metodología utilizada para la enseñanza de los conceptos es *top-down*, en el sentido de que en el primer tema se da una visión completa de la arquitectura AOS, y en el resto, se desarrolla cada uno de los componentes profundizando en los detalles de los mismos. Esta metodología facilita la comprensión del temario por parte del alumno al incorporarlo de forma relacionada con el concepto integrador de AOS.

En la literatura se encuentran diferentes visiones o enfoques de AOS. La que hemos adoptado en la asignatura es considerar que representa una arquitectura de software abierta, ágil, extensible, federada y combinada, constituida por servicios autónomos, capaces de gestionar la calidad del servicio, posiblemente de diferentes proveedores, inter-operativos, potencialmente reutilizables e implementados todos ellos como servicios web [4]. Los principales beneficios que promete la adopción de esta arquitectura es simplificar el desarrollo y la implementación, reutilizar software, aprovechar mejor el gasto en servicios de computación, dar un criterio para la toma de decisiones y guiar las acciones de la empresa y, en particular, del departamento de IT.

En primera instancia, se desarrolla el concepto de AOS a través de la plataforma de componentes en sus diferentes niveles. En la figura 1, en la parte izquierda, se representan dichos niveles, comenzando por los servicios web que se ubican por encima de los sistemas *legacy* de la empresa. El siguiente nivel está formado por los servicios de negocio que se generan por composición de *web-services (*WS), siguiendo luego con los procesos de negocio (BPM) y la capa de presentación. En la parte derecha de la figura 1, se muestran los componentes que AOS añade a la infraestructura, como son el bus de servicios, el registro, las reglas de negocio, la monitorización y la seguridad. La bibliografía básica de referencia utilizada para el

desarrollo de la plataforma AOS son los libros de Jeff Davis [5] y el de Juric [6]. Se comprende de este modo a la arquitectura AOS en forma global con sus propiedades y ventajas, por estar basada en servicios web y en comunicaciones estándares.

Luego, durante el transcurso del semestre, el alumno estudia en mayor profundidad cada uno de los componentes de la plataforma tecnológica sin perder la visión holística que se ha dado en un primer momento.

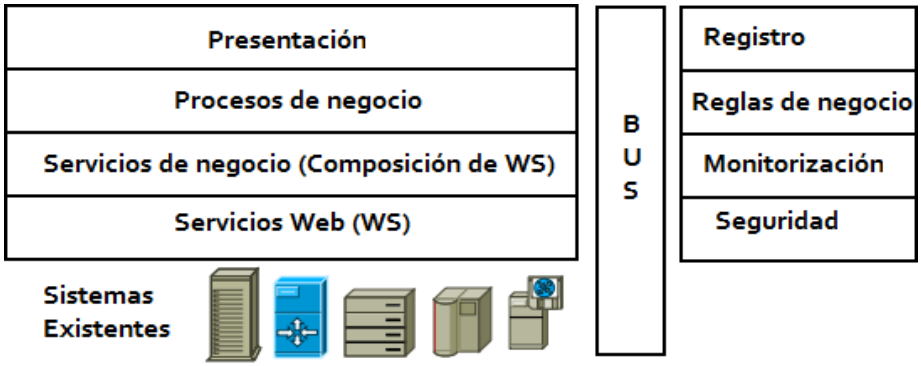

Figura 1. Plataforma tecnológica AOS [6].

4.2 Clases prácticas - Metodología

La metodología de las clases prácticas sigue las líneas del aprendizaje colaborativo basado en la elaboración de proyectos [7].

Durante el transcurso del semestre, se destinan tres semanas para las clases prácticas en el laboratorio que denominaremos 1, 2 y 3. Estas semanas se ubican entre las 8 y 17 de las 19 que normalmente tiene un semestre. Los alumnos trabajan en grupos de 2/3 personas, y durante dicha semana desarrollan un proyecto relacionado con el tema objetivo de cada práctica. A la semana siguiente, entregan y presentan los trabajos realizados.

Para que los alumnos dispongan de la información necesaria para llevar a cabo los proyectos de estas prácticas, se utiliza el blog Koonek [8], creado para esta asignatura. El blog está basado en videos-tutoriales que enseña al alumnado ejemplos de desarrollo de servicios web utilizando una plataforma basada en *Netbeans*[9], [10], *Glassfish* [11] y *Java* [12]. Esta plataforma ha sido seleccionada porque es una de las más consolidadas y desarrolladas en el uso de servicios web, y adoptada por un gran número de usuarios. Ha sido una de las pioneras en presentar implementaciones de servicios web para el desarrollo de aplicaciones. Esta vanguardia ha hecho que *Netbeans* disponga en la actualidad de un conjunto de opciones que facilitan y automatizan la conversión de una clase en servicio web y la creación de consumidores de estos servicios.

Las prácticas 1 y 2 son de servicios web, de SOAP [13] y ReST [14], respectivamente. Cada grupo ha de presentar una propuesta de trabajo práctico a desarrollar, a más tardar, una semana antes de la práctica. Dicha propuesta ha de

incluir, al menos, un proveedor y un consumidor de servicio web. La propuesta se discute con el profesor para asegurar que el nivel de complejidad sea el adecuado para completar el trabajo en el tiempo asignado. Los servicios web proveedores tienen que incluir gestión de bases de datos o, bien uso de estructuras de datos complejas. La siguiente tabla **1** muestra cada una de las prácticas, indicando el objetivo y las habilidades que los alumnos tienen que haber adquirido antes del inicio de cada una.

Tabla 1. Relación entre práctica, objetivo y habilidades requeridas

Práctica	Objetivo	Habilidades requeridas
1	Servicios WEB basados en SOAP	Creación de servicios web SOAP proveedores y consumidores.
2	Servicios WEB basados en ReST	Creación de servicios web ReST proveedores y consumidores.
3	Bus de servicios	Manejo de la herramienta de bus seleccionada

4.3 Clases prácticas - Desarrollo

Para la práctica 1 de WS SOAP se utiliza la especificación de java JAX-WS [15] y para la práctica 2, hemos adoptado el uso de la especificación JAX-RS [16], aunque en los servicios WS ReST también puede utilizarse XML y librerías de HTTP.

La instalación y configuración por defecto de las herramientas seleccionadas disponen de las librerías necesarias para el desarrollo de los servicios web. Esta característica, sumada a la provisión de *wizard* en *Netbeans,* permite que los alumnos desarrollen y desplieguen aplicaciones en forma muy rápida y sin errores, experimentando así los conceptos aprendidos, con un alto grado de satisfacción.

Los ejemplos que se presentan en el blog aumentan, gradualmente, el grado de complejidad hasta llegar al nivel que se les requiere en la propuesta a realizar.

En las clases anteriores a la entrega de propuesta, se presentan, comentan y discuten los video-tutoriales. Los alumnos disponen de la plataforma de desarrollo en sus máquinas, al menos uno por grupo, lo que les permite practicar en forma autónoma y/o en grupo. De este modo, los alumnos aprenden a utilizar las herramientas y a desarrollar WS en base a los ejemplos del blog.

La práctica 3 se dedica al uso de bus de servicios. Esta herramienta es de uso altamente recomendable en proyectos AOS debido a que permite integrar productos y servicios de diferentes tecnologías, centralizando en un lugar las políticas de gestión de diferentes protocolos y formatos de intercambio de datos, así como la política de ruteo de servicios, colas de mensajes, distribución y balanceo de carga, niveles de seguridad, control de accesos y auditoría. Para el desarrollo de esta práctica, se utiliza

el producto *Oracle Service Bus* [17] ampliamente conocido en el sector empresarial. Este aplicativo trabaja en conjunto con el servidor de aplicaciones *WebLogic*.

Hemos seleccionado este producto porque provee un conjunto de tutoriales que explican, paso a paso, la implementación de un escenario para una empresa de préstamos de capital que muestra cómo realizar ruteo, transformación y validación de servicios [18]. Además dispone de un procedimiento de instalación que, aunque de una complejidad media, se encuentra lo suficientemente consolidado y con el soporte necesario para llevar a cabo las instalaciones en el laboratorio sin mayores incidencias.

La tabla 2 muestra los video-tutoriales que se publican en el blog en el orden que son presentados a los alumnos antes de cada práctica. La práctica 3 utiliza tutoriales en papel disponibles de la página de Oracle [18].

Tabla 2. Ejemplos tutoriales publicados en el blog para las práctica 1 y 2

Práctica	Ejemplos	Propósito didáctico
1	Creación de WS SOAP proveedor	Procedimiento de desarrollo de un WS SOAP, despliegue y test a partir de una clase java.
1	Creación de WS SOAP consumidor	Creación de un programa java en web y cliente pesado que invoque al WS proveedor creado por el usuario o de uso público.
1	Proyectos de años anteriores	Proyectos con WS proveedores con estructuras de datos complejas o manejo de base de datos y consumidores de WS embebidos en programas con interfaz de usuario basadas en java script, *servlets* y/o *Java Server Faces*.
2	Creación de WS ReST proveedor.	Procedimiento de desarrollo de un WS proveedor ReST.
2	Creación de WS SOAP consumidor	Creación de una interfaz HTML5 que permita invocar a un WS ReST.
2	Proyectos de años anteriores	Proyectos que integren WS ReST proveedores y consumidores entregados los años anteriores.

Los documentos a entregar al final de cada práctica (proyecto) incluyen:

- **Memoria de contenido:** Tiene que incluir la propuesta, objetivo, conceptos teóricos relacionados, mapa de arquitectura de la solución, desarrollo del trabajo, forma de instalación y despliegue y conclusiones.
- **Presentación:** Cada grupo presenta el trabajo realizado en cada práctica.
- **Autorización para la publicación del trabajo.** Cada alumno entrega su consentimiento para la publicación del trabajo práctico en el blog.

4.3 Método de evaluación

El modelo de evaluación de la asignatura AOS consiste de:

- Primer Examen: 15%. Se realiza antes de la semana 8 para que los alumnos estudien los conceptos de la plataforma AOS que se desarrollan en la primera parte.
- Examen Final: 25%. Examen teórico de toda la asignatura al final del semestre.
- Prácticas: 30% . (10% cada una).
- Asistencia y actitud: 10%
- Informes de prácticas: 10%
- Presentaciones orales: 10%

Se ha optado por otorgar idéntico peso a la teoría y a la práctica, -ya que el 10% de diferencia corresponde a las presentaciones orales que se refieren a las prácticas grupales pero que se valoran en forma individual-, lo cual representa un incentivo para que los alumnos dediquen un esfuerzo adicional alas prácticas de la asignatura..

5 AOS en otros centros educativos

La asignatura AOS se imparte en la Universidad de Granada, pero a nivel de Posgrado [19]. En dicha propuesta, AOS es visto como una arquitectura de desarrollo de WS y no como una plataforma general de infraestructura y software empresarial.

En la Universidad de La Laguna, en el posgrado en informática para la obtención del Máster Universitario en informática y se incluye la asignatura Arquitectura Orientada a Servicios [20]. El temario propuesto es similar al de nuestra asignatura, pero el enfoque se basa en que el alumno diferencie bien tres roles, como son el de analista de negocios, arquitectos de software y desarrolladores/ programadores.

La Universidad *North Carolina State University* en el grado de *Computer Sciences* imparte el curso de *Service-Oriented Computing* [21]. En dicho curso se enfatiza en conceptos, teorías y técnicas de servicios web a alto nivel. No se relaciona con la orientación a los procesos de negocio.

La Universidad de California en *Berkeley School of Information*, ofrece un conjunto de cursos específicos [22] en los cuales incluye los cursos de *design and development of web-based products and services* y *Web-based Services*, en los cuales se cubre parte del temario de nuestra asignatura, pero con un enfoque basado al desarrollo de aplicaciones.

La Universidad de Manchester, en School of Sciences, ofrece un proyecto de postgrado Postgraduate Research Projects denominado Mapping Applications and Services on Computing Clouds and Service-Oriented Systems [23]. En este proyecto se usan los sistemas orientados a servicios en lugar de la computación paralela tradicional por considerarla más dinámica donde múltiples servicios pueden estar ejecutándose.

En la Universidad de Warwick, en *UK Manufacturing Group (WMG)*, que es el departamento de la facultad de ciencias dedicado a ofrecer cursos de Máster en tecnología e industria, imparte un curso denominado *Service Management and Design*

[24] . En dicho curso se imparte una formación orientada a los líderes de funciones de servicio de una variedad de industria, orientado a aprender los sistemas y las técnicas usadas para gestionar servicios y creación de valor en los negocios y en los procesos.

6 Resultados Obtenidos

La metodología del aprendizaje basado en proyectos para las prácticas se ha utilizado desde que se comenzó a impartir la asignatura para fomentar el aprendizaje autónomo y en grupo, la gestión del tiempo y los recursos disponibles el uso. Sin embargo, en el último curso académico 2013/2014, hemos incorporado el uso de blog con video-tutoriales como herramienta de soporte principal para los grupos de alumnos que encontraban dificultades durante el desarrollo de proyectos basados en servicios web. Se presentaban muchos errores, generalmente causados por un mal uso de las herramientas. El video-tutorial ayuda a mostrar cómo se han de utilizar correctamente. Al final del semestre, se realizó una encuesta, para saber la valoración del blog por parte de los alumnos, en la que preguntó *¿Cómo valoras el hecho que se disponga de un blog con ejercicios resueltos y trabajos prácticos anteriores?*. En una respuesta de 1 a 5, (5 es la máxima valoración), se obtuvo una valoración media de 4, es decir, un 80%. Sin embargo, hemos observado que en la ejecución de las prácticas, los alumnos realizaron propuestas que pueden adaptarse a proyectos existentes, y luego copiaron el código de los proyectos en las partes que podían, muchas veces sin una adaptación correcta. Para evitar este comportamiento, proponemos, por un lado, en el momento de la validación de la propuesta, se alerta sobre el parecido a un proyecto existente. Y por otro lado, se comunicará a los grupos la penalización grave en la nota en caso de encontrar copias de código que no corresponda a una correcta reutilización y adaptación de servicios.

7 Conclusiones

Se ha presentado la metodología actual de la asignatura Arquitectura Orientada a Servicios que se dicta en el tercer curso del Grado de Informática y Servicios de la Escuela Universitaria de Informática Tomàs Cerdà. La propuesta se basa en que los alumnos, en el transcurso del semestre, tengan una visión completa del potencial de la AOS para el desarrollo de software y de cómo esta arquitectura impacta en la gestión que la empresa realiza de un software integrado entre todos los sectores y departamentos. Una vez superada la asignatura, los alumnos son conocedores de la reutilización de los servicios y de su integración en aplicaciones independientemente de su tecnología y localización. AOS se convierte en un modelo altamente recomendable para que las empresas migren sus aplicaciones al paradigma del *Cloud computing*, debido a que por medio del servicio se puede administrar mejor la seguridad y realizar un uso más eficiente de recursos.

Referencias

1. Boixader, Francesc [et al.]. El Grado en Informática y Servicios. Una respuesta a la nueva demanda del contexto social. A: JENUI 2010. "XVI Jornadas de Enseñanza Universitaria de la Informática". Universidade de Santiago de Compostela. Escola Técnica Superior d'Enxeñaría, 2010, p. 130-135..
2. B. A. Maurizio, J. Sager, P. Jones, G. Corbitt, and L. Girolami, “Service Oriented Architecture : Challenges for Business and Academia 3 . The Role of Business Process Management in SOA,” pp. 1–8, 2008.
3. J. P. Lawler, H. Howell-barber, D. Colton, and K. A. Grant, “Critical Success Factors in the Planning of a Service-Oriented Architecture (SOA) Strategy for Educators and Managers,” vol. 7, no. 94, 2009.
4. T. Erl, Service-oriented architecture: concepts, technology, and design. Pearson Education India, 2005.
5. J. Davis, Open Source Soa. Manning Publications Co., 2009.
6. M. B. Juric, SOA Approach to Integration: XML, Web Services, ESB, and BPEL in Real-world SOA Projects (Google eBook). 2007.
7. B. Antoni and G. Consuelo, “Incorporación de las TIC en la enseñanza y el aprendizaje basados en la elaboración colaborativa de proyectos,” RUSC. Univ. Knowl. Soc. J., vol. 3, pp. 42–54, 2006.
8. “Koonek.” [Online]. Available: http://koonek.blogspot.com.es/.
9. “Netbeans.” [Online]. Available: http://www.netbeans.org.
10. H. Böck, The Definitive Guide to NetBeansTM Platform 7. Berkeley, CA: Apress, 2011.
11. “Glassfish.” [Online]. Available: http://www.glassfish.org.
12. “Java.” [Online]. Available: http://www.java.com/.
13. N. Mitra, Y. Lafon, and others, “Soap version 1.2 part 0: Primer,” W3C Recomm., vol. 24, p. 12, 2003.
14. R. T. Fielding and R. N. Taylor, “Principled design of the modern Web architecture,” ACM Trans. Internet Technol., vol. 2, no. 2, pp. 115–150, May 2002.
15. A. P. I. Java, “for XML-based web services (JAX-WS) 2.0.” .
16. M. Hadley and P. Sandoz, “JAX-RS: JavaTM API for RESTful Web Services,” 2008.
17. “Oracle Service Bus.” [Online]. Available: http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html.
18. “Oracle Service Bus Tutoriales.” [Online]. Available: https://java.net/downloads/oraclesoasuite11g/OSB/osb_examples_tutorials_111130.pdf.
19. P. García-Sánchez, M. Á. López Gordo, P. Castillo Valdivieso, J. González Peñalver, and M. I. García Arenas, “Web 2.0: Arquitectura orientada a servicios en Java,” 2011.
20. E. S. Nielsen, “Arquitecturas orientadas a servicios: EEES, metodología docente y primeros resultados,” Jornadas Enseñanza Univ. la Informática.
21. “North Carolina State University-Computer Sciences- Service-Oriented Computing.” [Online]. Available: http://www.csc.ncsu.edu/faculty/mpsingh/local/SOC/s13/.
22. “University of California, Berkeley School of Information.” [Online]. Available: http://www.ischool.berkeley.edu/courses/i290.
23. M. Stubbs and P. Range, “Service-oriented architecture and curriculum transformation at Manchester Metropolitan University,” Campus-Wide Inf. Syst., vol. 28, no. 4, pp. 299–304, 2011.
24. “Warwick University, UK Manufacturing Group (WMG).” [Online]. Available: http://www2.warwick.ac.uk/fac/sci/wmg/education/wmgmasters/courses/masters_services_management_design/.

DESDE EL PUPITRE
(Experiencias de estudiantes)

Ingeniería de Servidores desde la perspectiva de un estudiante

Jorge Chamorro Padial

Grado en Ingeniería Informática. E.T.S. Ing. Informática y de Telecomunicación, Universidad de Granada
Granada, España
jorgechp@correo.ugr.es

Resumen. Ingeniería de Servidores es una asignatura que se imparte en el tercer año del Grado en Ingeniería Informática. En este artículo se analiza, desde el punto de vista de un estudiante, la evolución de la asignatura. La metodología y el sistema de evaluación son descritos y examinados con el fin de aportar una valoración de debilidades y fortalezas observadas tanto por el autor como por estudiantes de otros grupos cuya opinión queda reflejada por medio de una pequeña encuesta.

Palabras Clave: Ingeniería de Servidores, Docencia, Motivación del estudiante

Abstract. Ingeniería de Servidores (English: Server Engineering) is a Computer Engineering subject taught on the third year of Computer Engineering degree. In this paper, the progress of the subject is analyzed from a student's point of view. Methodology and the evaluation system are described and examined in order to provide a feedback of weakness and strengths observed so much for the author as another students of different groups whose opinion is reflected through a survey.

Keywords: Server Engineering, Teaching, Student motivation

1 Introdución

Ingeniería de Servidores (en adelante, ISE) es una asignatura obligatoria de formación específica de rama que se imparte en el quinto semestre (tercer curso) del Grado en Ingeniería Informática.

El temario teórico impartido en la asignatura se muestra a continuación:

1. Introducción a la Ingeniería de Servidores
2. Componentes hardware de un servidor
3. Monitorización de servicios y programas

4. Análisis comparativo de rendimiento (benchmarking)
5. Análisis operacional en servidores
6. Exposición de casos de estudio[1]

En cuanto al temario práctico, se distribuye de la siguiente manera:

1. Instalación virtualizada de SO en servidores
2. Instalación y configuración básica de servicios
3. Monitorización de la actividad de un servidor
4. Benchmarking de servidores
5. Optimización del rendimiento de servidores

Además, se imparten dos seminarios ("Administración y mantenimiento de Servidores" y "Elaboración de un pliego de condiciones técnicas"). Todo esto, junto a un Trabajo autónomo que se elabora durante el transcurso del semestre, configuran la estructura de la asignatura y permiten alcanzar los objetivos recogidos en la Guía Docente de la misma. (Disponible en la web de Grados de la UGR: http://grados.ugr.es/informatica/pages/infoacademica/guias-docentes)

La evaluación de las diferentes partes de la asignatura se realiza de la siguiente manera:

- En convocatoria ordinaria:
  - Teoría: ponderada al 40% de la nota final. La valoración de esta parte se determina por medio de un exámen teórico.
  - Práctica: ponderada al 30% de la nota final. Incluye la evaluación de las memorias de cada tema, los dos seminarios y un examen de prácticas.
  - Trabajo: ponderado al 30% de la nota final.
- En la convocatoria extraordinaria:
  - Teoría: pondera al 40% de la nota. Se valora por medio de un examen teoórico.
  - Trabajos a entregar: ponderan el 30% de la nota.
  - Examen de prácticas y seminarios: Ponderan al 30% de la nota.

En todas las partes es necesario obtener, al menos, un 4 sobre 10 para poder obtener una calificación final positiva.

A continuación, se analizará y valorará, en sus diferentes partes, el desarrollo de la asignatura narrando la experiencia del autor como estudiante así como la de otros compañeros durante el curso 2013/2014.

1 El Tema 6, por motivos de falta de tiempo no se ha podido impartir en todos los grupos.

2 Desarrollo de la asignatura

2.1 Teoría

¿Cómo obtener el rendimiento más alto con el coste más pequeño?. Esta es una de las cuestiones principales que desde ISE se trata de responder. Para ello, el primer tema de la asignatura trata de introducir y aclarar conceptos básicos sobre Sistemas Informáticos y sus diferentes clasificaciones, paralelismo, fundamentos de la Ingeniería de Servidores, medidas de rendimiento y comparación de prestaciones y la Ley de Amdahl. En realidad, estos conceptos no se introducen por primera vez en esta asignatura sino que se realiza un breve repaso y se toma como punto de partida el conocimiendo adquirido en asignaturas como Estructura de Computadores, Sistemas Concurrentes y Distribuídos y, especialemente, de Arquitectura de Computadores. Esta última asignatura se podría considerar como el punto de partida de ISE.

El tema de introducción resulta importante de cara al resto de la asignatura ya que permite asentar conocimientos que, si bien ya se suponen adquiridos en otras asignaturas, no se han estudiado con el enfoque o la profundidad necesaria para ISE.

A pesar de que existen conceptos como el de rendimiento o como la Ley de Amdahl, que se estudian en Arquitectura de Computadores, no es hasta ISE cuando esos conocimientos se asimilan definitivamente y el estudiante logra ver la utilidad de los mismos.

Una vez se han establecido los cimientos de la asignatura, se da el salto a la Monitorización. *¿Cómo medir el rendimiento de mi servidor?,* es la pregunta a la que da respuesta este tema que comienza hablando de monitores, tipos de monitores, funcionamiento de los mismos y sus atributos (exactitud, precisión, frecuencia de muestreo, sobrecarga que generan...) para saltar al uso de las herramientas que nos proporciona Unix y alternativas a estas como SarCheck (http://www.sarcheck.com/) .

Aunque no es condición indispensable para asimilar positivamente este tema, es importante que el alumnado haya superado las asignaturas de Fundamentos del Software y la de Sistemas Operativos, donde ya en su parte práctica se ha trabajado con la mayor parte de las herramientas de monitorización del Sistema Operativo además de hablar a más bajo nivel de los procesos del Sistema y su funcionamiento.

Realmente, la segunda parte del tema, que trata de la Monitorización a nivel de aplicaciones (profilers) es la que aporta nuevos conocimientos al estudiante que no se han impartido en asignaturas anteriores. Puede resultar llamativo que hayan tenido que transcurrir tres cursos en una carrera donde, en prácticamente en cada asignatura impartida hasta entonces, ha sido necesario programar, para que el estudiante tenga su primer contacto con los profilers. Si bien es cierto que en primero se utilizó Valgrind en una asignatura, no se explotó el verdadero potencial que tiene este programa y cabe preguntarse si sería necesario que los estudiantes del Grado se familiarizaran con este tipo de herramientas cuanto antes mejor, del mismo modo que se hace con los depuradores.

Una vez que sabemos qué es el rendimiento y cómo medirlo, llega el momento de preguntarnos: *¿Qué servidor tiene mejor rendimiento?*. La comparación de diferentes sistemas por medio de índices, benchmarks y un análisis de sus resultados suponen los conocimientos a adquirir en este tema en el que, además, se plantea acabar con los mitos de la existencia de un "índice perfecto" que permita compararlo todo. El tercer tema culmina con una pequeña introducción al diseño de experimentos para la comparación de servidores. Quizás, la mayor dificultad de este tema radica en el uso de la Estadística, necesaria para comprobar si las diferencias entre las mediciones realizadas a varios sistemas son o no significativas. Si bien el nivel requerido al estudiante se adquiere en la asignatura de Estadística que se imparte en el segundo semestre, suele ser común que los estudiantes en tercero sigan arrastrando asignaturas de primero o bien dejen estas para el final de la carrera.

Terminado el tercer tema, entramos en el que tema que, probablemente, da mayor sentido a la asignatura: El análisis operacional de servidores. Los conceptos que aquí se estudian son nuevos y no se han visto en ninguna otra asignatura: La abstracción respecto a los servidores, que son representados con diferentes modelos de Sistemas Informáticos según su comportamiento con un conjunto de variables que les afectan y permiten sus análisis gracias a las Leyes Operacionales que rigen a dichas variables. Es en este tema donde se estudia qué es la hipótesis de equilibro de flujo, la Ley de Little, Ley de la Utilización, Ley del flujo forzado y la Ley general del tiempo de respuesta. A continuación, hacen análisis asintóticos para conseguir descubrir los cuellos de botella y las limitaciones en el rendimiento. En definitiva, la pregunta que nos hacemos esta vez es *¿Cómo modelar el rendimiento y predecir la capacidad de mi servidor?*

Tras un tema que quizás termina siendo algo denso, el siguiente de la asignatura podría catalogarse de "cultura general" que todo Ingeniero Informático debería conocer: Componentes hardware de un servidor. En este tema, se estudia qué es una Placa Base y sus componentes. Es interesante señalar que el enfoque que se le dio en clase de Teoría no fue meramente una lección teórica formal sino que fue acompañado con ejemplos de modelos de componentes obtenidos directamente desde la web del fabricante. Dando al estudiante una perspectiva más realista sobre las prestaciones y el precio actual en el cual oscilan esos componentes, así como cuáles son los principales fabricantes de estos componentes. En definitiva, responder a la pregunta *¿Qué hardware es el más adecuado para mi servidor?*.

Finalmente, el último tema enlaza con el Trabajo de la asignatura. Algunos estudiantes han expuesto sus temas a compañeros de grupo con el fin de que estos puedan aprender sobre otras áreas que ellos no han trabajado. Por problemas de tiempo, en el grupo del autor no se pudo llegar a realizar estas exposiciones. Por lo que se ha realizado una pequeña encuesta en la que han participado cuatro estudiantes de este curso y del curso anterior:

- Pregunta 1: *¿Sobre qué hiciste tu trabajo de ISE?*
    - Estudiante 1: Balanceo de carga y planificación de servidores.
    - Estudiante 2: Hice el trabajo sobre servidores de alojamiento de archivos.
    - Estudiante 3: Sobre refrigeración de centros de datos. Donde se explicaban las diferentes formas de refrigeración actuales con sus ventajas e inconvenientes.

    - Estudiante 4: Clustering, aunque lo quería hacer sobre free cooling.
- Pregunta 2: *¿Crees que el trabajo te motivó para investigar y aprender sobre este tema?*
    - Estudiante 1: Si que tuvimos que investigar ya que no sabíamos apenas del tema (algo de planificación si, por Sistemas Operativos, pero muy básico). La motivación fue, sobretodo, hacer un buen trabajo y hacerlo bien.
    - Estudiante 2: Sí, ya que los servidores de alojamiento de archivos representan un papel fundamental para compartir información de manera cómoda. Creo que conocer los avances tecnológicos en este campo y los sistemas que se desarrollan es importante. Como por ejemplo el caso de Dropbox, que permite alojar un gran número de ficheros de forma rápida siguiendo el modelo de almacenamiento en la nube. A su vez también es importante conocer como afecta el marco legal a estos sistemas, ya que los desarrolladores de algunos de estos sistemas han llegado a recibir acusaciones por infracción de derechos de autor. Siendo uno de los casos más conocidos el de Megaupload.
    - Estudiante 3: El trabajo me resultó muy interesante y me motivo para investigar más sobre nuevas tecnologias para la refrigeracion.
    - Estudiante 4: No, para nada.
- Pregunta 3: (solo si la tercera pregunta tiene respuesta afirmativa)*¿En tu grupo se hicieron exposiciones?*
    - Estudiante 1: Sí, se hicieron.
    - Estudiante 2: Sí, se hicieron exposiciones.
    - Estudiante 3: Sí.
    - Estudiante 4: Sí
- Pregunta 4: *¿Expusiste tu trabajo en clase?*
    - Estudiante 1: No, no dio tiempo.
    - Estudiante 2: No, ya que mi clase estaba formada por un gran número de alumnos y no pudimos exponer todos.
    - Estudiante 3: Sí.
    - Estudiante 4: No.
- Pregunta 5: *¿Crees que las exposiciones de otros compañeros te ayudaron a aprender e interesarte sobre otros temas relacionados con la Ingeniería de Servidores?*
    - Estudiante 1: Siempre que expone un alumno hay riesgo de que la exposición no sea muy buena y el tema se haga pesado, pero si que aprendimos.
    - Estudiante 2: Sí, ya que hubo una variedad de tecnologías importante. Mediante los trabajos de exposición de los compañeros pude conocer aspectos interesantes de estas tecnologías relacionadas con la Ingeniería de Servidores que desconocía hasta el momento.
    - Estudiante 3: Había exposiciones de temas muy interesantes de los cuales no conocía nada sobre ellos, y me instaron a buscar mas información.
    - Estudiante 4: No fuí a la de otros compañeros. Todo lo relacionado con servidores me parece aburrido.

2.2 Práctica

En la parte práctica de la asignatura, al estudiante se le sitúa en el rol de un administrador de sistemas que debe configurar y mantener un servidor. Se trabaja tanto en Windows Server como en GNU/Linux (Ubuntu Server y CentOS

principalmente, aunque también se pueden utilizar otras distribuciones como openSUSE).

Como instalar tantos Sistemas Operativos con el único fin de probarlos no es una situación ideal, las prácticas se realizan virtualizando estos Sistemas Operativos mediante VMware o VirtualBox. De esto trata el primer tema, donde se introducen conceptos como Máquina Virtual o virtualización. Tras esto, se comienza la instalación de los diferentes Sistemas Operativos creando RAIDs y usando volúmenes lógicos cifrados. Finalmente, se prueba la disponibilidad del servidor frente a fallos del RAID. Es decir, se realiza una instalación y configuración habitual en un entorno real de trabajo.

Una vez tenemos instalados nuestros distintos servidores, toca instalar diferentes servidores, tales como ssh, ftp, apache... Tareas que se realizan en la segunda práctica.

La tercera práctica permite al estudiante probar diversas herramientas para monitorización como Ganglia, Nagios, Munin, Perfmon, Awstats, Cactis así como profilers, concretamente gprof y Valgrind. Esta práctica, si bien es útil de cara al futuro, termina resultando bastante pesada y larga por la gran cantidad de programas que hay que instalar y probar. Quizás sería recomendable centrar la atención en una cantidad menor de programas y usarlos más a fondo en vez de recorrer un amplio espectro de alternativas de forma superficial. De hecho, varios compañeros tuvieron problemas para entregar esta práctica y la siguiente debido a la carga de trabajo que requieren y a que su periodo de realización coincidía con el de prácticas de otras asignaturas de tercero que también requerían bastante tiempo. Finalmente, se instalan y prueban diferentes profilers.

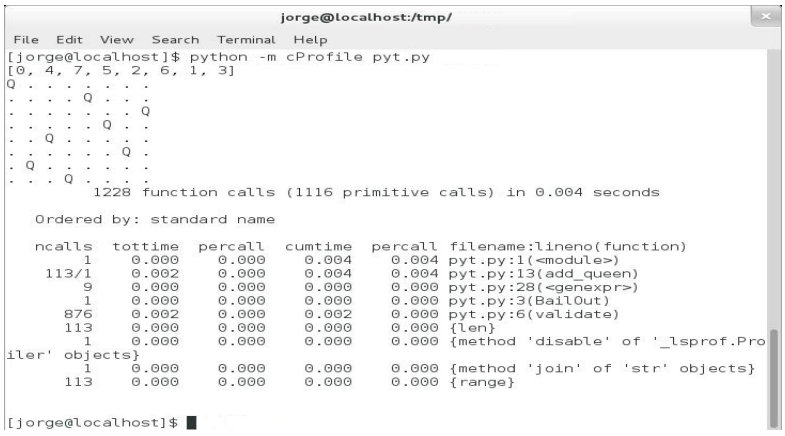

Figura1. Ejecución de la librería Cprofile (http://docs.python.org/2/library/profile.html) sobre una implementación del problema de las n-reinas como parte de un ejercicio de prácticas.

La cuarta práctica trata sobre benchmarks y test de estrés. Sigue una estructura similar a la anterior práctica: se prueban diferentes benchmarks tales como el Phoronix Suite, Jmeter, Siftsoftware Sandra y Aida64. Al final de la práctica, el estudiante puede programar su propio benchmark.

Tras dos prácticas que, como se ha mencionado anteriormente, resultan algo densas, la última práctica de la asignatura trata la optimización de los servidores para

mejorar sus prestaciones. Básicamente, consiste en mejorar el Servidor Web IIS, Apache, modificar parámetros del kernel de GNU/Linux y del registro de Windows y optimizar un servicio del sistema a elección del Estudiante.

2.3 Seminarios

Por último, durante las horas de prácticas se han impartido dos seminarios. El primero de ellos trata sobre Administración de Servidores y recorre diversos temas, empezando una introducción a las diferentes certificaciones profesionales que existen para continuar con una visión a las ofertas de empleo para administradores existentes hoy en día y terminando con buenas prácticas en la administración de servidores, ética, seguridad y una breve introducción a Windows PowerShell.

El segundo seminario introduce al estudiante en la legislación actual sobre pliegos. Ambos seminarios aportan conocimientos que todo estudiante del Grado debería conocer. Una vez más, ISE actúa como "cajón de sastre" donde se imparten contenidos que no necesariamente están totalmente relacionados con la asignatura (en alusión al segundo seminario).

2.4 Trabajo Grupal/Autónomo

El trabajo que el estudiante debe ir realizando a lo largo de la asignatura versa sobre la mejora de uno de los factores (disponibilidad, eficiencia energética, fiabilidad, mantenimiento, coste, escalabilidad, seguridad y extensibilidad), que contribuyen en la mejora del diseño o configuración de los servidores. El trabajo se divide en dos fases:

- **Primera fase**: Por equipos de hasta nueve personas. Se buscan y discuten sobre tecnologías y sistemas que permitan mejorar alguno de los factores anteriormente mencionados. Se fomenta en todo momento que se formen grupos grandes, ya que el número de miembros no influye en la calificación de esta fase. Pero los grupos con un gran número de integrantes tendrán ventaja a la hora de encontrar más soluciones y, por lo tanto, más posibilidades de obtener una mayor calificación. Desde mi punto de vista, esta política es acertada ya que así se consigue que cada estudiante, a nivel individual, obtenga información sobre un mayor número de soluciones que las que podría buscar de forma individual o en un grupo reducido. En concreto, en nuestro grupo de trabajo encontramos las siguientes soluciones:

Tabla 1. Soluciones aportadas por un grupo de trabajo para trabajar sobre los diferentes factores implicados en el diseño y/o configuración de servidores

Solución	Factores implicados
RAID	• Rendimiento • Disponibilidad

	• Coste
SAI	• Disponibilidad • Fiabilidad • Seguridad • Coste
Arquitectura ARM	• Eficiencia energética
Bioclimatización	• Eficiencia energética • Coste
Virtualización por hardware	• Coste • Escalabilidad • Disponibilidad
Proxy inverso	• Seguridad
Iptables	• Seguridad
VPN	• Seguridad
Rack TS IT	• Eficiencia • Matenimiento
Balanceo de carga	• Rendimiento
Cifrado de datos	• Seguridad • Disponibilidad
Cloud Hosting	• Rendimiento • Disponibilidad • Mantenimiento
Herramientas de administración y monitorización	• Rendimiento • Disponibilidad • Mantenimiento
Copias de seguridad	• Seguridad • Disponbilidad
Granjas de servidores	• Rendimiento • Disponibilidad
Blade (*Servidor*)	• Eficiencia energética
Algoritmos genéticos aplicados a servidores	• Rendimiento • Eficacia • Coste
Refrigeración líquida de servidores	• Coste • Disponibilidad

Todos estas soluciones fueron sometidas a una puesta en común entre los miembros del grupo. Por lo que, de una forma simple, cada estudiante ha podido adquirir unas nociones sobre la existencia de estas soluciones y en qué consiste cada una de ellas. De una de estas soluciones, cada miembro el grupo desarrollará la parte autónoma del trabajo. No existen dos trabajos sobre una misma solución. De manera que gran parte de estas soluciones al final serán investigadas y desarrolladas por los estudiantes a lo largo de la asignatura.

- **Segunda fase**: Como se ha mencionado anteriormente, cada miembro del grupo realiza un trabajo sobre una de las soluciones aportadas por el grupo. El objetivo es presentar tres documentos:
  - Memoria: De hasta doce páginas, donde se plasma todo lo que se ha investigado acerca de la solución.
  - Presentación: De cara a una posible exposición en clase.
  - Test: Preguntas tipo test sobre la presentación realizada.

En el caso del autor, el trabajo desarrollado fue sobre Arquitecturas de microprocesadores con el objetivo de conseguir eficiencia energética. Durante el mismo, se investigó sobre la demanda actual del mercado de procesadores de este tipo para pasar a continuación a un análisis de la arquitectura RISC y sus ventajas frente CISC en cuanto a consumo energético se refiere. Comparando ARM y su evolución histórica con la alternativa de Intel y sus procesadores Atom [1] [2]. Posteriormente, se realizó una comparativa con diferentes benchmarks y tests de estrés sobre ARM, Atom y modelos de otros fabricantes [3] y, finalmente, se analizaron las perspectivas de futuro de los procesadores de bajo consumo. Una vez terminado el trabajo, fue enviado a cuatro estudiantes al azar para que lo revisaran y evaluaran en base a una plantilla establecida por los profesores. La identidad del autor del trabajo nunca era conocida por la de los revisores, para asegurar la mayor imparcialidad posible. Del mismo modo, el autor puede ver las correcciones realizadas en su trabajo pero no puede ver quién se las hizo. Un estudiante corrige un total de tres trabajos. Con este Sistema, los trabajos se cruzan y cada estudiante, además de su propio trabajo, aprende sobre lo que sus compañeros de grupo han hecho y se toma en serio la lectura y la asimilación de contenidos de sus compañeros, puesto que luego deberá evaluarlos.

Finalmente, y en los grupos donde ha sido posible (porque no haya faltado tiempo de clase en dichos grupos), algunos trabajos han sido expuestos. Si analizamos con detalle este sistema. Podemos ver cómo cada estudiante, a lo largo de la asignatura:

- Ha buscado e investigado, al menos de forma superficial, sobre un gran número de diferentes soluciones para el diseño, configuración y mantenimiento de servidores.
- Ha aprendido de una manera completa acerca de algunas soluciones en concreto, expuestas por sus compañeros en clase.
- Conoce en profundidad tres soluciones, ya que ha tenido acceso y ha podido evaluar el trabajo de tres compañeros.
- Además, puede ponerse tanto en la piel de un profesor que debe evaluar trabajos y, a su vez, obtener retroalimentación con las opiniones de sus compañeros sobre los puntos fuertes y los puntos débiles de su trabajo. Propiciando la adquisición de competencias tranversales establecidas en el Grado. [5]

3 Conclusiones

ISE es una asignatura donde se combinan, de forma bastante acertada, conocimientos teóricos sobre la Ingeniería de Servidores y conocimientos prácticos sobre configuración y administración de los mismos con un enfoque práctico simulando la vida real de muchos Ingenieros que tienen que lidiar con la administración de sistemas. Como parte negativa, la práctica tercera y cuarta han

requerido demasiado tiempo. El hecho de que, con un solo ejercicio sin realizar la memoria pueda calificarse de forma negativa, puede dar lugar a situaciones injustas, por lo que habría que plantearse una relajación de ese criterio de evaluación.

La parte teórica de la asignatura quizás podría reconfigurarse con el fin de menguar la curva de aprendizaje, ya que el penúltimo y especialmente el último tema contienen una gran cantidad de conceptos nuevos que deben ser asimilados en poco tiempo.

El trabajo incentiva a investigar sobre temas que resultan novedosos, y la posibilidad de acceder y aprovechar los trabajos de los compañeros es una forma de conseguir que el estudiante aprenda más y empleando menos tiempo del que necesitaría, por sí mismo, para investigar sobre cada una de las diferentes soluciones.

Por último, tanto los seminarios como el tema sobre componentes del hardware pueden servir como forma de motivación, ya que "rompen" la rutina a la vez que se imparten conocimientos necesarios para el estudiante. Este sistema podría ser exportado a otras asignaturas, consiguiendo así enriquecer la formación del alumnado.

Finalmente, cuando ha llegado el momento de preparar el examen de teoría, el estudiante tenía a su disposición, como material de apoyo, problemas cuya solución venía dada en la relación de ejercicios. Hay que tener en cuenta que, en época de exámenes, el estudiante se encuentra "solo" ante los apuntes de clase. Introducir soluciones en las relaciones de problemas permiten al estudiante medir su progreso y no navegar en un mar de dudas ("*¿estoy aprendiendo correctamente o me estoy equivocando por completo?*") y sirve como motivación ("*lo que estoy aprendiendo lo aplico de forma correcta, merece la pena que siga preparándome la asignatura*", "*me he equivocado, pero ahora sé cómo tengo que hacerlo para que esté bien*").

Agradecimientos

A Héctor Emilio Pomares Cintas y a Eloi Montilla i Busquets por la orientación y ayuda prestada a la hora de redactar este artículo.

A Isaac Morely Rodríguez, Natalia Romero Parra, Emilio Nieto Morilla, Germán Iglesias Padial y Fernando Palacios Lopez por su participación en este artículo aportando su visión y opiniones sobre ISE.

Referencias

1. Stokes, J.: RISC vs. CISC in the mobile era. arstechnica. http://arstechnica.com/gadgets/2008/05/risc-vs-cisc-mobile-era/ (2008)
2. Bichenn, S.: ARM to Intel: it's not just about the transistor. hexus.net. http://hexus.net/business/features/components/30305-arm-intel-its-just-transistor/ (2011)
3. Chak, J.C.: Benchmarks i mètriques d'avaluació de plataformes encastades multimèdia. Diposit de la Rercerca de Catalunya. Escola Tècnica Superior d'Enginyeria Secció d'Enginyeria Informàtica. Universitat Autònoma de Barcelona, http://www.recercat.net/bitstream/handle/2072/14087/PFC+Joan+Carles+Chak+Ma.pdf?sequence=1 (2008)
4. Schönborn, T.: SoC Shootout: x86 vs. ARM. NotebookCheck.net http://www.notebookcheck.net/SoC-Shootout-x86-vs-ARM.99496.0.html (2013)
5. Ministerio de Educación, Cultura y Deporte. Memoria del Grado en Ingeniería Informática. P6. http://grados.ugr.es/informatica/pages/infoacademica/archivos/verificaingenieriainformatica/%21

Concepción de un sistema de iluminación inteligente en Smart Cities como Proyecto Fin de Carrera de Ingeniería Informática

Víctor Martín, Miguel Damas, Jesús González, Héctor Pomares

[1] Departamento de Arquitectura y Tecnología de Computadores
Granada, España
{vicmr, mdamas, jesusgonzalez, hector }@ugr.es

Resumen. Este trabajo presenta el proyecto fin de carrera "Iluminación inteligente en las ciudades del futuro" defendido en Septiembre de 2013 en la titulación Ingeniería Informática de la Universidad de Granada, un proyecto distintivo y que ofrece al alumno una formación y una madurez añadida. Durante el desarrollo de este proyecto se presenta un sistema de gestión de farolas inteligentes basado en lámparas LED, diseñadas para facilitar su despliegue en las instalaciones existentes. El enfoque propuesto, basado en tecnologías de comunicación inalámbrica, permite reducir al mínimo el coste de la inversión de los sistemas de cable tradicionales. Gracias a la realización de este proyecto, se consigue aprender tecnologías y sistemas que no son tratados en la carrera, permitiendo al alumnado conocer uno de los temas más demandados en la actualidad.

Palabras Clave: Ciudades inteligentes, Alumbrado Inteligente, Ingeniería Informática.

Abstract. This paper presents the final project "Intelligent lighting in the cities of the future" presented in September 2013 at the Computer Engineering degree at the University of Granada, a distinctive project that offers students an added training and maturity. During the development of this project a management system based on smart streetlights LED lamps designed for easy deployment in existing facilities is presented. The proposed approach, based on wireless communication technologies, can minimize the investment cost of traditional cable systems. Thanks to this project, one gets to learn technologies and systems that are not treated in the career, allowing the students to know one of the most popular topics today.

Keywords: Smart City, Smart Lighting, Computer Engineering.

1 Introducción

En las antiguas titulaciones de Ingeniería Informática existía una carencia en cuanto a que los temas de actualidad no se estudiaban, aunque con la implantación del nuevo Grado se han propuesto asignaturas que estaban más relacionadas, como puede ser la asignatura “Sistemas Empotrados”. La mayoría de las veces, los estudiantes eligen proyectos fin de carrera relacionados con temas conocidos y anteriormente tratados que realmente no aportan nada a la formación del alumno. Se debería de proponer la realización de proyectos fin de carrera orientados a temas de actualidad y que, en cierto modo, no se hayan tratado específicamente en la carrera, con el fin de dotar al estudiante de una competencia extra y acercarlo un poco más a los temas que se va a encontrar en su futuro profesional. Sin ir más lejos, este proyecto fin de carrera ha supuesto un gran aporte académico, ya que se ha tratado con tecnologías novedosas y hardware (tema bastante olvidado en las antiguas titulaciones de Ingeniería Informática). Por este motivo, a continuación el primer autor de este trabajo presenta un proyecto fin de carrera que esta fuera de lo normalmente realizado.

La urbanización ha aumentado encarecidamente en los últimos años, y las previsiones muestran que la migración de la población hacia las zonas urbanas no va a decrecer [1]. Encontrar ciudades con decenas de millones de habitantes ya no es algo extraño. Esta concentración de población en las ciudades plantea numerosos desafíos en términos de gobernabilidad de la ciudad y la vida de las personas. Como consecuencia, soluciones "más inteligentes" son necesarias para mejorar, día a día, las ciudades. Las ciudades inteligentes o Smart Cities han reclamado un lugar central en las agendas de innovación de los gobiernos, organizaciones de investigación y proveedores de tecnología, lo que plantea desafíos únicos y difíciles. Desde la perspectiva de la investigación, las ciudades inteligentes son inherentemente interdisciplinares: requieren la investigación y la cooperación de varias disciplinas, que abarcan desde la economía a las ciencias sociales, y desde la política a la gestión de infraestructuras. Por tanto, el paradigma de Smart City se puede considerar como un sistema distribuido en el que las diferentes fuentes de información proporcionan datos a un conjunto de aplicaciones que los utilizan para la elaboración de respuestas a nivel estratégico y táctico.

El alumbrado público en España representa el 10% del consumo total de energía en la iluminación y se sitúa en 116 kilovatios por año y habitante [2], en comparación con los 91 kW o 43 kW en Francia y Alemania, respectivamente [3]. Las últimas cifras disponibles del Ministerio de Industria, Turismo y Comercio suponen un consumo de energía eléctrica de 3.630 GWh / año para el conjunto de España. Vale la pena señalar que, en total, había 4.8 millones de puntos de luz estimados en el año 2010, y que un tercio del alumbrado público se basa en tecnologías anticuadas e ineficientes. La iluminación pública tiene el mayor impacto en el consumo de energía de un municipio, y puede representar hasta el 54% del consumo total de energía y el 61% del consumo eléctrico de las instalaciones municipales. La importancia de las instalaciones de alumbrado público es tal que, en algunos municipios, este hecho ahora representa hasta el 80% de la electricidad consumida y hasta el 60% del presupuesto de consumo de energía del municipio.

Luego, teniendo en cuenta las anteriores consideraciones, la innovación en los sistemas de gestión eficientes de alumbrado público es una necesidad, de ahí que en este trabajo se proponga una solución escalable, integral y eficiente que proporcione la iluminación sólo cuando sea necesaria (según las condiciones meteorológicas instantáneas o la presencia de personas y vehículos), con el objetivo de reducir los costes relacionados en los municipios, y así ayudar a la recuperación económica. Además, el nivel de iluminación será convenientemente regulado, evitando de esta manera el deslumbramiento, y el coste de la instalación será reducido al mínimo utilizando las comunicaciones inalámbricas. Todas estas mejoras tienen que ser consideradas a fin de lograr una reducción significativa del consumo de energía en la iluminación y por lo tanto contribuir al desarrollo sostenible. Además, es un sistema autónomo, donde se presenta un enfoque integral que garantiza la estandarización, la interoperabilidad y la adaptación a las necesidades y escenarios. Este es un aspecto no funcional pero distintivo respecto a otras soluciones antes mencionadas, que están diseñadas específicamente para un escenario concreto, y con pocas posibilidades de migración a otro entorno.

El resto del articulo está organizado de la siguiente forma: en la sección 2 se introduce el concepto de Smart City y se describe el servicio en el que está basado este artículo; en la sección 3 se analiza el diseño adoptado para el desarrollo de una plataforma de alumbrado inteligente; en la sección 4 se presenta la implementación realizada para dicho sistema y en la sección 5 se muestran los resultados obtenidos. Finalmente se termina el artículo presentando las conclusiones generales y la bibliografía utilizada.

2 Smart City

En las antiguas titulaciones de Ingeniería Informática el concepto de Smart City pasaba totalmente inadvertido pero en realidad es uno de los temas de mayor impacto en la actualidad, por lo que tener conocimiento de este nuevo paradigma proporciona un aporte extra a la formación del alumno. A continuación, se presenta dicho concepto.

Más de la mitad de la población del planeta vive en ciudades. Se espera que la cifra supere el sesenta por ciento en 2050. La calidad de vida de miles de millones de personas dependerá de su capacidad para innovar, y ofrecer mejores servicios y oportunidades a sus habitantes (por ejemplo, ciudades más limpias, más cómodas, más verdes, y en definitiva más inteligentes).

El modelo actual de ciudad está en crisis. En general son complejas, contaminantes y caras. Las nuevas tecnologías ofrecen la oportunidad de revolucionar su gestión y sus procesos y con ellos mejorar la vida de sus ciudadanos, e incluso la salud del planeta. Las ciudades consumen tres cuartas partes de la energía que se produce en el mundo y son responsables del 80% del dióxido de carbono que se emite a la atmósfera. Mejorar la eficiencia de las urbes es, por lo tanto, pieza clave en la lucha contra el cambio

climático. Hacerlo exige un esfuerzo multidisciplinar, que conecte todos sus elementos (personas, edificios, vehículos e infraestructuras), y la tecnología para hacerlo realidad ya existe. A continuación, se muestran una serie de ejemplos donde se adopta el concepto de Smart City [4].

• Exprimir la energía

La mayor parte de la energía que se produce y no se consume al momento se pierde. Una red eléctrica inteligente (Smart Grid) puede ayudar a gestionar su distribución para que llegue a los lugares en los que hace falta solo cuando ésta se necesite. La mejor manera de aprovechar la energía, especialmente cuando proviene de fuentes renovables, es producirla cuando las condiciones son óptimas, almacenarla y después consumirla cuando y donde sea necesario.

• Energía móvil

Las baterías de los coches, especialmente los eléctricos, permiten acumular el excedente de energía que se produce, e introducirla en el sistema en el momento en que se necesita. Una tecnología así permitiría exprimir al máximo las fuentes renovables, y reducir el trabajo de las centrales de combustibles fósiles. La mayoría de las marcas de automóviles ya disponen de vehículos con esta capacidad.

• Aparcamientos en red

Plazas de aparcamiento que pueden saber cuándo están libres, y actualizar su estado para que cualquiera pueda encontrarlas gracias a su Smartphone es otro de los servicios que ofrecen las Smart Cities. Por ejemplo, la empresa catalana Urbiotica ya ha instalado un sistema así en Niza (Francia).

• Jardines que se riegan solos

El riego de parques y jardines se puede automatizar con la instalación de sensores de humedad y temperatura, para que solo rieguen cuando sea necesario. Los mismos equipos pueden, además, avisar de posibles enfermedades en las plantas, y cómo resolverlas. Gracias a este servicio, las ciudades maximizan el ahorro de agua y el mantenimiento de la misma.

• Recogida inteligente de residuos

Otro servicio interesante que pueden ofrecer las Smart Cities es la recogida inteligente de residuos. Por ejemplo, la empresa Urbiotica ha instalado en Sant Cugat del Vallés un sistema que diseña rutas para que los camiones de la basura sólo recojan los contenedores que están llenos o casi llenos. Así consiguen que se ahorre en combustible y viajes, además de reducir la contaminación acústica de la ciudad.

• Patrullas dinámicas de la policía

Gracias a las Smart Cities, ciudades como Memphis (EE.UU.) ha reducido un 30 por ciento sus tasas de delincuencia grave gracias a un sistema de IBM que geo localiza los crímenes que se cometen en la ciudad, y diseña las rutas más eficientes para los cuerpos de seguridad.

• Sistema de alumbrado inteligente

Por último, se va a presentar el sistema de alumbrado inteligente en la Smart City [5], servicio en el que se centra este artículo. La idea es conseguir un ahorro energético optimizando el uso que se le da al alumbrado público. Para ello se necesita disponer de sensores que detecten la presencia de personas y que midan la cantidad de luminosidad, para así poder regular la intensidad de luz que emiten las farolas, es decir, conseguir de forma automática la cantidad de luz artificial justa, en función de la presencia y la cantidad de luz natural que tenemos en cada momento. En este sentido, la iluminación LED ofrece un mayor rendimiento, más horas de vida útil, consumo menor de energía y son menos contaminantes que las actuales farolas de vapor de sodio y vapor de mercurio usadas para iluminar.

3 Diseño de un sistema para el alumbrado inteligente

De nuevo y desde el punto de vista de un estudiante recién titulado, a pesar de la importancia de los elementos que se van a presentar a continuación, en gran parte de las titulaciones de Ingeniería Informática existía una carencia importante. Realmente, para un alumno es difícil aprender elementos de tan bajo nivel por sí mismo, pero por el contrario, una formación autodidacta proporciona una madurez al estudiante que le permitirá la resolución de problemas cada vez mayores por sus propios medios.

La base sobre la que se sustenta la plataforma de alumbrado inteligente está formada por los siguientes elementos:

• Dispositivo con microcontrolador que nos permite administrar y gestionar nuestro sistema. Se ha utilizado el Freescale de tercera generación MC13224 [6], el cual incorpora un procesador de baja potencia de 2.4 GHz de frecuencia, núcleo ARM7 de 32, aceleración hardware para las aplicaciones IEEE 802.15.4, temporizadores programables e interrupciones externas en tiempo real.

• Sistema Operativo instalado en los dispositivos empotrados, que nos facilita entre otras muchas cosas un protocolo de comunicación indispensable para la realización de esta tarea.

• Sensores que permiten conocer características del mundo exterior y dotar al sistema de inteligencia.

• Actuadores que permiten reflejar el resultado de la inteligencia del sistema.

El sistema operativo elegido para el desarrollo del sistema de alumbrado inteligente es Contiki, que es un sistema operativo multitarea compacto de código abierto, altamente portable para sistemas empotrados en red y redes de sensores inalámbricas [7]. Necesita de un kernel, librerías, un programa de carga y una serie de procesos para poder funcionar, y ofrece las siguientes características:

• Los programas pueden ser sustituidos mientras el dispositivo está en funcionamiento.
• La comunicación entre procesos siempre se realiza a través del kernel.
• El kernel no proporciona una capa de adaptación con el hardware sino que deja a las aplicaciones comunicarse directamente con el hardware.

Como se ve en la Figura 1 el sistema Contiki está dividido en dos módulos: el núcleo (core) y la aplicación (loaded program). La partición se realiza durante la compilación y es específica del mapa de memoria de cada plataforma. Normalmente, el núcleo está formado por el kernel, el cargador de la aplicación (program loader), el entorno de ejecución del lenguaje de programación de la aplicación, y una pila de comunicación con drivers para el hardware de comunicación.

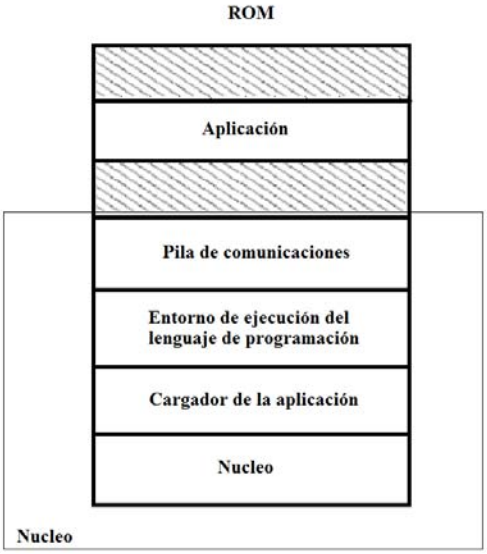

Figura 1. Partición entre el núcleo y programas cargados.

El kernel de Contiki consiste en un sistema de planificación de eventos de bajo consumo que reparte eventos entre los procesos en ejecución, y que periódicamente llama a una serie de procesos que controlan dichos eventos. El kernel soporta dos

tipos de eventos: asíncronos y síncronos. Los asíncronos son colocados en una cola y se envían a los procesos un poco más tarde. Los eventos síncronos son similares a los asíncronos pero hacen que el proceso objetivo sea inmediatamente programado para ser ejecutado.

Además de lo ya mencionado, el kernel proporciona un sistema polling o de control de eventos. Este puede verse como un sistema de prioridad de eventos que hace que sean programados entre los eventos asíncronos. Suele ser usado por procesos que se encuentran "cerca" del hardware para controlar de forma periódica el estado de los dispositivos de dicho hardware. Por ejemplo, si se usa un puerto serie se tendrá que utilizar dicho mecanismo para ver el flag de recepción del UART, para comprobar si ha llegado un carácter y en caso afirmativo procesarlo.

En resumen, se ha elegido Contiki porque es un Sistema Operativo para redes WSN relativamente ligero. Ofrece soporte para redes IPv6 de forma nativa (6LoWPAN) así como IPv4, CoAP y una pila propia para comunicaciones (entre otras muchas capacidades). El modelo de programación que ofrece es dirigido por eventos y las aplicaciones se programan en lenguaje C. Además, Contiki implementa Protothreads, que permiten la ejecución secuencial de código en sistemas dirigidos por eventos, permitiendo que la ejecución se bloquee y reanude según los eventos dirigidos, evitando la necesidad de implementar complicadas máquinas de estado, como sucede en sistemas operativos similares. También implementa un cargador de módulos dinámicos en tiempo de ejecución, lo que permite modificar el comportamiento de un nodo en tiempo de ejecución.

3.1 Arquitectura del sistema

Una vez se ha presentado la idea general del sistema, se está en disposición de presentar la arquitectura llevada a cabo. Se van a identificar y describir los diferentes componentes, poniendo nombres a algunos de ellos para evitar futuras confusiones y una mejor identificación.

3.1.1 Subsistema Web

Hoy en día, uno de los temas más demandados en la vida profesional de un informático son los sistemas web y, por lo tanto, cuando un alumno realiza la carrera de Ingeniería Informática espera recibir una formación adecuada sobre esta temática. En el nuevo plan de estudios del grado, han sido incluidas asignaturas como "Servidores Web de Altas Prestaciones" que permiten al alumno aprender a diseñar, instalar y configurar servidores de altas prestaciones, pero de nuevo, en las antiguas titulaciones de Ingeniería Informática solo se podía aprender haciendo cursos de libre configuración.

Concretamente, en este proyecto se ha diseñado un sistema que permite al usuario interactuar con cada una de las farolas registradas realizando acciones tales como: visualizar el estado de las farolas, cambiar la intensidad lumínica, cambiar el máximo de luz permitido, etc. Al tratarse de un Subsistema Web, es accesible desde cualquier ordenador o dispositivo móvil, con lo cual tenemos un gran abanico de posibilidades para coordinar nuestro sistema (ver figura 2).

3.1.2 Dispositivo Microcontrolador

Se trata de un dispositivo que va a desempeñar dos roles distintos dependiendo de si está conectado únicamente a los sensores y actuadores (en este caso actuará como dispositivo hoja), o va a ser conectado únicamente al pc (en este caso actuará como dispositivo coordinador).

Dispositivo coordinador. Este dispositivo será el encargado de gestionar y dirigir todos los dispositivos hoja. Su función principal es recibir los mensajes del subsistema web, por el puerto serie, y reenviar los mensajes al dispositivo hoja correspondiente mediante el protocolo de comunicación 6lowpan [8].

Dispositivo hoja. Se pretende que cada farola tenga acoplado uno de estos dispositivos para tener un control de lo que está midiendo en cada momento, pudiendo notificar estas medidas a otros dispositivos hoja para tener una total conexión entre ellos, a la vez que notifica al dispositivo coordinador para tener centralizados los datos de todos los sensores y qué este tome las decisiones que estime oportunas conforme a las reglas que tenga configuradas (ver figura 3).

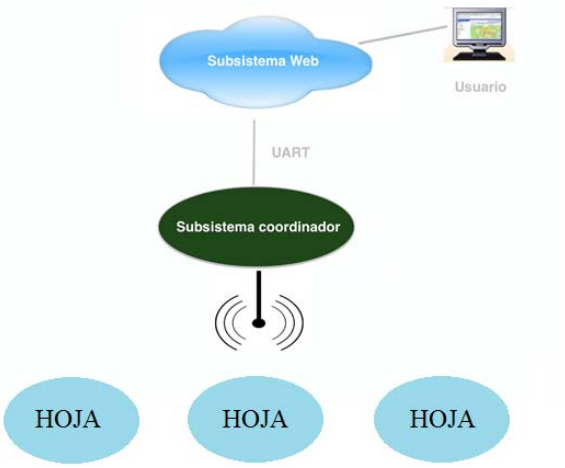

Figura 2. Arquitectura del sistema.

Este dispositivo está compuesto por un equipo de medida formado por tres componentes distintos. Uno de ellos contiene el sensor de luminosidad (LDR), el otro posee una pantalla de LED que actuará como farola, por último se dispone de un sensor de movimiento (PIR) para detectar los transeúntes. Tanto el sensor de luminosidad (LDR) como el sensor de movimiento (PIR) proporcionarán medidas que permitirán regular la intensidad de la pantalla de LED acorde a las circunstancias.

• **Sensor Luminosidad (LDR).** Gracias a este sensor se obtendrá la cantidad de luz ambiental cada vez que se necesite. Este sensor ayudara a encender la farola cuando la luz ambiente decaiga por debajo de un cierto umbral.

• **Sensor Movimiento.** Gracias a este sensor se detectará el movimiento de los transeúntes en las calles. Cada vez que el sensor proporcione un valor positivo indicando que hay movimiento en su zona, la farola se encenderá si la luz ambiental es baja, para que los transeúntes puedan pasear con total normalidad.

• **Placa de LEDs.** Permite simular el efecto de una farola, tanto el encendido y apagado como la atenuación de la misma.

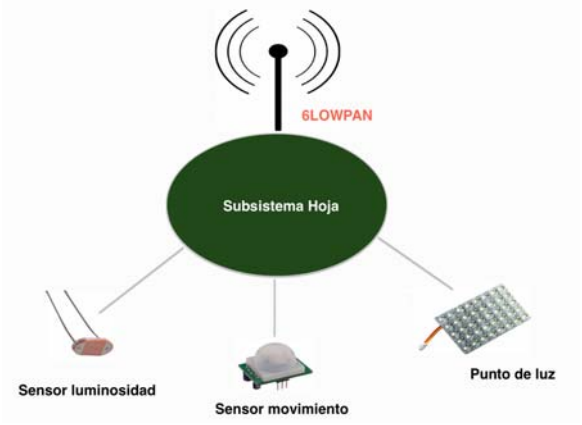

Figura 3. Subsistema hoja.

4 Implementación del sistema

A continuación, se muestra la implementación del sistema de control de luz inteligente basado en la arquitectura presentada en la sección anterior.

4.1 Subsistema Web

En primer lugar, el subsistema web está formado por un PC con apache conectado a internet y una mota coordinadora en un puerto serie. Es el encargado de mantener el contacto directo con el usuario. Bajo este subsistema, mostrado en la figura 4, podemos realizar cualquier acción sobre el alumbrado. Este subsistema mostrara las distintas farolas instaladas en una mapa, al pinchar sobre una determinada farola mostrara un menú donde se podrá actualizar la intensidad de esta farola o incluso cambiar el máximo de intensidad que tiene esa farola. Cuando el usuario indique la acción que quiere realizar sobre esta farola, el sistema prepara el mensaje formado por (acción, ip). En primer lugar consultará la IP de la farola seleccionada y en segundo lugar mandará por la UART la dupla (acción, ip) al dispositivo controlador.

Además de estas funcionalidades, el subsistema proporciona al usuario la posibilidad de registrar/eliminar nuevas farolas y de conocer el estado actual de las farolas que tiene registradas.

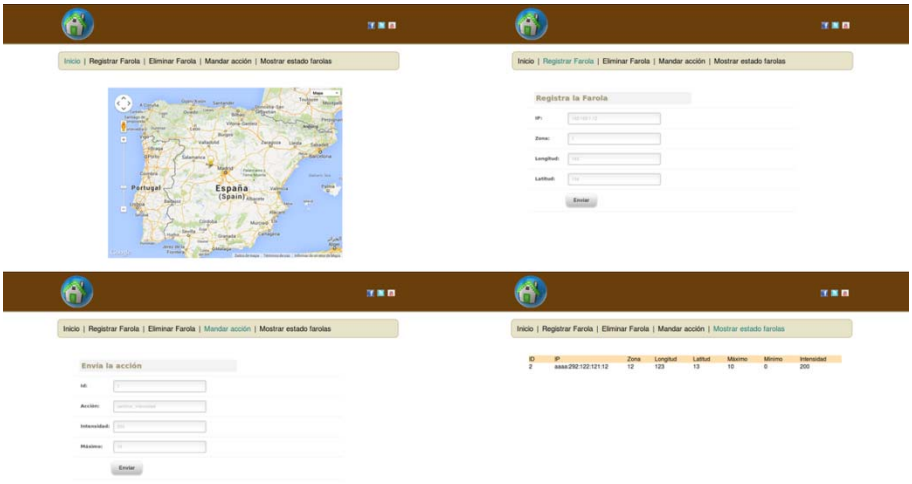

Figura 4. Interfaz gráfica subsistema web.

4.2 Subsistema Coordinador

A continuación se describirá el subsistema coordinador, el cual está formado por una mota conectada por el puerto serie a un PC con internet. Este subsistema, mostrado en la figura 5, actúa meramente de intermediario entre el subsistema web y el subsistema hoja. Las funciones principales a realizar por dicho subsistema son las siguientes:

• Cada vez que un dispositivo hoja se conecta manda un mensaje de bienvenida. El subsistema coordinador recibe un evento del tipo "tcpip_event" y mediante el manejador tcpip diseñado, guarda su dirección ip, para tener un control total de los dispositivos hoja que tiene a su alcance.

• Por otra parte este subsistema está escuchando por la UART los mensajes enviados por los usuarios desde el subsistema web. Si un usuario ha decidido realizar alguna acción sobre la farola, enviará el mensaje (acción, ip), y en tal caso, el subsistema coordinador reenviara el mensaje (acción) a dicha farola llamando a la función send_packet.

```
while(1) {
  PROCESS_YIELD();
  if(ev == tcpip_event) { //espera a la recepción de un evento
   tcpip_handler(); //llama al manejador de eventos
  }
  if(etimer_expired(&periodic)) { //Si el temporizador a expirado
   etimer_reset(&periodic); //resetea el temporizador
   ctimer_set(&backoff_timer, SEND_TIME, send_packet, NULL);  //Llama a la función send_packet
  }
 }
```

Figura 5. Implementación subsistema coordinador.

4.3 Subsistema hoja

Por último, el circuito realizado para el nodo hoja del sistema de alumbrado inteligente se puede ver en la figura 6, donde se aprecia la placa de led encargada de proporcionar luz eficiente, el sensor de movimiento encargado de avisar al sistema en el caso de detectar transeúntes y el detector de luminosidad encargado de proporcionar al sistema la cantidad de luz ambiental.

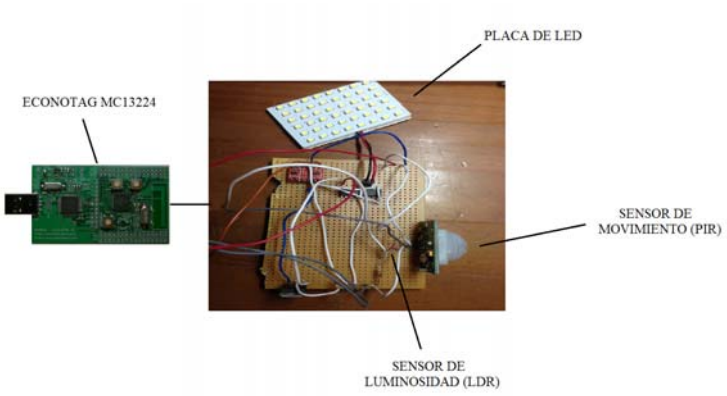

Figura 6. Nodo hoja del sistema de alumbrado inteligente.

Este subsistema es el encargado de interaccionar directamente con la farola (ajustándole su intensidad) en función de los datos proporcionados por sus sensores.

Como podemos observar en la figura 7, el subsistema hoja inicializa los parámetros necesarios para poder utilizar los sensores y recoge los resultados cada cierto tiempo. En el caso de que el sensor de movimiento dé una respuesta afirmativa la luz se encenderá, y si es el sensor de luminosidad el que avisa de que la cantidad lumínica ha variado se ajustará la intensidad lumínica de la farola en función del valor proporcionado. Por otro lado este subsistema escuchará cada cierto tiempo los mensajes recibidos vía 6lowpan y en el caso de que el usuario haya indicado alguna acción, realizará las acciones pertinentes.

```
PROCESS_BEGIN();
PROCESS_PAUSE();
Inicializcacion_adc(); //Inicialización del controlador analógico digital
Inicializacion_pwm(); //Inicialización del pwm
ctimer_set(&c, MY_TIMEOUT, Recogida_Resultados, NULL); //Recoger resultados de sensores cada MY_TIMEOUT
PROCESS_END();
```

Figura 7. Implementación subsistema hoja.

4.3.1 Modulación por ancho de pulso (PWM)

Una de las tareas más importantes de este trabajo ha sido conseguir diferentes intensidades de luz ya que un led solo tiene dos funciones: cuando se supera un determinado voltaje el led se enciende, mientras que si el voltaje baja de ese umbral el led permanece apagado. Para el funcionamiento de la plataforma de alumbrado inteligente, era estrictamente necesario conseguir regular el nivel de intensidad y se ha conseguido gracias a la dimerización de los leds mediante la modulación por ancho de pulsos (PWM) a nivel de software ya que el microcontrolador MC133324 de Freescale no tenía esta característica implementada por hardware.

La modulación por ancho de pulso (PWM) (pulse width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo (D) de una señal periódica (una sinusoidal o cuadrada), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga (ver figura 8).

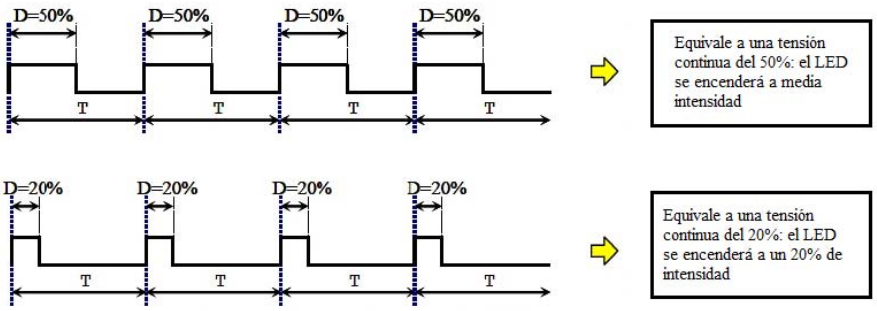

Figura 8. Modulación por ancho de pulso PWM.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

D : es el ciclo de trabajo.
τ : es el tiempo en que la función es positiva (ancho del pulso).
T : es el período de la función.

$$D = \frac{\tau}{T}$$

En concreto, en el subsistema hoja se ha definido un PWM con 5 niveles de luz, de tal manera que un LED pueda regular su intensidad en los siguientes rangos: 0%. 25%, 50%,75%,100%. De esta forma, dependiendo del nivel de luminosidad proporcionado por el sensor LDR, se decidirá qué porcentaje es el más adecuado y se generará el tren de impulsos correspondiente.

5 Resultados

En este apartado se van a exponer todas las pruebas realizadas para comprobar el correcto funcionamiento de los diferentes módulos que componen el sistema. Se van a proponer diferentes test sobre los posibles casos de uso más importantes que se pueden dar en el sistema.

En esta primera prueba se va a intentar simular el paso de una persona. Según se ha diseñado, la farola puede encenderse o no encenderse, dependiendo de los siguientes casos:

- Si hay mucha luz y se detecta movimiento -> no hace falta encender la farola para que vean las personas, ya que con la luz ambiental disponible es suficiente.
- Si hay poca luz y se detecta movimiento -> se enciende la farola para permitir a los transeúntes pasear con total normalidad.

Podemos comprobar en la figura 9 que en estas dos situaciones sucede lo correcto. En la foto de la izquierda la luz ambiental es relativamente baja, por lo que es conveniente encender la farola en caso de movimiento. Por otro lado, en la foto de la derecha, hay gran cantidad de luz, y aunque el detector de movimiento informe de que hay personas andando, no es necesario encender la farola.

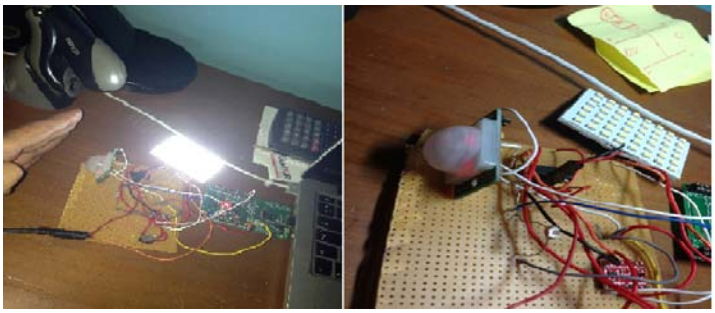

Figura 9. Prueba detección de movimiento.

En esta segunda prueba se va a simular el detector de luminosidad instalado en el sistema. Según se ha diseñado, la farola debe encenderse a una determinada intensidad dependiendo de la cantidad de luz ambiental que se tenga.

- Si hay mucha luz -> No hace falta encender la farola.
- Si hay luz media -> Es necesario encender la farola a un 50% de su intensidad total para así conseguir el óptimo de luz ambiental.
- Si hay poca luz -> En este caso, la farola se enciende al 100% de su intensidad ya que se necesita contrarrestar la baja cantidad de luz ambiental.

Podemos comprobar en la figura 10 que en estas tres situaciones sucede lo correcto. En la foto de la izquierda se encuentra el caso de uso donde la luz ambiental es muy baja y la farola se enciende al 100%, en la foto central la farola se enciende al 50% de su intensidad ya que existe una luz ambiental media, y por último en la foto de la derecha la farola no se enciende porque la cantidad de luz ambiental es muy alta.

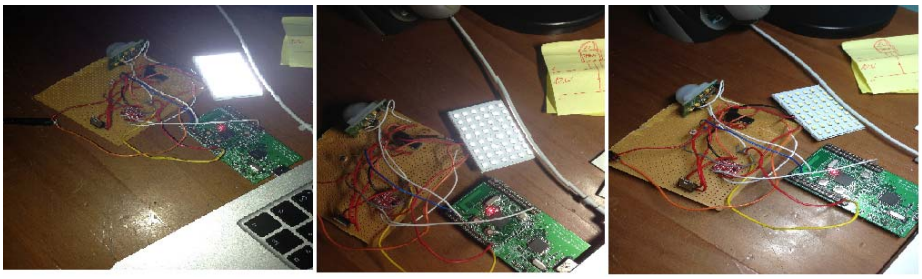

Figura 10. Prueba detección de luminosidad.

Una vez terminado este sistema se puede calcular cuánto se podría llegar a ahorrar en un caso práctico. Supongamos que tenemos una vía muy poco transitada a la entrada de una población en la que hay instaladas 50 luminarias LED inteligentes de 50 vatios de potencia, que han sustituido a los anteriores modelos de 150 vatios de sodio. Si no anda nadie en los alrededores su consumo baja en un 70%, hasta situarse en unos 15 vatios de potencia que aportan una uniformidad y luz de fondo más que suficiente. Si se detecta presencia suben a un 100% pasando a consumir unos 50 vatios. Si en el trayecto no hay presencia durante el 80% del tiempo, el ahorro energético obtenido es superior al 85%, ya que se pasa de tener un gasto de unos 54000 vatios al año a 8030 vatios, lo que supone un gran ahorro si se instalara en grandes ciudades como Madrid o Barcelona.

Una vez visto el resultado de las pruebas experimentales, el proyecto fin de carrera puede parecer bastante básico, pero detrás de todos estos ensayos hay una gran labor de investigación y aprendizaje para el estudiante que acaba de terminar su formación académica en una titulación en la que no se han tratado los temas con los que ha tenido que lidiar, y solamente la asignatura de Redes de Computadores ha ayudado en temas de comunicación.

6 Conclusiones

El objetivo de este documento es la presentación del trabajo fin de carrera “Iluminación inteligente en las ciudades del futuro” en la titulación de Ingeniería Informática de la Universidad de Granada.

El trabajo defendido en Septiembre de 2013 presenta un sistema de alumbrado público inteligente, con alto grado de adaptabilidad y facilidad de instalación. El sistema se aprovecha de una estrategia de control centralizada y el uso de elementos de conexión inalámbrica para lograr una mayor eficiencia energética. El sistema propuesto puede acomodarse a las diferentes necesidades de cada municipio asegurando la escalabilidad, la interoperabilidad y la accesibilidad (en el sentido de que el sistema es accesible desde múltiples plataformas: teléfono móvil, PC, Tablet).

Bajo la opinión del primer autor, el plan de estudios de las antiguas titulaciones estaba muy desfasado y lejos de lo que realmente se está viviendo actualmente en el mundo de la Informática. Sin embargo, con la adopción del nuevo plan de estudios del Grado de Ingeniería Informática se han corregido la mayoría de estas carencias ofreciendo al alumnado asignaturas mucho más cercanas a lo demandado en al ámbito profesional.

Referencias

1. World Urbanization Prospects, United Nations, Dept. of Economic and Social Affairs, Population Division, 2011 revision, 2012.
2. Instituto para la Diversificación y Ahorro de la Energía (IDAE). Disponible online: http://www.idae.es/index.php/id.644/relmenu.355/lang.uk/mod.pags/mem.detalle (accedido el 12 Marzo 2014).
3. Lighting of Streets and Buildings in Cities. Disponible online: http://energythic.com/view.php?node=406 (accedido el 12 Marzo 2014).
4. G. Piro, I. Cianci, L.A. Grieco, G. Boggia, P. Camarda. Information centric services in Smart Cities, JOURNAL OF SYSTEMS AND SOFTWARE Volumen: 88 Páginas: 169-188 Fecha de publicación: FEB 2014.
5. Miguel Castro, Antonio J. Jara and Antonio F. G. Skarmeta. Smart Lighting solutions for Smart Cities, IEEE 27TH INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS (WAINA) Páginas: 1374-1379 Fecha de publicación: 2013.
6. Reference Manual MC1322x, Document number: 1322xRM; Rev.0.0; 08/2008.
7. Dunkels, A; Grönvall, Björn; Voigt, Thiemo. Contiki- a Lightweight and Flexible operating System for Tiny Networked Sensors. Swedish Institute of Computer Science, PROCEEDINGS - CONFERENCE ON LOCAL COMPUTER NETWORKS Páginas: 455-462 Fecha de publicación: 2004.
8. Guido Moritz, Frank Golatowski, Christian Lerche, and Dirk Timmermann. 6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach, NTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS Volumen: 25 Número: 9 Número especial: SI Páginas: 1189-1212 Fecha de publicación: SEP 2012.

Fundamentos de Informática: exposición y utilidad desde la perspectiva de un alumno

Miguel Hoyo García

Grado en Ingeniería de Tecnologías de Telecomunicación. E.T.S de Ing. Informática y de Telecomunicación, Universidad de Granada
Granada, España
miguelhg@me.com

Resumen. Fundamentos de Informática es una asignatura que se imparte en el Grado en Ingeniería de Tecnologías de Telecomunicación durante el primer semestre del primer curso del grado. En este artículo se analizará, desde la perspectiva de un alumno de cuarto curso, la evolución de la asignatura, así como su metodología, las aplicaciones y utilidad de dicha asignatura en años posteriores del grado. Además, se tratarán las fortalezas y debilidades de la asignatura y su método de evaluación.

Palabras Clave: Fundamentos de Informática punto vista estudiante .

Abstract. Fundamentos de Informática (English: Computers Basics) is a subject taught in the Telecomunications' Technologies Engineering Degree during the first semester of the grade. This article will analyse, from a forth year student point of view, the evolution (course) of the subject, as well as its methodology, applications and utility of this subject in the following years of the degree. In addition the strengths and weakness of the subject and the method of evaluation will be exposed.

Keywords: Computers Basics point of view student

1 Introdución

Fundamentos de Informática se trata de una asignatura básica impartida en el primer semestre del primer curso del Grado en Ingeniería de tecnologías de Telecomunicación.

El temario teórico que se impartió en la asignatura se distribuyó de la siguiente manera:

1. Introducción y conceptos básicos
2. Representación de la información en computadores
3. Estructura funcional de computadores
4. Elementos de programación

5. Fundamentos de Sistemas Operativos
6. Bases de datos

Por otro lado, el temario práctico fue el siguiente:

1. Uso del sistema Operativo
2. Herramientas informáticas con aplicación en Ingeniería
3. Funcionamiento a bajo nivel de un ordenador
4. Uso básico de un Sistema Gestor de Bases de Datos

Del mismo modo, se impartieron un total de tres seminarios: "Estructura y montaje de un PC", otro seminario sobre el funcionamiento de CODE-2 y por último, un seminario sobre software libre y software de código abierto. Además, se realizaron un total de 11 trabajos adicionales (llamadas en la evaluación como "actividades en casa") relacionados con la materia que se estaba impartiendo en clase.

Por último, el método de evaluación se realizó como se muestra a continuación:

- Evaluación contínua:

Table 1. Método de evaluación contínua.

Aspectos a Evaluar	Técnica de evaluación	Ponderación	Nota máxima	Nota mínima para aprobar
Seguimiento y asistencia	Asistencia a clase de teoría y actividades en casa	10%	1	0
Prácticas y seminarios	Evaluación del trabajo de las sesiones de laboratorio y de los seminarios	20%	2	1
Prueba final	Examen tipo test	20%	7	3,5
	Examen de ejercicios prácticos	50%		
Total		100%	10	5

- Prueba única final:

Table 2. Método de evaluación con una prueba única final.

Técnica de evaluación	Ponderación	Nota máxima	Nota mínima para aprobar
Prácticas y seminarios	20%	2	1
Examen de Test	20%	2	3,5
Examen de ejercicios prácticos	50%	5	
Examen teórico	10%	1	0
Total	100%	10	5

En este artículo se tratará y analizará el desarrollo de la asignatura bajo la experiencia del autor del artículo durante el curso 2010/2011.

2 Desarrollo de la asignatura

2.1 Teoría

El primer tema se trata de una introducción a conceptos básicos y necesarios para el desarrollo posterior de la asignatura. Conceptos tales como qué son los datos, qué es una codificación, unidades de información (bit y byte), las diferencias entre el soporte físico y el lógico o la estructura funcional de los computadores son estudiados en este primer tema. Además, se explica cómo hacer un análisis de las prestaciones de un computador, es decir, medidas como tasa de transferencia, velocidad del procesador (frecuencia del reloj e instrucciones por segundo), la memoria del sistema, tanto masiva como principal y el tiempo de acceso del computador a los periféricos se analizan en este tema introductorio. Del mismo modo, se detallan la diferencia entre programas e instrucciones y los conceptos lenguaje máquina, lenguaje de alto nivel y el concepto de traductor. Por otro lado, se realiza un estudio de los tipos de computadores existentes, que van desde un servidor hasta un supercomputador. Por último, se analizan los tipos de software de un computador (software de control, sistema operativo y software de aplicaciones), las distintas aplicaciones de la informática y los tipos de aplicaciones de ingeniería con sus distintos campos de aplicación.

Este primer tema es de gran importancia, pues es la primera toma de contacto con una asignatura sobre informática y se aprenden conceptos que la mayoría estudiábamos por primera vez y que servirán como base de conocimientos de la asignatura en sus posteriores temas.

Una vez se han estudiado los conceptos básicos de la asignatura, se hace un análisis de cómo es representada la información en los computadores en el segundo tema. En primer lugar se estudian los tipos de numeración usuales en informática, tales como el binario, donde se estudiarán sus distintos tipos de representación: coma flotante, complementos a 1, 2, 3… Además de operaciones aritméticas. También se analiza el código hexadecimal y la conversión entre decimal, hexadecimal y binario. Posteriormente, se analiza la representación de textos (SBCD, EBCDIC, ASCII y UNICODE), la representación de sonidos (conversión A/D, códecs…), la representación de imágenes (mapas de bits y sus formatos y mapas de vectores) y por último la representación de datos numéricos. También se estudian métodos de corrección de errores, dónde se introduce el concepto de bit de paridad y finalmente se expone en qué consiste la compresión de datos.

En el tercer tema se realiza un análisis de la estructura funcional de los computadores, es decir, se presenta un análisis de los distintos componentes de un computador, comenzando por las primitivas del nivel de lógica digital (biestables, registros, contadores, ALU, puestos E/S y buses) explicando el funcionamiento de cada elemento. Se continúa analizando el procesador y su funcionamiento, estudiando sus elementos internos, cómo se ejecutan las instrucciones y cómo se puede cambiar el orden secuencial de un programa. Seguidamente se analiza el sistema de memoria del computador, donde se diferencia la memoria interna y externa y se introducen conceptos básicos como bloques físicos, tiempo de acceso a memoria, tiempo de ciclo de memoria y ancho de banda. Estos conceptos serán necesarios para la comprensión de la jerarquía de memoria, que consiste en el análisis del rendimiento y la capacidad de los sistemas de memoria mediante tasas de aciertos y fallos, además del funcionamiento de estos sistemas. Así mismo, se explican los conceptos de memoria interna (caché y memoria principal) matizando la diferencia entre ellas y sus funciones además del funcionamiento de cada una, para posteriormente introducir los tipos de memorias externa (soportes magnéticos, discos ópticos y memorias flash USB) explicando cómo funciona cada tipo de memoria, de manera que se puedan comparar los rendimientos de ambos tipos de memorias. Por último se nos presentó cómo se interconecta todo el sistema, para ello se explica qué es un bus y su funcionamiento para finalizar con el concepto de bus del sistema.

El cuarto tema es algo más práctico que los anteriores. En primer lugar se presenta el formato de las instrucciones del lenguaje máquina y su funcionamiento, para después aplicarlo en la práctica con CODE-2. En la segunda parte del tema se profundiza en el funcionamiento teórico de CODE-2, que se complementa con distintos ejemplos prácticos. Finalmente, se explican teóricamente algunos de los lenguajes de alto nivel y el proceso de traducción y ejecución de los programas.

El quinto, y penúltimo tema, es una introducción a los fundamentos de un sistema operativo. El tema comienza con una definición de "sistema operativo" explicando su objetivos y funciones, además del contexto en el que es utilizado y el método de arranque de un ordenador con un sistema operativo ya instalado. Posteriormente se explica cómo el sistema operativo realiza la gestión de recursos. Primeramente se trata la gestión de recursos del procesador y cómo éste gestiona los procesos desde la monoprogramación hasta los sistemas concurrentes, así mismo se explican los puntos de decisión de la planificación y sus respectivos algoritmos. A continuación se estudia la gestión de memoria por un sistema operativo: la gestión de la memoria principal, incluyendo sus direcciones de memoria y los registros base y se explica en qué consisten las particiones dinámicas y distintos métodos de gestión de la memoria (paginación y segmentación). El tema continúa con el estudio de la gestión de las entradas y salidas, donde se introducen algunos conceptos como la atención de interrupciones y los problemas que se pueden causar si no hay tiempo para tratarlas. Por último, se explica en qué consiste un controlador DMA, así como su funcionamiento y modelos de E/S en sistemas operativos como UNIX y Windows NT. Finalmente, en cuanto a gestión de recursos se refiere, se nos presentó la gestión de archivos, directorios o carpetas, o sea, cómo se organizan y cómo se distribuyen en un sistema operativo. Finalmente se expusieron algunos de los sistemas operativos más relevantes, tales como Windows, UNIX, Linux, Mac OS… Y se exponen algunas de las amenazas a las que éstos están expuestos, además de una categorización de estas amenazas y algunas soluciones a ellas.

Para terminar, en el sexto tema se hace una introducción a las bases de datos explicando en primer lugar conceptos básicos sobre ellas, los tipos de arquitecturas de bases de datos y los modelos de datos (entidades, atributos, relaciones y restricciones) y de bases de datos (distribuidas y orientadas a objetos). Posteriormente se analiza más en profundidad las bases de datos relacionales: su modelo relacional y sus respectivas características. Por último, se presenta cómo es el diseño de una base de datos y sistemas gestores de bases de datos como SQL.

2.2 Prácticas y seminarios

Durante el curso se realizaron un total de cuatro prácticas y tres seminarios con objetivos muy diversos: desde elegir los componentes para montar un PC hasta manejar CODE-2 mediante instrucciones máquina.

La primera práctica titulada "Uso del sistema operativo" se trata de una introducción a Linux para que posteriormente lo instalásemos de manera nativa o con máquina virtual a nuestro ordenador. Una vez hecho esto, se pondrían en práctica una serie de órdenes de Linux en la línea de comando y se estudiarían la estructura de directorios de Linux. A continuación se analizaría en entorno gráfico de X-Windows y se pondrían en práctica diferentes comandos para la obtención de información del

sistema. En último lugar habría que realizar una serie de ejercicios, del que destaca principalmente, la instalación de una aplicación propia de Windows en Linux mediante la herramienta Wine.

En la segunda práctica ("Herramientas informáticas con aplicación en Ingeniería") se centra principalmente en el funcionamiento de MatLab y en la programación de funciones para el mismo MatLab. Una vez explicada la teoría de cómo funcionaba esta herramienta, se realizaron varios ejercicios entre los que hubo que programar algunas funciones, como por ejemplo, una función que calculase la resistencia equivalente a tres resistencias dadas en serie o en paralelo.

Posteriormente se realizó un seminario sobre CODE-2 a modo introductorio, explicando sus instrucciones, su funcionamiento y sus características, para posteriormente, enlazarlo con la práctica tres ("Funcionamiento a bajo nivel de un ordenador") que principalmente trataría sobre CODE-2. En esta práctica nos familiarizaríamos con el entorno de programación de CODE-2, con su repertorio de funciones y entendiendo la función de un programa ensamblador, el cual nos permitiría programar un programa para CODE-2 mediante texto en lugar de hacerlo en código hexadecimal. Por último habría que elegir uno de los ejercicios propuestos que consistían en programar un programa en CODE-2 para posteriormente presentárselo al profesor y corroborar que se ha entendido todo el proceso.

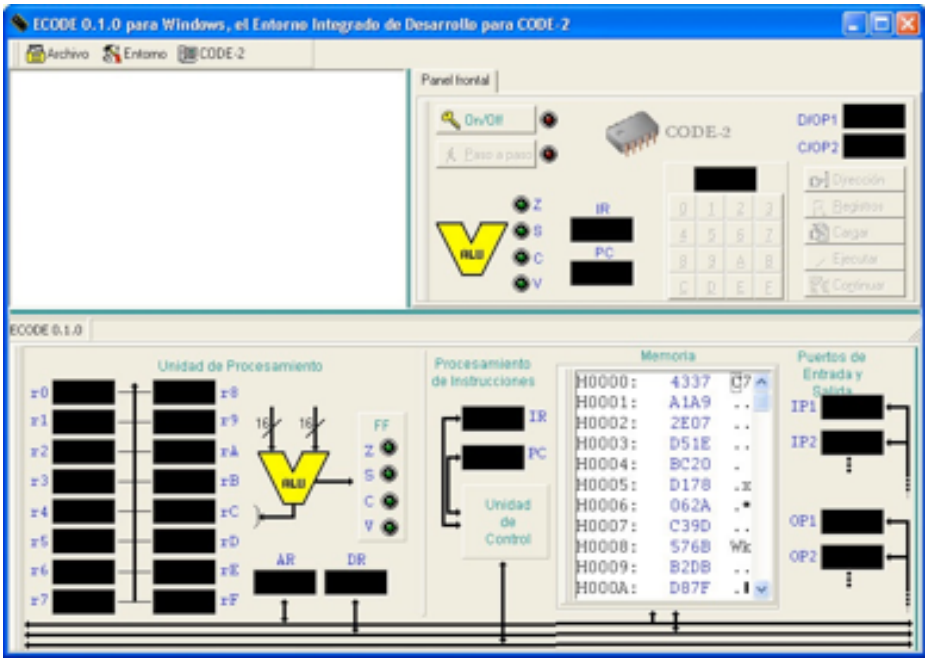

Figura 1. Entorno de ECODE, el simulador de CODE-2.

En la última práctica ("Uso básico de un Sistema Gestor de Bases de Datos") hubo que crear una base de datos mediante el programa Microsoft Access. Primero se comenzó con una introducción de cómo crear una base de datos con esta herramienta, cómo realizar consultas a la misma y cómo realizar informes de la base de datos. Una vez hecho esto, había que realizar una serie de seis ejercicios que consistían principalmente en crear una base de datos sobre libros con sus respectivos autores e ir repitiendo los pasos seguidos en la explicación de la práctica. Una vez hechos todos los ejercicios, se presentaría la base de datos al profesor para comprobar su correcto funcionamiento.

Por otro lado, se realizaron dos seminarios más. El primero, "Estructura y montaje de un PC" que consistía en el estudio de los componentes de un ordenador convencional y la interconexión entre sus componentes, para seguidamente realizar algunos ejercicios como: "comprar" nuestro propio PC eligiendo nosotros mismos sus componentes, uno a uno, o realizar un análisis de los componentes de nuestro ordenador.

El último seminario fue sobre software libre y los programas en código abierto. Durante el mencionado seminario se nos explicó los puntos a favor que tienen este tipo de proyectos, además de la historia del software libre y alguno de los proyectos más importantes por aquel momento.

2.3 Actividades en casa

Además, se realizaron algunas actividades complementarias a los temas que se estaban dando en la clase de teoría para completar la información proporcionada en el aula. Algunos de estos trabajos fueron: estudio del TOP 500, cómo se codifican algunos símbolos de ASCII, estudio de algunos procesadores INTEL, análisis de las diferencias entre sistemas analógicos y digitales, identificación y análisis de los componentes de nuestro propio ordenador, estudio sobre las amenazas de seguridad (software maligno)…

3 Perspectiva

Después de cuatro años estudiando el grado y habiendo cursado todas las asignaturas básicas, troncales y obligatorias de mi rama, he de decir que la asignatura está bastante bien planificada y organizada, facilitando la comprensión de los contenidos y actividades que se realizan en la asignatura.

Además, muchos de los contenidos que se impartieron en la asignatura han servido como base de conocimiento para otras asignaturas del grado, principalmente en asignaturas de electrónica como "Electrónica Digital" donde el uso del código binario es esencial y es impartido por primera vez en esta asignatura. También en la

asignatura troncal “Sistemas Electrónicos Digitales” los conocimientos impartidos en la asignatura han sido bastante útiles, principalmente para entender del funcionamiento en sí de los sistemas electrónicos y sus respectivas instrucciones, ya que el análisis del funcionamiento del reloj, además de otras características de los sistemas digitales y el estudio y uso de CODE-2 sirvieron como base para esta asignatura, de la misma forma, en la asignatura optativa “Implementación de Algoritmos en Hardware”, la introducción a estos sistemas, como CODE-2, ha sido muy útil como punto de partida para la comprensión de los contenidos de esta asignatura.

Por otro lado, la introducción a MatLab durante la práctica dos ha sido para mí esencial a lo largo de todo el grado, ya que en bastantes asignaturas he tenido que hacer uso de MatLab, siendo las nociones impartidas en esta asignatura de gran ayuda para programar o llevar a cabo funciones más complejas con esta herramienta. En la misma línea, en esta asignatura tuve mi primera toma de contacto con el sistema operativo Linux y su línea de comandos, usado en la mayoría de las asignaturas, especialmente las de la rama de telemática, haciendo que los conocimientos en Linux sean algo fundamental para el Grado en Ingeniería de Tecnologías de Telecomunicación y más para mi caso en concreto, ya que me encuentro estudiando la rama de Telemática del grado, donde el uso de Linux, o al menos el uso de las líneas de comandos de cualquier sistema operativo, es constante y obligatorio para la programación y configuración de los instrumentos del laboratorio de telemática: routers, switches, PABX, etc.

Por último, otros conocimientos teóricos, como tasas de transferencia, codificación de imágenes, textos, vídeos, sonidos…, incluso programación de bases de datos, han sido prácticos pues muchos de estos conceptos los he tenido que aplicar a alguna que otra asignatura.

4 Conclusiones

Fundamentos de Informática es una asignatura que sirve como toma de contacto para posteriores asignaturas del grado, por lo cual lo hace bastante útil a la hora de aprender conocimientos básicos y necesarios para la comprensión de otras asignaturas del grado, que no serán continuación de ésta, pero sí que en cierto modo toman como cimientos algunos conceptos aprendidos en esta asignatura. Además, recuerdo que la asignatura se me hizo amena, con unas prácticas bien planificadas (si era necesario, acompañadas de un seminario) y bien repartidas, lo que favorecía la comprensión de lo que se estaba haciendo y la finalización de las prácticas de una manera satisfactoria.

Quizás, como parte negativa de la asignatura, es que en algunos momentos se echaba en falta algo más temario de prácticas, ya que en alguna ocasión al

enfrentarnos a los problemas, al no haber incidido mucho en ese aspecto práctico, encontraba algunas dificultades. También, en el apartado de prácticas, en el seminario sobre el montaje de un PC eché en falta que se nos enseñase de manera práctica cómo montar un PC y el por qué de cada conexión y cómo funcionaban, ya que, aunque bien es cierto que lo vimos de manera teórica, algo más práctico lo habría hecho más tangible, por lo que probablemente podría mejorar nuestro aprendizaje.

No obstante, he de decir que es una asignatura a la que guardo buen recuerdo ya que el temario y las clases que se impartían me resultaban bastante interesantes, además de que todos los contenidos estaban bastante bien explicados y se nos aportaba suficiente material para poder entender correctamente la asignatura y lograr unos buenos objetivos finales.

Agradecimientos. Agradezco a Héctor Pomares Cintas su colaboración e implicación durante el proceso de creación de este artículo para que arribase a buen puerto.

Referencias

1. Prieto, A., Lloris, A. y Torres, J. C. Introducción a la informática. Ed. McGraw- Hill, 2006.
2. Universidad de Granada. Guía docente de la asignatura Fundamentos de Informática. http://grados.ugr.es/telecomunicacion/pages/infoacademica/guias_docentes/bsicas/fundamentos-tecnolgicos-y-empresariales/fundamentos-de-informtica/!

Instrucciones para Autores

Enseñanza y Aprendizaje de Ingeniería de Computadores (Teaching and Learning Computer Engineering) es una revista de Experiencias Docentes en Ingeniería de Computadores que edita el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, se publica anualmente, y se difunde tanto en papel como electrónicamente, a través del repositorio institucional de la Universidad de Granada (http://digibug.ugr.es/).

Los artículos remitidos para su evaluación pueden estar escritos en castellano o inglés, incluyendo un resumen y palabras clave en inglés en caso de que estén escritos en castellano, y deben seguir el formato descrito en la dirección web:

http://atc.ugr.es/pages/actividades_extension/

El correspondiente fichero .pdf debe enviarse a la dirección de correo electrónico jortega@ugr.es o mdamas@ugr.es

Los artículos deben abordar, tanto contenidos relacionados con la docencia universitaria en general, como con la docencia de asignaturas específicas impartidas por las áreas de conocimiento involucradas en estudios relacionados con la Ingeniería de Computadores, y también pueden aspectos relativos a las competencias profesionales y la incidencia de estos estudios en el tejido socio-económico de nuestro entorno.

En particular, se anima a antiguos alumnos de los estudios de Informática y a estudiantes de grado y posgrado a que envíen colaboraciones relacionadas con sus experiencias al cursar asignaturas relacionadas con la Ingeniería de Computadores, sugerencias, propuestas de mejora, etc.

Teaching and Learning Computer Engineering

Journal of Educational Experiences on Computer Engineering

May 2014, Number 4