Comprehensive simulation of a Software Project throughout several subjects

Maria Ferré, Carlos García-Barroso, Montse García-Famoso, David Sánchez, Aida Valls

***Abstract*— Software projects are amongst the most common professional activities of computer engineers. Gaining competences on the design and development of software projects is, however, a complex issue that cannot be tackled within a single subject. In this paper, we detail the design and implementation of a teaching methodology that aims at providing a comprehensive simulation of the whole life cycle of software projects throughout the coordination of the practical exercises of subjects of different courses. The proposal consists in developing the same software project in consecutive courses under the perspective of the three main roles involved in the project development: designer, developer and director. In the second course, the students exercise the *developer role* in teams of 4 people under the supervision of a student of the fourth course, who acts as the *director*. The director has previously proposed a formal design of the project in the third course (*designer role*). This methodology has been successfully deployed for 4 years in the Computer Engineering Degree at the Universitat Rovira i Virgili. The results show an improvement of the skills and competences related to the three involved subjects, which include better programming quality, better team coordination and fulfilment of deadlines, as well as a much practical view of the director's role.**

***Index Terms*— Integrated project, project management, software design and development.**

I. Introduction

SOFTWARE architect and developer is amongst the most demanded profiles for computer engineers [1]. Software projects are, in fact, the natural professional activity of computer engineers because it requires of competences that are exclusive to their profile [2]. Therefore, it is of great importance that computer engineering degrees prepare future engineers in the competences related to their professional jobs.

A relevant feature of software projects is their complexity [3]. Software, as an industrial product, is singular (that implies slow production) and complex (which makes it difficult to test it under all possible conditions). Productivity tends to be low and careful planning, cost estimation and development management are required.

Training on software design and development is, consequently, also complex. The multiple perspectives that should be considered (problem analysis, design, management, programming and test, see Section 3) and the competences that should be covered (detailed in Section 4) cannot be encompassed in just one subject. Also, the dependencies among those competences and the different degrees of maturity that are needed to practice them all, make it difficult cover all of them through several subjects of just one academic course.

If we implement training in a traditional way, that is, by means of independent practical exercises among subjects spread across the degree, we risk offering a partial and simplistic view of software projects. To tackle this problem, within the Computer Engineering Degree at the Universitat Rovira i Virgili, we have designed and implemented an integrated training methodology that is based on coordinating and giving continuity to the practical exercises of three compulsory subjects of different courses that have contents nuclear to software development: Programming (2nd course), Analysis and Design of Applications (3rd course) and Projects and Computer Systems (4th course). In a nutshell, our method consists of developing the same software project in a coordinated way through the degree under the perspective of the three main roles involved in a software project: director, software architect and programmer. This results in a comprehensive simulation of a software project in which each student exercises all roles (throughout several courses) and interacts with students with the same or different role (programmer – programmer and director – programmer). This gives students a practical and complete view of software projects and their interactions, and enables simulating the scenarios and difficulties of the profession.

This proposal has been presented in JENUI 2019 [4], where it won the SISTEDES 2019 award given by the *Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software*. In the current paper, we extend the work presented in JENUI by providing additional details on how the project has been put into practice across the different subjects and providing a more comprehensive analysis of the results obtained in several courses.

The remainder of the paper is organized as follows. Section 2 describes and discusses related works. Section 3 details the project life cycle that we apply in the degree and describes the involved roles. Section 4 explains the subjects that participate in this experience and depicts their contents, competences and evaluation. Section 5 describes the implementation of the

Maria Ferré, Carlos García-Barroso, Montse García-Famous, David Sánchez and Aida Valls are with the Department of Computer Engineering and Mathematics of the Universitat Rovira i Virgili, Tarragona, Spain (email maria.ferre@urv.cat, carlos.garciabarroso@urv.cat, montse.garcia@urv.cat, david.sanchez@urv.cat, aida.valls@urv.cat).

DOI (Digital Object Identifier)

integrated project across the involved subjects, details the contents of the practical exercises and discusses the tasks performed by the students under different roles. Section 6 details the evaluation criteria and the tools employed to quantify the obtained results, both quantitatively and qualitatively. Section 7 discusses the results and impact, comparing them with a more traditional methodology based on independent practical exercises. Finally, Section 8 presents the conclusions of the work.

II. Related Work

Several examples of similar approaches can be found in recent literature, mainly, within the context of *Project-based learning* [5,6]. In [7] the authors propose a 15-week project in which the students, organized as teams, distribute the roles involved in the design and implementation of a distributed application with a database. In our case, the distribution of roles is done throughout several courses according to the degree of maturity of the student. In this way, each student ends up exerting all the roles involved in a software project.

Other works have also proposed projects that encompass several subjects. In [8] the authors propose coordinating three concurrent subjects belonging to the software engineering specialization in order to carry out a common project. A similar experience is presented in [9], but covering concurrent subjects of different specializations. Finally, in [10] the authors integrated the entire contents and practical works of two subjects during a semester in order to carry out a common project. Our proposal also implies the coordination of several subjects, but these take place in different courses. In this way, it is possible to practice roles that require different degrees of maturity in the most suitable moment, while also considering the dependencies among the competences that are needed to exert them. Working in the same project throughout several courses also makes it possible to increase the complexity of the project and leave a greater imprint in the student.

Finally, unlike other integrated projects developed throughout the degree but executed independently for each subject [11], our approach is based on teamwork among students of the same and distinct subjects. In this way, we can simulate the interactions between roles that may occur in real-word projects: director-programmer and programmer-programmer.

III. Software Life Cycle

The methodology we propose is based on the traditional phases of the cascade life cycle, which are depicted in Fig. 1.

The project begins by collecting the requirements. The student should compile and write a document with the functional and non-functional requirements of the application from the scripts provided by the teacher. Functional requirements will be analyzed and transformed into a formal specification. This specification, together with the non-functional requirements, will be employed to design the application with a concrete technology.

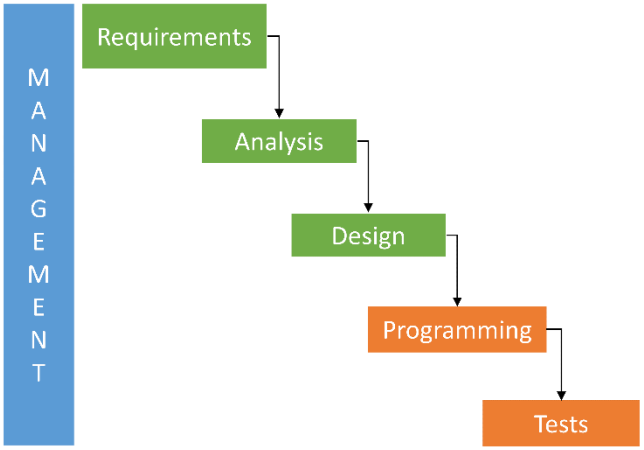

Fig. 1. Main phases of the software life cycle. In blue: phases of the director; in green: phases of the architect; in orange: phases of the programmer.

The outcome of these steps is a detailed UML specification depicting the classes and modules of the application, as well as the design of the graphical interface and of the data persistence. This will constitute the input of the programming phase, which will implement the proposed design. Finally, the software obtained from the programming phase will be tested (for correctness) and validated (for compliance with the requirements). During the requirements, analysis and design phases we follow the indications of the Unified Process [12]; in particular, these phases are executed in an iterative and incremental way.

Regarding the roles involved, as shown in Fig. 1, the three first phases encompassing the design of the application are executed by the *software architect*. In these phases, the students employ software engineering techniques and tools (UML), but they do not produce any code. The last two phases refer to the programming and test of the application, and they are executed by a team of *programmers* from the specification obtained in the design phase. These phases just require programming skills. In parallel to all the phases of the project, the *director* manages the process. In the first stage, management consists of identifying and assigning tasks and performing a temporal planning. Afterwards, he should supervise and monitor the whole project. This requires planning and team management skills.

IV. Subjects Involved in the Integrated Project

Our project integrates the practical exercises of three different subjects: PR (Programming), ADA (Analysis and Design of Applications) and PSI (Projects and Computer Systems). All are compulsory subjects of the Degree of Computer Engineering at the Universitat Rovira i Virgili, under the specialization of software development. The three subjects have 6 ECTS credits and are organized as shown in Table 1.

Specifically, in Table 1 we depict the main information on these subjects: academic course, semester, number of lecturers involved, average number of students per year and weight in the evaluation of the practical exercises involved in the project. We can see that the project involves more than 180 students and a

team of 4 lecturers.

Although the subjects are studied in the order shown in Table 1, the project begins in ADA. In this subject, the student analyses and designs the problem that constitutes the project. This work is performed in the second semester of the third course. In the next semester (1st of 4th course), the same students (within the PSI subject) use the design they made in the former course to act as directors of the project by supervising a team of students of the 2nd course (within the PR subject) that will take care of programming the application. These latter students will start a new project in the 3rd course when they enrol into a new course of ADA, becoming new architects and directors.

The deployment of the project along three courses allows us to work on different competences at different moments and at various levels. In Tables 2, 3, and 4 we show the competences involved in the three subjects. Some of these competences would be hardly treatable with individual and independent practical tasks. For example, the competency CM3 is practiced when the team of students from PR works under the supervision of the director from PSI. Negotiation, communication and work habits are important in PR and ADA, but leadership is put into practice in PSI. On the contrary, the competences A2, CM1 and CM2 are treated in ADA at a theoretical level and in PSI at a practical level. We also consider that the competences B1 and CT5 (from PSI) are better acquired through the proposed integrated project.

V. The Project in Detail

Due to the complexity of software, its development is usually carried out by teams. Moreover, as we have introduced in Section 3, any project with a minimum level of complexity will go through several phases in which different professionals exert different roles and interact and coordinate between them. To make the student simulate an experience similar to what she/he may encounter in a professional environment, and to acquire the skills and competences associated to all the involved roles in an integrated way, we have distributed the tasks and roles in the following way (see also Fig. 2):

- In ADA, each student exerts the *role of architect* and analyses and designs the application proposed within the project.
- In PSI, the same students exert the *role of director* and use the design they made in ADA to supervise the programming done by teams of students from PR.
- In PR, teams of students exert the *role of programmers* and testers under the supervision of students from PSI.

Table I
Information of the subjects involved in the project

Subject	Course Semester	#lecturers	#students	Evaluation weight
PR	2nd, C1	3	100	20%
ADA	3rd, C2	1	55	50%
PSI	4th, C1	1	30	15%

Table II
PR Competences

Code	Competence
CM3	Be able to understand the importance of negotiation, effective work habits, leadership and communication skills in all aspects of software development.
B8	Be able to work in groups and in a multilingual and multidisciplinary environment.
CT5	Communicate information clearly and precisely to a variety of audiences.

Table III
ADA Competences

Code	Competence
A2	Have knowledge of taking measurements, calculations, evaluations, valuations, surveys, studies, reports, work plans and other similar studies in IT.
CM1	Be able to design, develop, select and evaluate IT applications and systems, ensuring their reliability, security and quality, in accordance with ethical principles and the current legislation and regulations.
CM2	Be able to plan, conceive, implement and manage IT projects, services and systems in all areas, leading their start-up and ongoing improvement, and evaluating their economic and social impact.

Table IV
PSI Competences

Code	Competence
A2	Have knowledge of taking measurements, calculations, evaluations, valuations, surveys, studies, reports, work plans and other similar studies in IT.
CM1	Be able to design, develop, select and evaluate IT applications and systems, ensuring their reliability, security and quality, in accordance with ethical principles and the current legislation and regulations.
CM2	Be able to plan, conceive, implement and manage IT projects, services and systems in all areas, leading their start-up and ongoing improvement, and evaluating their economic and social impact.
CM3	Be able to understand the importance of negotiation, effective work habits, leadership and communication skills in all aspects of software development.
B1	Be able to manage projects within the field of IT.
B8	Be able to work in groups and in a multilingual and multidisciplinary environment.
CT5	Communicate information clearly and precisely to a variety of audiences.

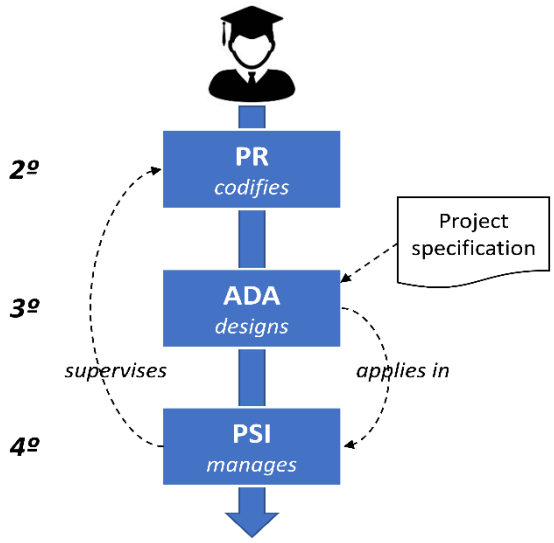

Fig. 2. General diagram of the integrated project.

Our aim is to give the students a practical and integral view of the whole development process and its interactions. More in detail, the project consists of developing a software application with a graphical interface and data persistence. Some examples of projects from past courses are:

- An application to manage the catalogue of a travel agency that allows customers to make and manage bookings.
- An application to order food online, which allows configuring and preparing menus.
- An application to manage different types of membership to a library, and allows members to perform searches and requests for loaning books.
- An application to manage the catalogue of an on-line computer shop, which allows customers to perform searches and personalize the configuration of desktop computers.

All these applications have common aims that correspond to the contents and learning results associated to the subjects PR and ADA. In a nutshell, the specifications of the projects consider the following aspects:

- Object Oriented Programming.
- Class inheritance and use of abstract classes.
- Class composition and implementation of collections of objects.
- Reading and writing sequential files, both textual and binary.
- Definition and management of exceptions.
- Design and implementation of a graphical user interface using Java libraries.

In the following, we describe in detail how the practical tasks of each of the three subjects involved in the project have been designed.

A. ADA's practical exercise

The project begins with the ADA's practical exercise. The students are provided with a textual specification describing, at a high level, the functionalities that should be provided by the application to be developed, the actors who may use them and several non-functional requirements related to usability, security and technology.

Throughout several incremental deliveries in the second semester of the third course, the students will develop the phases corresponding to the collection of requirements, analysis and design of the application (Fig. 1, green color).

In a first stage, the students must formalize the requirements of the application. As result, they obtain an UML use case diagram, the textual specification of each use case and a set of business rules. Then, they perform the analysis of these requirements by defining the entity class diagram (classes and attributes) and, for each use case, a sequence diagram that depicts the behavior of the objects involved in each use case and identifies the methods of the classes. With this, the students will obtain a solution to the functional requirements that is still agnostic to the technology.

The result of this first stage is delivered to the teacher, who highlights mistakes and asks the students to correct them so that they are not propagated to the design phase.

The design stage consists in defining the solution obtained during the analysis according to the implementation technology. The software always consists of a desktop application coded in Java with a Swing graphical interface and a relational SQL database. Therefore, the design implies defining the specific Java classes (attributes and methods), the appearance of the graphical interface (metaphor, dialogs and windows) and the database schema. The result is delivered again to the teacher who will again highlight possible mistakes.

The practical exercise is done in groups of two students in order to facilitate discussion and reach an agreement between different points of view. However, each member of the team is responsible of the design of a subset of use cases. Therefore, evaluation marks are independent.

B. PSI's practical exercise

The same students continue the project in the fourth course within the PSI subject (Fig. 1 blue color). In this subject, the students put into practice transversal competences related to the supervision of work teams. These involve personal aptitudes that are of great importance in their future jobs and that include planning, and resources, personnel, risks and conflicts management.

The students will be in charge of managing the project and supervising a team of 4 programmers of PR, who will implement the design proposed by the director in ADA (or a piece of it, according to the complexity of the project). The project developed in PSI follows that proposed in ADA, but with a more general perspective that allows the students to tackle project planning (time, resources and personnel) according the competences of the subject. In this paper, we describe the part of the practical exercise that implies the supervision of students from the second course. The ratio of students of PR and PSI (see Table 1) allows that, in most cases,

the supervision can be done by a single director, as it happens in a professional environment.

In the first phase, the directors explain the design of the application to the students of PR. For this, they should adapt to the (limited) background of the students of PR, who are only familiar with the basic elements of the class diagram. The development of the application is also adapted to their skills and background, in particular, the SQL database that is designed in ADA is replaced by textual or binary files. Then, the directors plan and assign the tasks to be done according to the identified use cases and the classes to be implemented. In this way, each student of PR will have clear and unique duties within the project. The initial planning also includes: a protocol for creating internal working documentation, planning of periodical meetings, the use of management and communication tools and defining the quality guarantees that should be fulfilled in the project.

The directors have access to an academic calendar detailing the contents of PR, so that they can plan development consistently with the programming skills that the students of PR acquire through the semester.

In a first phase, the directors are asked to write two documents for the team. The first one is a short questionnaire that allows the directors to know aspects that are relevant to the team, and that may influence the development of the project. The second one is a welcome document that describes the protocols and, in some cases, gives short manuals or references associated to the management and communication tools that the director plans to use during the development of the project.

The second phase encompasses the development of the project and relies on two fundamental tools. On the one hand, a forum, which is also accessible to lecturers of PSI and PR, in which the decisions made by the team are reported. On the other hand, the minutes of the meetings, which should collect all decisions and documentation that may be generated during the periodical meetings. These tools contribute towards integrating the team members in the project, monitoring possible deviations and planning countermeasures.

Once the students of PR deliver the implementation of the project (being successful or not), the students of PSI must perform a self-evaluation of the project and discuss any deviations that may have occurred with respect to the initial plan; they should also highlight and discuss the issues that may have happened. This is one of the most important aspects of the supervision work, because it allows the students to identity and discuss the issues that may happen in a real scenario and that have influence in the result of the project. Moreover, it allows them to assess their own supervision and justify its influence on the project.

C. PR's practical exercise

The students of PR must follow the design and plan proposed by their director. At the same time, they put into practice the competences related to programming. In particular, they are asked to follow the object-oriented paradigm (composition, inheritance, polymorphism) and to develop efficient code. The programming language is Java.

Each student is responsible for one part of the application, but they should be aware of the whole project. The evaluation of the practical exercise considers both their individual work together with their contribution towards the success of the project. In addition to programming, they should perform testing under the supervision of the director in order to validate the application (Fig.1, orange colour). Testing is proposed as a process that should be done through the whole project. As long as new classes are developed, those should be validated by means of unit tests. In a second phase, testing should be done on the integrated application in order to validate the interactions between different classes. Finally, they should validate the functional requirements detailed in the specification of the project.

This practical exercise has a duration of 7 weeks and it is developed in the second half of the semester, which is the time in which the students of PR have acquired the technical skills required to embrace programming. The students are organized into groups of 4 people of their choice. Once the teams are constituted, 1 or 2 directors from PSI and a reference lecturer from PR are assigned. The latter is in charge of supervising the interaction between the members of the team and the director and, also, to solve technical doubts related to the basics of programming of students from PR. They are also given several tools in order to facilitate collaborative development through the Moodle platform, in particular:

1. Each team (4 programmers + director/s) has a private forum that is also accessible to the reference lecturer. In this forum, the director will upload the minutes of each meeting so that every member of the team can ask for clarifications.
2. Each team has a GIT repository in a local server at the university. This allows them embracing programming in a collaborative and synchronized way. It also enables the lecturer to closely supervise the development, including the parts of the code that each student has programmed.
3. All students are given a template detailing the contents of the documentation they should deliver accompanying the developed application. Even though this documentation is done in collaboration with the director, it mainly influences the evaluation of the students from PR.

VI. STUDENT EVALUATION

In this section, we detail the evaluation criteria used in each subject.

Although the work in PR is done in teams, the evaluation of each student is individual. The evaluation has into account mainly the quality of the code implemented by each student, the testing set designed to validate the code, the pieces of the documentation written by each student, the teamwork and, finally, the answers obtained in an individual interview. Consequently, each student may obtain a different grade. This also enables evaluating students even when some of them abandon before finishing his/her assignment. Moreover, the students in PR evaluate the work done by its director when the project has finished.

In ADA, the work is done in pairs, but each student is responsible of some tasks. So, although the design of the information model is common, each student is in charge of a subset of the use cases. Therefore, the grades are individually given. The work has two deliverables: the first includes the

requirements and the analysis; the second is related to the design. After each delivery, the student receives comments and the possibility of correcting the mistakes to increase marks. As general criterion, the student cannot pass the practical exercise (and, therefore, the course) if his/her specification and design contain mistakes that may raise later problems when they present the design proposal to the students of PR.

The evaluation criteria focus on the correctness and details of the analysis and design, and the appropriate use of UML diagrams and of the software engineering methodology. Finally, the grade also considers the answers given in the interview with the lecturer. The ADA grade depends more on making a good design the first time, than in the final design, which may have been corrected with the indications received in the multiple revisions.

The evaluation of the work done in PSI is made based on several documents delivered during the course. For each evaluation of the management of time, scheduling and personal organization, the students must deliver two documents that include:

[1] The initial planning of the work and subsequent revisions performed during the development of the project.
[2] The team organization and follow up.
[3] Anticipation of possible incidences and mechanisms to solve them.

For the evaluation of "Proactivity and anticipation", as well as the capacity of "Conflict management", we use an activity analysis tool that is available during the execution of the project. Moreover, the grade includes the evaluation made by the students of PR.

With the goal of knowing the opinion of students about this proposed project, we have conducted two surveys, one for PR students and another for PSI. The surveys have also the purpose of detecting problems in order to implement corrective measures in next editions.

The Moodle virtual campus has been used to collect the answers to those questionnaires, using the tool "Feedback". It was anonymous and it is voluntarily made. Being voluntary, some students have not answered it, but being anonymous allows collecting more sincere opinions.

A. *Survey to students in PR subject*

The survey to students in PR course has 12 questions about the experience with the programming project. The questions were designed to be fast to answer and only 2 questions have an open answer. In those ones, the student must elaborate on the learning experience, more than on the technical concepts learned and, also, they can propose improvements to the project. Nine questions are single-choice, with four possible answers: *very high, high, low, very low*. These questions are about the deadlines, the tools (GIT for version control, forum to facilitate communication among the team), the attitude of the team members (commitment, deadline accomplishment) or about the satisfaction with the team members and the director, among others. The last question is multiple-choice and the students must select the aspects that they consider that have been done better with the help of director than without him/her. Options include: team coordination, accomplishment of the deadlines or better implementation.

B. *Survey to students in PSI subject*

Similarly, we have conducted a survey to the PSI students (who were former students in ADA). The questionnaire not only gathers the opinion of the direction work but also the experience of doing an integrated project that involves 3 subjects. The survey has 11 questions of single choice and two with an open answer.

About the supervision of the project, the students must evaluate their experience as directors, the degree of commitment of the students of the second course, including the planning, the results achieved or the communication in the team. There are also questions about the facilities they had to make a good direction of the project, such as validation of the code the students were developing or the minutes of the meetings. They must also think about the difficulties they faced.

An important question regards the degree of satisfaction with respect to the experience of working in an integrated project during 3 subjects through 3 years. In this case, the student must evaluate the advantages of using the design solution made in ADA as the starting point for the project developed in PSI. They must also evaluate if this integral project allows for a clearer perspective of the software development cycle.

VII. Results and Impact

In this section, we explain the grades obtained by the students in the Programming subject (PR), comparing them with previous years where the methodology we present was not applied. Next, we analyze the answers to the survey both for the PR students and the PSI students. Finally, we discuss the impact of the proposal.

A. *Grades in PR*

We have considered the grades obtained in the third practical exercise of PR (e.g. the one that is made under the supervision of PSI students) in the two courses before implementing the current methodology (i.e. when students worked without a director), and we compare the grades with the ones obtained in the courses where the integrated methodology was applied. To have a general view, we have calculated the average of the grades in these two scenarios and we consider percentages in order to facilitate interpretation. We have grouped the data of two courses before and after applying the methodology in order to minimize the impact of particular situations that may happen in a certain course.

Figure 3. shows the grades obtained in these two scenarios. We can see that the percentage of students that failed (or did not complete) the practical exercise decreases when the integrated methodology is used. In addition, the percentage of students that obtained a Sufficient qualification also decreased. On the opposite side, the percentage of students with Good or Excellent grades increased.

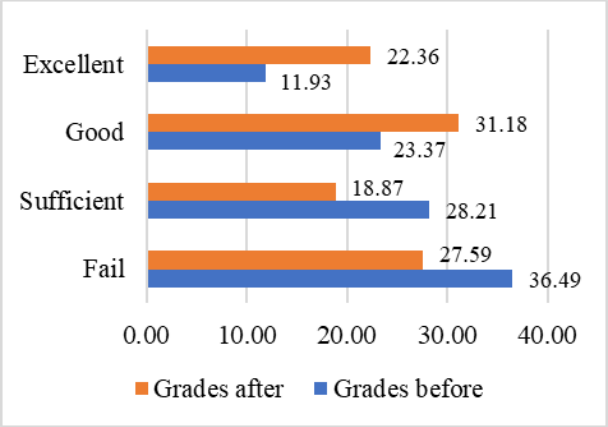

Fig. 3. Grades obtained before and after applying the methodology

These results are clearly positive: the grades have improved and the number of students that pass the course has increased. From the analysis of the methodology as well as the feedback of the students we have found some reasons for that improvement. On the one hand, having a student of the 4th course supervising students in the 2nd course has important implications, such as starting the work earlier, an environment that favours the continuous work (as the supervisors checks it regularly) and it avoids that some students abandon. On the other hand, the director has the responsibility of giving a correct design of the exercise, making an appropriate and balanced task distribution and following a well-defined time schedule. This enables the students of PR to concentrate efforts in the programming task. Finally, the continuous monitoring of their work forces them to be constant in their work and to have a sense of responsibility with the other teammates.

B. Analysis of the survey in PR

When evaluating the methodology, not only the academic results are important, but also the perception of the students who participate in the experience. For this purpose, we conducted the surveys presented in Section 6 and below we collect and discuss the results obtained.

We have collected 52 answers from students in PR subject. We have analysed each of the questions separately, but, in general we observed that positive evaluations (*very high, high*) are always more than negative ones (*low, very low*), except in two questions. They refer to the evaluation of the minutes of the meetings and to the use of GIT repository to share code. In the first case, the team of lecturers considers the minutes as a requirement, because they show the decisions taken along the project and are used by the lecturer to detect possible problems in advance. We have identified that the main problem for students is the lack of format of the content of the minutes. In this regard, we propose to prepare a guide for the content of the minutes, maybe as a predefined form with clear fields to fill in. Regarding GIT, we have detected some problems in its use. We will improve the user manual and documentation in the future. We know that learning this kind of tool for the first time is not easy, and it adds extra complexity to the project. For this reason, we are considering to dedicate soma laboratory sessions to improve the skills of the students in the use of this tool.

In the survey we had five questions related to the experience of working in an integrated project. In Fig. 4, we can see the results obtained.

The best rated aspect is the opinion of the director's work, followed by the satisfaction with the teamwork. In a middle range we have the fulfillment of the deadlines. On the other contrary, it is evident that almost half of the students consider that their classmates were not sufficiently committed to the work to be done. The worst assessment corresponds to the usefulness of the minutes of the meetings. We have analyzed the open answers, but no reference is made to this point. Therefore, we plan to continue with the modification stated in the previous paragraph.

Another of the key questions is related to the aspects that the students believe that have improved thanks to working with a director. The students had 7 options to choose a subset of them. The question was the following: *Choose among the following options, the ones that you have done better thanks to the director.*

The possible answers were:

1. Team coordination
2. Keep work up to date and meet deadlines
3. Design the program
4. Implementation of the code
5. Elaboration of the documentation
6. Avoid confusions in the development of the project.

The answer selected the most was number 6, by 61% of the students. It is followed by options 1, 2 and 3, with values of 44, 42 y 40%, respectively. Answer number 4 was only selected 30% of the students and number 5 by 15%.

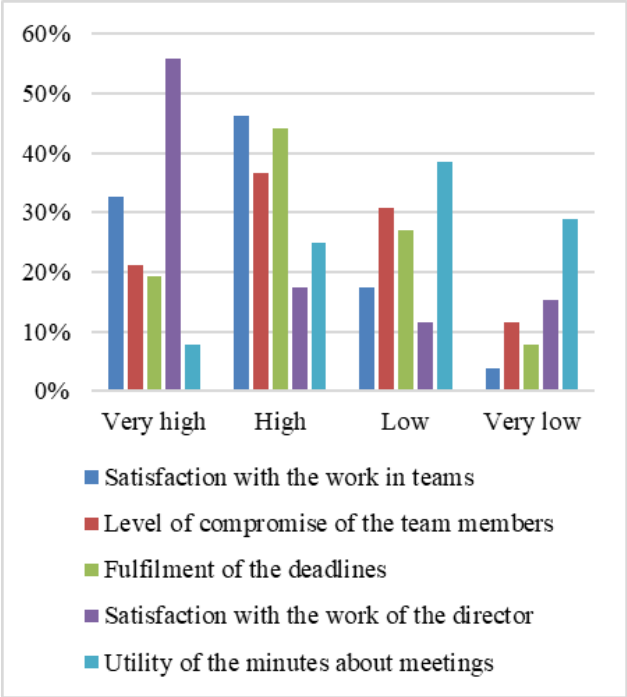

Fig. 4. Survey results on working in an integrated project in PR.

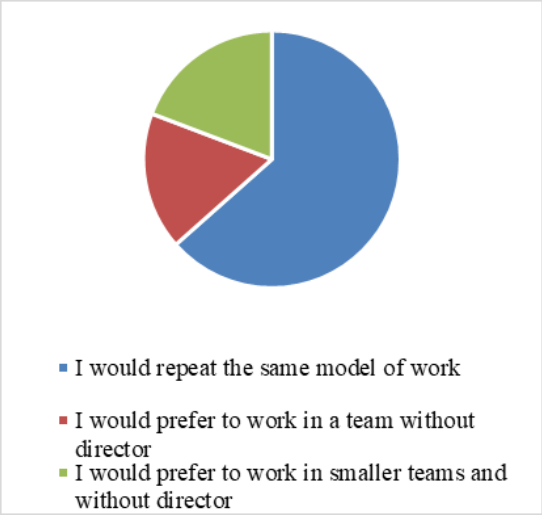

Fig. 5. Analyzing the working model.

Another interesting question is whether the working model would be repeated, or other similar alternatives would be preferred. The results are depicted in Fig. 5, where we can see that most students would repeat the same model.

We have also analyzed the textual answers given by the students on the two open-ended questions. The first question was about what the student considers that he/she has learned beyond the technical knowledge and the experience. The second question asked what improvements the student would introduce to improve the project, from his/her point of view.

In these two questions we find positive answers ranging from learning shared responsibility, to the feeling of learning how to follow a plan on a project, and to the motivation that arises from feeling that their work influences the team as a whole and the success of the project. The number of positive opinions is much larger than the number of negative opinions.

The negative reviews have focused on teams where communication has not worked and problems appeared between the director and the programmers. Specifically, a minority have considered that the indications of the director have harmed them.

A more detailed analysis relates the responses of the different questions so that we can see dependencies between the answers to some questions when they have answered with negative (or positive) opinions in others. In general, the students that consider that the director did not perform a good work, they also say that they prefer to work without a director. We can also find a dependency between the answers about the fulfillment of deadlines and the question about the compromise of the team members. We can see that the negative evaluations do not depend on a unique factor, but they depend on the aggregation of several problems inside the team.

C. Analysis of the survey in PSI

We have collected 17 answers from the students in the PSI subject. We have analyzed each question separately and the percentage of positive evaluations was always higher than the negatives ones in all questions, except the one related to the usefulness of writing meeting minutes.

First of all, we take a look at the answers related to the satisfaction with the direction work. They were asked about the interest of the experience of leading a team with 4 students of the PR subject, about the easiness to contact with the team members and the use of the minutes. The results are shown in Fig. 6.

The element with negative opinions is again related to the minutes. After reading the answers to the open questions, no student justifies the lack of usefulness of the minutes. In fact, they do not mention this issue at all. In any case, we will revise the instructions about the minutes to simplify their use.

The biggest problems stated in the open answers are related to the lack of involvement of some students of the team. The survey has a specific question about this aspect, where 30% of the directors say that PR students were poorly committed with the work, 53% say that they were sufficiently engaged and only 17% are totally satisfied with the level of engagement of the programmers. There was another question related to the previous one; in that question they should indicate if the deadlines given by the director were fulfilled. To this question, 23% state that they were not satisfied with the fulfillment at all, 17% that the deadlines where poorly followed, 53% are quite satisfied with the fulfillment and only 7% are very satisfied with the work of the PR students.

The students of 4th grade had also a question about the utility of the GIT tool and how it can help in the supervision of the work. In this case, 41% of the directors are very satisfied with the use of this took, 24% are moderately satisfied and only 35% are not satisfied. This change in the evaluation of this tool in comparison with the one of the students of 2nd course may be due to the fact that the students in the 4th course are more used to work with these tools as they are used in other subjects too. However, for students of the 2nd course, it is the first time and they find more difficulties in working with shared code.

In the survey, a question was devoted to analyse the difficulties found by the director in his/her role. The question was a multi-choice one and the results are displayed in Fig. 7. We can see that the main problems are in the supervision of the documentation. In general, the directors say that the students completed the documentation too late and, consequently, they had few time to revise it.

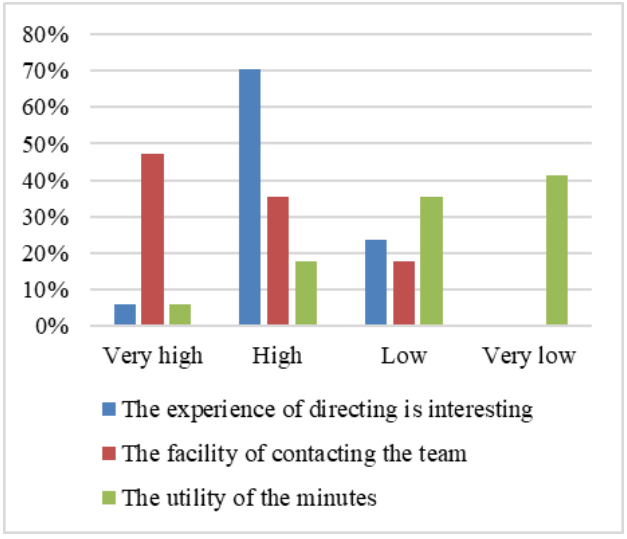

Fig. 6. Analysis of answers related to the satisfaction with the direction work.

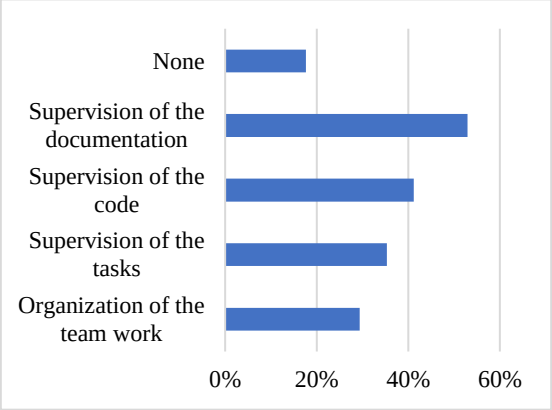

Fig. 7. Difficulties found by the director in his/her role.

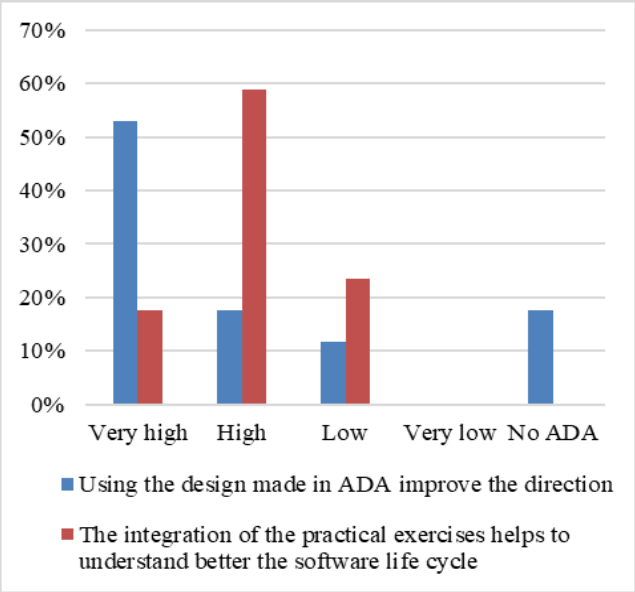

Fig. 8. Satisfaction of integrating the practices of different subjects.

Another aspect that we have analysed regards the experience of integrating the practical part of 3 different subjects. In the survey, we have included two questions about this issue and the results are shown in Fig. 8.

The first question, shown in blue in the bar chart, is about the advantage resulting from using the design made in ADA in the direction of the project and the team of programmers. The second question asks the opinion about whether the integration of the three practical exercises helps to understand better the software life cycle or not.

In Fig. 8, we can see that the students are greatly satisfied with the methodology, giving very positive answers.

It is worth to note that there is a subset of students that do not carried out ADA the year before but in some previous years, in that case when they are in PSI the project is about a different topic than the one they solved in ADA. This is the case shown in the last bar, denoted "No ADA", which corresponds to only 3 students.

D. Impact

Given the results, both academic and personal opinions, we consider that the integrated project has had a very positive impact on the training of computer engineering students.

From the surveys, we can conclude that the experience acquired throughout the different subjects and roles has favored learning personal values, not only specific competences to the profession, but also transversal skills. This is the feeling expressed by the students in the collected answers. They consider to have improved in:

[1] Being aware of the work performed at different levels of the process.

[2] Knowing the importance of good design, good planning and proper follow-up and risk management.

[3] Learning to work as a team with people with different personalities and different knowledge levels, either as a director or as a programmer.

[4] Understanding the importance of committing to the team and completing the tasks within the stipulated period.

[5] The experience has also had a positive impact at the academic level, which can be seen in the success rate and the grades obtained in PR.

VIII. CONCLUSIONS

After the experience of deploying this integrated project during 4 courses, and looking at the results presented in this paper, we consider that the proposed model is highly positive both from the academic point of view (with a significant improvement in the quality of the programming, which in turn improves the grades in PR) as well as from the point of view of the training of competences in the 4th grade (mainly about leadership, teamwork, planning and work commitment).

We find it interesting that the same exercise serves as a practice in several subjects, instead of working with different practical problems independently. In this way it is easier to see the relations between software creation and development steps, obtaining an overall view of this area of Computer Engineering. From the point of view of the Bachelor study plan, the proposed methodology helps also to improve the vertical coordination among the three subjects participating, avoiding overlapping or gaps in the content given to the students.

As general recommendations collected throughout the years of implementation, we want to highlight the need to provide clear instructions on the roles of each student in each subject. Also, during the management-development phase, it is important that students have a teacher in PR as reference tutor, in order to help to solve incidents related to student behavior in the team.

Finally, it is worth mentioning that we have observed some collateral advantages derived from this experience. In particular, the students in the 2nd course think that knowing other students from 4th course has been positive, it has helped them to have a broader view of the studies and understand the relations between different software courses. Moreover, we have detected that the project improves the participation of students that usually would be isolated.

ACKNOWLEDGMENTS

The authors acknowledge the financial and technical support of the School of Engineering and the Department of Computer Engineering and Mathematics of the Universitat Rovira i Virgili.

REFERENCES

[1] Adecco, "Empleos tecnológicos en el mercado laboral español," April 2016. Available at: https://adecco.es/wp-content/uploads/notas-de-prensa/764.pdf

[2] R. Colomo-Palacios, E. Tovar-Caro, A. García-Crespo and J.M. Gómez-Berbís, "Identifying Technical Competences of IT Professionals: The Case of Software Engineers," *International Journal of Human Capital and Information Technology Professionals*, vol. 1, pp. 1-13, 2010.

[3] Brian Fitzgerald, "Software crisis 2.0," *IEEE Computer*, vol. 45, pp. 89-97, 2012.

[4] M. Ferré, C. García-Barroso, M. García-Famoso, D. Sánchez and A. Valls, "Mejora de la formación en el diseño y desarrollo de software a partir de la coordinación de distintas asignaturas", in *Actas de las XXV Jornadas de Enseñanza Universitaria de Informática, Jenui 2019*, pp. 23-30, Murcia, July 2019.

[5] J.W. McManus and P.J. Costello, "Project Based Learning in Computer Science: A Student and Research Advisor's Perspective," *Journal of Computing Sciences in Colleges*, vol. 34, no. 3, pp. 38-46, 2019.

[6] A. Breiter, G. Fey and R. Drechsler, "Project-Based Learning in Student Temas in Computer Science Education," *SER.:ELEC. ENERG.*, vol. 18, no. 2, pp. 165-180, 2005.

[7] I. Calvo, J.M. López-Guede and E. Zulueta, "Aplicando la metodología Project Based Learning en la docencia de Ingeniería Técnica en Informática de Gestión," *Revista de formación e Innovación Educativa Universtiaria*, vol. 3, pp. 166-181, 2010.

[8] F.O. García, D.G. Rosado, M.A. Moraga and M.A. Serrano, "Formación integral en la intensificación de Ingeniería del Software en el grado en Ingeniería Informática," in *Actas de las XXIV Jornadas de Enseñanza Universitaria de Informática, Jenui 2018*, pp. 197-204, Barcelona, July 2018.

[9] J. Ruiz de la Peña, L. Lamoth-Borrero, M.R. Concepción-García and F. Rodríguez-Expósito, "El proyecto integrador como experiencia didáctica en la formación del ingeniero informático: Universidad de Holguín, Cuba (UHOLM)," *Escenarios*, vol. 10, pp. 106-115, 2012.

[10] P. Sánchez, C. Blanco, A. Pérez, J. Medina, P. López, A. de la Vega, D. García and M. Sierra, "Experiencia y Lecciones Aprendidas durante el Desarrollo de un Proyecto Software Común a Diversas Asignaturas," in *Actas de las XXIII Jornadas de Enseñanza Universitaria de Informática, Jenui 2017*, pp. 291-298, Cáceres, July 2017.

[11] R. Asenjo-Plaza, S. González-Navarro, F.J. Corbera-Peña, A. Navarro, A. Rodríguez-Sabadell, J. Villalba and E. Hendrix, "La plataforma Raspberry Pi como base para la coordinación vertical," *Enseñanza y Aprendizaje de Ingeniería de Computadores*, vol. 7, pp. 5-20, 2017.

[12] I. Jacobson, G. Booch and J. Rumbaugh, "The unified software development process," Addison-Wesley Longman Publishing Co., Boston, USA, 1999.

María Ferré got a MSc and PhD in computer science by the Polytechnic University of Catalonia in 1990 and 2003, respectively. She has authored several papers in journals and national and international conferences. She has participated in several research projects in the field of computer graphics and has conducted technology transfer in the areas of medicine and nursing. (e-mail: maria.ferre@urv.cat)

Carlos García Barroso got a MSc in computer science by the Polytechnic University of Catalonia (1992). He has authored several papers in international journals and conferences in the areas of computer vision and teaching (e-mail: carlos.garciabarroso@urv.cat)

Montse Garcia-Famoso got a MSc in computer science by the University of Deusto (1994). Her interests include teaching methodologies and tools applied to computer engineering. (e-mail: montse.garcia@urv.cat)

David Sánchez got a MSc and a PhD in computer science by the Universitat Rovira i Virgili (2003) and the Polytechnic University of Catalonia (2008), respectively. He has authored more than 150 papers in journals and international conferences. He has participated and coordinated several national and international projects in the areas of artificial intelligence and information security. (e-mail: david.sanchez@urv.cat)

Aïda Valls has a MSc and a PhD in computer science by the Polytechnic University of Catalonia in 1997 and 2002, respectively. She has authored more than 160 papers in journals and international conferences. She is the coordinator of the doctoral program in computer science at the Department of Computer Engineering and Mathematics of the URV. She is also the mobility coordinator of computer engineering students at the URV. (e-mail: aida.valls@urv.cat)