RESEARCH ARTICLE

Adaptation of a Computer Programming Course to the ESHE Requirements: Evaluation Five Years Later

Ernest Valveny[a,b], Robert Benavente[a,b*], Àgata Lapedriza[c,b], Miquel Ferrer[b], Jaume Garcia-Barnés[b], and Gemma Sànchez[a,b]

[a]*Department of Computer Science, Universitat Autònoma de Barcelona, Bellaterra (Barcelona), Spain*; [b]*Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra (Barcelona), Spain*; [c]*Computer Science, Multimedia and Telecommunications Studies, Universitat Oberta de Catalunya, Barcelona, Spain*

In academic year 2010-2011 Spain has finished the process of introducing the regulatory changes derived from the Bologna Declaration and the new European Space for Higher Education (ESHE). These changes have implied the updating of university degrees' structure as well as the inclusion of the European Credit Transfer System (ECTS).

This paper describes the process of adaptation of two basic first-semester core subjects of computer engineering to one of the basic aspects of the ESHE, the adoption of the ECTS. The process described in the paper was developed in the framework of the pilot plan undertaken by the Universitat Autònoma de Barcelona between 2005 and 2010. The proposed course design implies a better coordination and integration of the contents of two different subjects that students follow simultaneously, and it is based on the combination of project-based learning and cooperative learning.

Once the experience has finished, an extended quantitative and qualitative analysis of the academic results over the five years period has shown an improvement in the students' learning outcomes.

Keywords: algorithms; computer engineering education; cooperative learning; programming; project-based learning.

1. Introduction

The European university system has experienced important changes since 1999, when the Bologna Declaration (Bologna 1999) established the basis of the new European Space for Higher Education (ESHE). With the development of a Europe of knowledge in the horizon, the main goal of the Bologna Declaration was the establishment of comparable degrees across Europe to encourage mobility and employability of European graduated students. One of the basis to achieve this goal has been the establishment of the European Credit Transfer System (ECTS)(ECTS 2005). The ECTS is based on the use of students' work as the unit to measure degrees' work load with 1 ECTS credit being between 25 and 30 hours of students' work. This new system is totally opposed to the former one which was simply based on the hours of lectures received by students.

Therefore, the adaptation to the ESHE implied many changes in higher education organization and methodology (Salas-Morera *et al.* 2009, Magdalena *et al.*

*Corresponding author. Email: robert@cvc.uab.cat

2008). First, education must be focussed on students and must take their learning process into account. Some proposals have been done in this direction such as the one described in (Tien 2000) where a completely individual-centered approach to engineering education is proposed. Moreover, achieving the goals of a good engineering education implies that courses can not be only based on the transmission of contents (Baum *et al.* 1994, Pister *et al.* 1995) and students must be provided with useful skills for their professional career.

The Spanish Education Ministry established the 2010–2011 academic year as the limit to adapt the existing degrees to the ESHE. To achieve this goal, a pilot plan to progressively adapt the previous degrees to the new framework was developed at the Universitat Autònoma de Barcelona from academic year 2005–2006. The pilot plan allowed the faculty to carry out innovative experiences by introducing active methodologies in their teaching and to perform a thorough testing of their academic results.

The experience presented in this paper involved two first-semester courses of the computer engineering degree that students follow in parallel. These subjects are Algorithms and Programming (hereafter AP) and Programming Languages (hereafter PL). They are core subjects in the curriculum and students enrolled in these courses were taking computer engineering degree as their major subject. The number of students varied each year between 150 and 300, with a mean number of students of 250 in AP and 230 in PL during the period of the experience. Both subjects belong to the same area of knowledge, that is, computer programming in imperative high-level languages (Wirth 1976). However, whereas AP has higher theoretical and abstraction components, PL is more practical and it is focussed on programming using a specific language (in this case, the C programming language (Kernighan and Ritchie 1978)).

In spite of their complementary nature, these courses had always been considered as independent subjects. As a consequence, it was difficult for students to understand the relationship between the two courses and to have a global view about the process of developing software applications.

In this paper, we present the adaptation process of AP and PL courses to one of the ESHE requirements, the ECTS as a standard measure of teaching activities. As ECTS moves the emphasis of teaching activities towards the effort of students, the proposed design is based on the combination of two active methodologies, namely Project-Based Learning (PBL) (Heitmann 1996, Kolmos 1996, Thomas 2000) and Cooperative Learning (Johnson and Johnson 1975). In addition, PBL and cooperative learning allowed us to propose an effective way of coordinating these two related courses to provide students with a global view of all programming stages. Students, in small groups, had to develop a project that was common for both courses. In this way, the contents of AP and PL became fully integrated providing a unified view of programming. Most of the lectures were related to the project through cooperative activities.

Once the pilot experience has finished, we present the analysis of the results obtained in this five-year experience to show the improvement of students' performance when these pedagogical approaches are applied. Our proposal has been the basis of a first-year module devoted to programming on the ESHE fully adapted degree that has started in academic year 2010-2011 at the Universitat Autònoma de Barcelona.

2. Course Design

In the design of the new organization of AP and PL courses, some constraints had to be taken into account due to the general organization of the degree. First, each subject had to maintain its independent status and, therefore, its own assessment system so that students could register to only one of them. Second, the total number of teaching hours of each subject could not vary. Following these constrains a new proposal for AP and PL courses was designed.

In the former system, AP was a subject of 6 credits (60 hours of teaching) distributed in 30 hours of theory, 15 of problems and 15 of practical works. PL had 4.5 credits distributed in 15 hours of theory and 30 hours of practical works. The duration of each course was 15 weeks. Teaching was done in a classical way, that is, theory and problems in teacher-led lessons, and practical works in the laboratory. Assessment was based on a weighted mean between the marks of a final practical work (30% of the mark in AP and 40% in PL) and a final exam (70% of the mark in AP and 60% in PL).

Before the pilot experience, the fact that AP and PL had always worked independently caused inefficient situations such as topics that were explained in both subjects, activities that were very similar (e.g. the practical works of both subjects were almost the same), and lack of synchronization between the theory about the design of an algorithm (in AP) and its implementation in C (in PL).

Due to these problems, most students perceived AP and PL as unrelated subjects and did not have a global view about their contents. Consequently, many students did not feel motivated for these subjects, as the number of drops during the year showed. Therefore, apart from adapting the courses to the ESHE, the goals of the experience were:

- To provide students with a global view of programming, from the beginning (analysis of the problem and design of an algorithm) to the end (implementation with a programming language)
- To coordinate the planning of the two subjects avoiding repetitions of topics or activities
- To increase the involvement of students in their learning
- To provide students with useful skills for their future employment as engineers, namely real-problems solving, analysis and synthesis abilities, initiative and self-organization, communication abilities, and teamwork.

The central point of the approach to achieve all these goals was the adoption of a PBL methodology. During the semester, students worked in small groups to develop a joint project which covered all the topics of the syllabus of both subjects. Thus, although both subjects remained as independent courses, the project acted as the thread that related all the activities done in both courses. In addition, it allowed students to have a global view of programming and to face a real programming project.

At the beginning of the experience, the adaptation process began with an analysis of the contents and requirements of each subject to define a shared syllabus including all the stages of the classical programming cycle: analysis, design, coding and testing (See Fig. 1). At this point, the topics that were included in this shared syllabus are: basic programming concepts, control structures, simple and complex data types, functions, dynamic memory, and files.

The contents of each topic in the syllabus were arranged in four different modules and for each of them, different teaching strategies were adopted to enhance the students' learning process. These four modules and their respective teaching

strategies are the following:

(1) *Basic algorithmics theory* (in teacher-led lessons). The theoretical contents of the subject were presented in a general and abstract way by using pseudocode.
(2) *Analysis and synthesis of algorithms* (in problem-solving seminars). Some problems were presented to the students and they proposed an algorithm (in pseudocode) suitable for its solution.
(3) *Basic knowledge about C programming language* (in programming seminars). The syntax of the C programming language was explained showing its correspondence to the generic structures introduced in the theoretical lessons. Exercises consisting in the translation of algorithms from pseudocode to a C program were proposed.
(4) *Coding algorithms in C* (in the laboratory sessions). A series of programming exercises were proposed to be implemented in the computer.

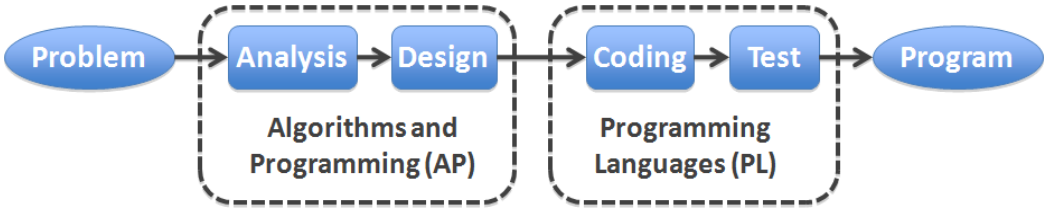

Figure 1. The first stages of the programming cycle (analysis and design) are taught on AP whereas the stages of coding and testing correspond to the areas covered by PL.

Taking the nature of each subject into account, the two first modules were included in AP whereas the other two were included in PL.

One of the key issues of the experience was the precise coordination of the calendars of both courses. Hence, for each unit on the syllabus, the sequence of a teacher-led lesson, a problem-solving seminar, a programming seminar, and a laboratory session was always assured. With this planning students followed a coherent learning sequence, from the abstract level in the theory lessons to the practical level in the laboratory sessions. Fig. 2 shows a scheme of the course timing.

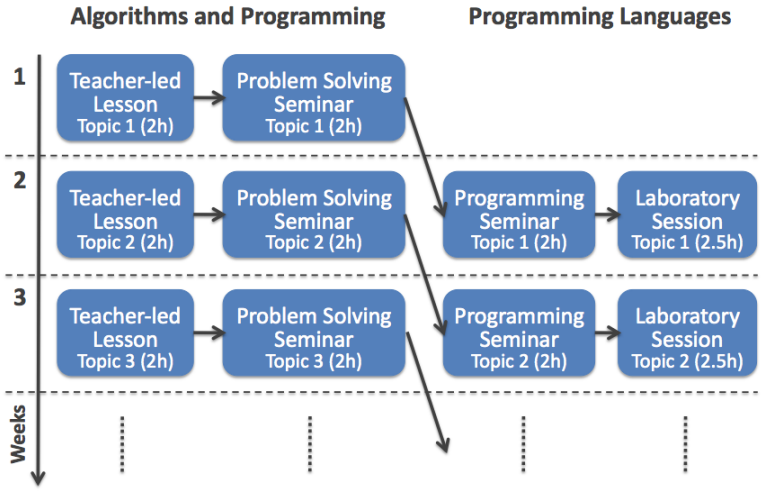

Figure 2. Timing of the courses. The teaching sequence guarantees a coherent learning sequence to students. PL course starts one week later than AP.

This scheme was followed during the first eight weeks of the course when the

foundations of programming were introduced. Given that the projects developed by students integrated all the contents of both courses, most of the problems proposed in this part of the course corresponded to small parts of the final project. This fact optimized the available time and also contributed to keep the involvement of students in the project.

In the last four weeks of the course, students were mainly focused on the final development of the project by integrating the parts carried out during the previous sessions and implementing the remaining and most complex parts.

Another key point of the course design was the change to a continuous assessment system to monitor the learning process of students, detect deficiencies to be solved during the course, and advise the students during the development of the project. Thus, in addition to the final exam, all the work done by students was taken into account to obtain the overall mark. At the end of each seminar and laboratory session there was an assessment activity such as a problem, a test, or a C program. There were also some milestones with deliverables of the project and two partial exams during the course.

All these changes were carefully designed taking into account that the total time devoted by teachers to activities with students should not be higher than in the original design. However, it has to be remarked that the total time spent by teachers increased due to the continuous assessment system, although not in a very significant amount. In the following sections, the learning methodologies and the continuous assessment system followed in the experience are detailed.

3. Methodology

PBL and cooperative learning were the basic learning strategies used to implement the course design. Both strategies have been suggested as effective strategies for achieving engineering expertise (Smith 1988). Fig. 3 shows a scheme on how these methodologies were integrated. The joint project was developed under the PBL methodology and some of its parts were developed in the seminars, where students worked in small groups under the cooperative learning methodology. In the following sections details on how these methodologies were applied in the experience are given.

3.1 *Project-Based Learning*

Project-based learning (Heitmann 1996, Kolmos 1996, Thomas 2000) is a teaching methodology related to the more general problem-based learning (de Graaf 1993, Hmelo-Silver 2004, Dolmans *et al.* 2005), which has been successfully applied in engineering education (Turner and Zachary 1999, Rebelsky and Flynt 2000, Blake 2005, Magdalena *et al.* 2008, Chang *et al.* 2008, Mantri *et al.* 2008). It is a dynamic approach to teaching in which students solve real-world problems working in small groups and acquiring knowledge as they need it to develop the different parts of the project. Furthermore, PBL allows developing skills such as teamwork, analysis and synthesis abilities, and project management.

On our courses, students worked in groups of two or three members to develop a project common to both subjects. Through the project, students applied all the contents of the courses and went through all the stages of the classical programming cycle. However, since these were first-semester courses, the process was highly guided by the teachers and each group had an advisor to guide them in the process

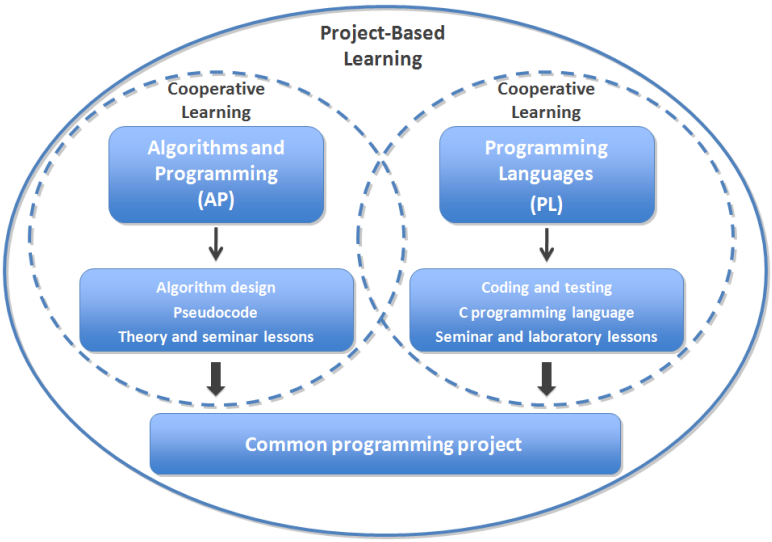

Figure 3. Methodologies involved in the courses of Algorithms and Programs (AP) and Programming Languages (PL).

and to give them feedback as they developed the project.

One important issue in PBL is the selection of a good project topic. To motivate students, projects should be as real and challenging as possible. Game programming has these and other interesting characteristics (Chen and Cheng 2007) and, for this reason, the proposed projects were simplified versions of well-known computer games (e.g. Tetris or PacMan).

At the beginning of the course, students received a generic statement with the basic specifications of the project. The statement was open enough to allow students to take their own design decisions, but always under the supervision of their advisor, which is a key issue to guarantee a correct student-centered learning process (Hmelo-Silver and Barrows 2006). The project was highly modular and most of the exercises proposed in the seminars and the laboratory sessions were preparatory exercises for the final project or small parts of it. The work done at these sessions should be the basis for the students to complete the project by themselves in a semi-supervised way. They should take their own responsibility to find the time and space to discuss about the project and integrate all the small modules into the final result. In any case, they could rely on the guidance of their advisor for any problem encountered during the implementation of the project. For a good follow-up of their progress, there were two milestone deliverables at the fourth and eighth weeks of the course. For these deliverables, students received a more specific statement with the requirements of the module to deliver. In AP they presented the design of the requested module and in PL its corresponding implementation. At the end of the semester, students had developed the whole project, from the basic design of the data structures and modular decomposition to the final implementation in C language.

3.2 *Cooperative Learning*

Cooperative learning (Johnson and Johnson 1975) is a learning strategy in which students work in small groups to build knowledge. Each student contributes to the proposed task and their interaction results in more benefits to each member than if they worked alone. This methodology also develops some useful skills such as critical analysis and communicative abilities.

In our approach, cooperative learning was applied in the seminars of both sub-

Table 1. Example of a cooperative activity performed in a PL seminar

Subject: Functions in C Language.
Task: Implementation of a calculator for complex numbers.
Background: You need to review the theory of control structures, functions, the syntaxis of function in C and the parameter passing methods.
Instructions: (1) Form an even number of groups of 2 or 3 people. (2) Perform one of the following activities: *PART A: Code a function to add two complex numbers, and another one to multiply two complex numbers.* *PART B: Code a function to subtract complex numbers, and another one to compute the module of a complex number.* (3) Form new groups joining each *PART A* group with a *PART B* group. (4) Implement the entire program corresponding to a calculator of complex numbers. Code the *main* function that shows a menu, scans the complex numbers and allows choosing the operation to be performed. Use the previously coded functions and take the parameters passing into account.
Evaluation: The final mark of the activity (all marks are computed in a range of values from 0 to 10) will be the mean between the marks of the first part (A or B) and the mark of the final program.

jects. Cooperative activities were carried out by small groups from two to four students, which were randomly created for each session. Therefore, students from different project groups could work together in the cooperative activities at the seminars. In this way, as the tasks proposed in the seminars are preparatory exercises or small parts of the final project, we were promoting that each project group could compare and discuss several approaches to each small part of the final project. Before each session, a brief script was prepared as a route-map for the students. It included:

- A summary of the previous knowledge needed for the session.
- A description of the proposed activity.
- The assessment criteria.

After a brief theoretical introduction, students must read the session script and work on the proposed activities, which could be solved in different ways. At the end of each seminar session, students must deliver the solution of the proposed activity which was evaluated taking both the individual and the collective contribution to the task into account. Table 1 shows one of this activities as example.

Table 2. Summary of overall results in Algorithms and Programming. Marks are in the range 0-10.

	2003–04	**2004–05**	**2005–06**	**2006–07**	**2007–08**	**2008–09**	**2009–10**
% Drops	21.78	31.35	21.71	19.19	10.98	23.65	18.75
% Fails	37.54	26.02	34.16	37.37	39.31	29.05	31.82
% Passes	40.69	42.63	44.13	43.43	49.71	47.30	49.43
Mean Exam	4.31	5.08	5.23	4.62	5.46	5.72	5.49
StDev Exam	2.02	1.40	1.56	2.28	2.20	2.21	2.02
Mean Passes	6.37	5.29	6.54	5.65	6.98	6.75	6.50
StDev Passes	0.91	1.46	0.98	0.64	0.92	1.17	0.89

4. Assessment

In the courses' design a continuous assessment system was implemented and, hence, the whole learning process of students was taken into account to obtain the overall mark of these courses. The scoring rule was the same for both subjects. The overall grade was a weighted mean of three marks: exams (40%), project (30%), and activities from seminars and laboratories (30%). Eq. 1 summarizes the rule:

$$Mark = 0.4 \cdot Ex + 0.3 \cdot Pr + 0.3 \cdot Ac \tag{1}$$

where Ex, Pr, and Ac are the marks of the exams, the project and the activities performed in the seminars and/or laboratories, respectively.

Although a final exam was still done at the end of the semester, two partial exams were done after the fourth and the eighth weeks of the course. Thus, the exams' mark (Ex in Eq. 1) was in turn a weighted mean of the partial exams (25% each) and the final exam (50%). Notice that with this criteria, the final exam mark is only a 20% of the final mark instead of the previous 70% in AP or 60% in PL in the initial system.

The mark of the project (Pr in Eq. 1) was also a weighted mean of the marks obtained in the two deliverable milestones (20% each) and in the final project evaluation (60%).

The third partial mark in the scoring rule (Ac in Eq. 1) was obtained as the weighted mean of the activities carried out in the seminars and the laboratory sessions.

It is important to remark that although the individual written exams still had an important weight (40% of the overall mark), the work carried out along the course, which was mainly related to the project, supposed the remaining 60% of the final mark. This fact encouraged students to have an active participation in the courses.

5. Results and Discussion

After the five years of the experience, a quantitative and qualitative analysis of the results has been done.

The results from academic year 2003-2004 (two years before beginning the experience) to year 2009-2010 have been analyzed in terms of the drops, passes and fails percentages, mean mark on the final exam, and mean overall mark. Tables 2 and 3, and Figs. 4 and 5 summarize the academic results of these years.

Table 4 presents the values of all the indicators calculated for the periods 2003-2005 (before the beginning of the experience) and 2005-2010 (after the courses' adaptation).

Table 3. Summary of overall results in Programming Languages. Marks are in the range 0-10.

	2003–04	**2004–05**	**2005–06**	**2006–07**	**2007–08**	**2008–09**	**2009–10**
% Drops	15.02	32.53	24.29	22.66	21.97	14.39	5.92
% Fails	38.98	37.67	25.36	31.53	33.53	33.81	34.91
% Passes	46.01	29.79	50.36	45.81	44.51	51.80	59.17
Mean Exam	4.83	3.80	5.36	4.98	5.29	6.62	6.25
StDev Exam	2.72	2.56	2.39	2.74	2.69	2.43	2.49
Mean Passes	6.68	7.38	7.46	7.35	7.82	7.86	7.63
StDev Passes	1.10	1.33	0.90	1.21	0.94	1.04	1.02

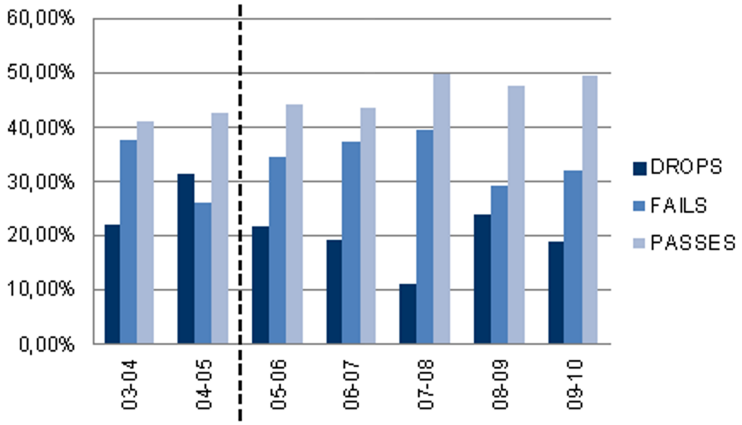

Figure 4. Results for 'Algorithms and Programming' (AP). The dashed line indicates the beginning of the new methodology.

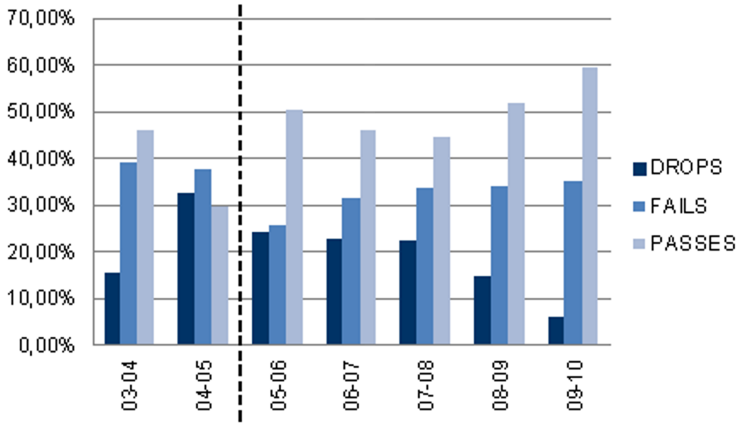

Figure 5. Results for 'Programming Languages' (PL). The dashed line indicates the beginning of the new methodology.

Before starting the experience, both courses had showed a progressive decrease in the percentage of passes and an increase in the percentage of drops and fails. This fact was critical in the case of PL with less than 30% of passes in year 2004-2005. However, after the beginning of the experience results tended to improve progressively, as we detail in the following paragraphs.

In AP (see Table 2 and Fig. 4), passes increased from 42.63% in 2004-05 to percentages near 50%, whereas drops decreased from 31.35% in 2004-05 to values around 20% (with a minimum of 10.98% in 2007-08). The percentage of fails has experienced an increase in the years after the beginning of the experience (2.49% in the means of the periods as shown in table 4). We think that this fact is related

Table 4. Global results on the two subjects for periods 2003–2005 (before the experience) and 2005–2010. Marks are in the range 0-10.

	Algorithms and Programming		**Programming Languages**	
	2003-2005	**2005-2010**	**2003-2005**	**2005-2010**
% Drops	26.35	19.06	23.47	18.88
% Fails	32.04	34.53	38.35	31.02
% Passes	41.62	46.41	38.18	50.10
Mean Exam	4.57	5.25	4.36	5.58
StDev Exam	1.87	2.07	2.69	2.60
Mean Passes	5.84	6.61	6.93	7.59
StDev Passes	1.32	1.01	1.23	1.03

to the decrease in the number of drops (7.29% in the means of the periods) because part of the students that did not drop the course passed, but the rest increased the percentage of fails. However, we must point out that the increase on the percentage of passes between periods (4.79%) is almost twice the increase on the percentage of fails which is 2.49%.

In the case of PL (see Table 3 and Fig. 5), results also show an improvement on students' performance. A similar pattern can be observed: drops were reduced from 32.53% in 2004-2005 to only 5.92% in 2009-2010, and passes increased from 29.79% in 2004-2005 to 59.17% in 2009-2010. Unlike AP, in PL the number of fails was also reduced with respect to the years before the experience. However, after a sharp decrease of 12.31% in course 2005-2006, the percentage of fails progressively increased in the subsequent years. Again the explanation to this fact can be found in the decrease on the number of drops that makes increase the number of passes, but also the number of fails. Nevertheless, the highest percentage of fails in the period after the beginning of the experience (34.91% in year 2009-2010) is still lower than the percentages on the period 2003-2005 and, as can be seen in Table 4, the global results for both periods show a reduction of 7.33% on the number of fails.

Moreover, the mean mark on the exams and the overall marks (computed according to Eq. 1) improved since the beginning of the experience (see Tables 2 and 3). We want to point out that the change to a more practical methodology did not mean a decrease in the course contents. Indeed, the difficulty level of the exams was increased each year.

To consider the opinion of the students about the methodology employed and its results, a survey based on the SEEQ (Students' Evaluation of Education Quality) survey (Marsh 1982) was handed out at the end of the last year of the experience. For each of the statements in the survey, students had to score between 1 (Very poor) to 5 (Very good). Table 5 summarizes the results on the questions that are more related to the goals of the experience.

Questions about their own learning process (Q2, Q3 and Q4) show that students had the perception of learning useful things in these courses and their interest in the subject increased after taking them.

Regarding the assessment strategy (questions Q10 and Q11), students thought that the assessment strategy was fair and appropriate mean score of 3.17 with 44.77% of the students scoring their agreement to the statement as good or very good). Score increases about Q11 ("The contents of the exams and tasks of the course agree with the contents of the course"), with a mean score of 3.58.

A set of specific questions (Q20 to Q23) about the PBL methodology employed in the experience were also proposed to the students. In general, they were very

Table 5. Survey results: Questions about the courses. (Scores: 1=Very Poor, 2=Poor, 3=Moderate, 4=Good, 5=Very Good)

Question	Mean	STDev
Q02: I have learnt topics that I consider as valuable	4.11	0.79
Q03: My interest in the subject has increased as a result of this course	3.68	0.98
Q04: I have learnt and understood the contents of this course	3.89	0.85
Q10: The assessment method of these subjects is fair and adequate	3.17	1.15
Q11: The contents of the exams and tasks of the course agree with the contents of the course	3.58	0.91
Q20: The fact that most of the activities of the course are related to the project has helped me to have a global view of the subject	3.63	0.94
Q21: The topic of the project is interesting	3.84	0.94
Q22: The knowledge acquired developing the project helps understanding theoretical contents	4.11	0.85
Q23: Doing the design of the project algorithm previously to its implementation in C helped me to obtain a better code	2.74	1.19

positive about the methodology used (Q20), the topic proposed for the project (Q21) and the usefulness of the different activities in their learning process (Q22). Surprisingly, question Q23 only obtained a mean score of 2.74 and a significant part of the students (about 44%) disagreed about the statement "Doing the design of the project algorithm previously to its implementation in C, helped me to obtain a better code". This might be because students were too novice to see the value of a good design. However, understanding the need of designing an algorithm before implementing it is one of the main goals of these subjects, and hence, this point should be improved in the new degree.

Finally, from a qualitative point of view, we consider that the problems detected before starting the experience (i.e. contents overlaps, lack of a joint view, unmotivated students) were mostly solved. The coordination of the contents and the sequence of lectures (theory-seminars-labs) optimized the existing resources (time and faculty) and allowed the teachers to follow the students' progress better. This fact contributed to increase the students' interest for the subjects, which could be observed in a higher attendance to lectures and also a more active participation of students in lectures and seminars.

6. Conclusions

In this paper we have presented the process to adapt two first-year subjects on computer engineering to the ESHE. The adaptation was performed during a five-year experience and this fact has given the opportunity to analyze the results in a longer than usual period. The data collected during these five years allows having a wider perspective on the benefits of the new design based on active methodologies.

The goals proposed at the beginning of the experience were achieved through the coordination of the two courses and the application of PBL and cooperative learning. The two courses involved in the experience have been successfully adapted to the ESHE requirements and the approach presented in this paper is the basis of a module devoted to the basics of programming in the new degree adapted to the ESHE that has started in academic year 2010-2011.

The analysis of the results demonstrates an improvement of students' performance shown by an increase on the percentage of passes and on the overall marks, and by a decrease on the percentage of drops.

The new design of the courses, based on PBL, also allowed the students to work on a real programming project where they went through all the stages of programming and acquired some useful skills for engineering such as teamwork, critical analysis, and communicative abilities. Finally, the survey handed out to students shows their satisfaction with the methodology employed in these courses.

References

The Bologna Declaration of 19 June 1999. Joint declaration of the European Ministers of Education. Bologna (Italy).

ECTS Users' Guide, European Credit Transfer and Accumulation System and the Diploma Supplement, 2005. Directorate-General for Education and Culture. Brussels (Belgium).

Baum, E., *et al.*, Engineering Education for a Changing World, 1994. Technical report, Amer. Soc. Eng. Education, Washington, DC.

Blake, M., 2005. Integrating Large-Scale Group Projects and Software Engineering Approaches into Early Computer Science Courses. *IEEE Trans. Educ.*, 48 (1), 63–72.

Chang, G., *et al.*, 2008. A Progressive Design Approach to Enhance Project-Based Learning in Applied Electronics Through an Optoelectronic Sensing Project. *IEEE Trans. Educ.*, 51 (2), 220–233.

Chen, W. and Cheng, Y., 2007. Teaching Object-Oriented Programming Laboratory With Computer Game Programming. *IEEE Trans. Educ.*, 50 (3), 197–203.

de Graaf, E., 1993. Problem-based Learning in Engineering Education. SEFI Cahier No.4: Project-organized Curricula in Engineering Education.

Dolmans, D., De Grave, W., Wolfhagen, I., and van der Vleuten, C., 2005. Problem-based learning: future challenges for educational practice and research. *Medical Education*, 39 (7), 732–741.

Heitmann, G., 1996. Project-oriented study and project-organized curricula: A brief review of intentions and solutions. *European Journal of Engineering Education*, 21 (2), 121–131.

Hmelo-Silver, C., 2004. Problem-based learning: What and How Do Students Learn?. *Educational Psychology Review*, 16 (3), 235–266.

Hmelo-Silver, C. and Barrows, H., 2006. Goals and strategies of a problem-based learning facilitator. *Interdisciplinary Journal of Problem-based Learning*, 1 (1), 21–39.

Johnson, D. and Johnson, R., 1975. *Learning Together and Alone: Cooperation, Competition and Individualization*. Englewood Cliffs, NJ: Prentice-Hall.

Kernighan, B. and Ritchie, D., 1978. *The C Programming Language*. Englewood Cliffs, NJ: Prentice-Hall.

Kolmos, A., 1996. Reflections of Project Work and Problem-Based Learning. *European J. Eng. Educ.*, 21 (2), 141–148.

Magdalena, R., *et al.*, 2008. A Teaching laboratory in Analog Electronics: Changes to Address the Bologna Requirements. *IEEE Trans. Educ.*, 51 (4), 456–460.

Mantri, A., Dutt, S., Gupta, J., and Chitkara, M., 2008. Design and Evalutation of a PBL-Based Course in Analog Electronics. *IEEE Trans. Educ.*, 51 (4), 432–438.

Marsh, H., 1982. SEEQ: A Reliable, Valid, and Useful Instrument for Collecting Students' Evaluations of University Teaching. *Brit. J. Educ. Psy.*, 52 (1), 77–95.

Pister, K.S., *et al.*, 1995. *Engineering Education: Designing an Adaptive System*. Washington, DC: National Academy Press.

Rebelsky, S. and Flynt, C., 2000. Real-world program design in CS2: The roles of a large-scale, multi-group class project. 192–196.

Salas-Morera, L., Berral-Yerón, J., Serrano-Gómez, I., and Martínez-Jiménez, P., 2009. An Assessment of the ECTS in Software Engineering: A Teaching Experience. *IEEE Trans. Educ.*, 52 (1), 177–184.

Smith, K., 1988. The nature and development of engineering expertise. *European J. Eng. Educ.*, 13 (3), 317–330.

Thomas, J., 2000. A Review of Research on Project-Based Learning [online]. Available from: http://bobpearlman.org/BestPractices/PBL_Research.pdf [Accessed 18 July 2011].

Tien, J., 2000. Individual-Centered Education: An Any One, Any Time, Any Where Approach to Engineering Education. *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, 30 (2), 213–218.

Turner, J. and Zachary, J., 1999. Using course-long programming projects in CS2. 43–47.

Wirth, N., 1976. *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ: Prentice Hall.