PRACTICING THE SOFTWARE DEVELOPMENT PROCESS

Angelo Martins
António Costa
Alberto Sampaio
Paulo Sousa
ISEP – Instituto Superior de Engenharia do Porto
Porto Polytechnic Institute – Engineering College

Abstract
The Bologna Declaration principles adoption in Portugal led to a complete overall of university degrees, so a new LEI-ISEP (baccalaureate degree in Informatics Engineering) was proposed which broadly incorporates the CDIO paradigm and many of its techniques and suggestions, as well the ACM guidelines for computing curricula.
A key requirement of CDIO is the students' ability to autonomously conceive and implement systems, using industry best practices. To this effect, five multidisciplinary project courses were created, one in each of the first five semesters, complemented by a capstone project/internship. In this paper we present the rationale behind these courses and describe the actual implementation and results of three of them (LAPR1, 3 and 5), that took place in the first semester of the 2007/2008 school year.

Keywords: multidisciplinary, project courses, software development process, CDIO

Introduction

The Bologna Declaration principles adoption in Portugal led to a complete overall of university degrees, though often the changes were little more than curriculum cosmetic adjustments. LEI-ISEP (baccalaureate degree in Informatics Engineering) was decidedly an exception, as it was decided to go a bit further and implement also the CDIO approach for engineering teaching, having successfully applied for a Portuguese Government grant. So, a new LEI-ISEP (baccalaureate degree) was proposed which broadly incorporates the CDIO paradigm and many of its techniques and suggestions, as well the ACM guidelines for computing curricula [1].

A key requirement of CDIO is the students' ability to autonomously conceive and implement systems, using industry best practices [2]. To this effect, five multidisciplinary project courses (LAPR) were created, one in each of the first five semesters, complemented by a capstone project/internship. It is now quite common to include this kind of course, also commonly referred as "integrating courses", in engineering programs adapted to the Bolonha Declaration principles in Portugal. The most common approach to implement these courses is to use them for several small interdisciplinary projects development, usually in close connection with the other courses in the same semester.

This approach is useful, but we believe that better results can be attained if we decoupled these courses from other courses and give each of them an individual purpose and give the whole set a common purpose and set of objectives. LEI-ISEP core areas are computer science, networks and software engineering, thus is was only natural that the chosen common purpose for the LAPR courses was the mastering of the software development process, a key competence of the software and computer engineer. The software development process integrates technological knowledge in each of the program's core areas with soft skills such as teamwork, process quality and control and project management. Figure 1 presents the rational behind the Lab/Project courses (LAPR) approach to software development teaching: a three step, spiral-like approach based on market defined professional competences.

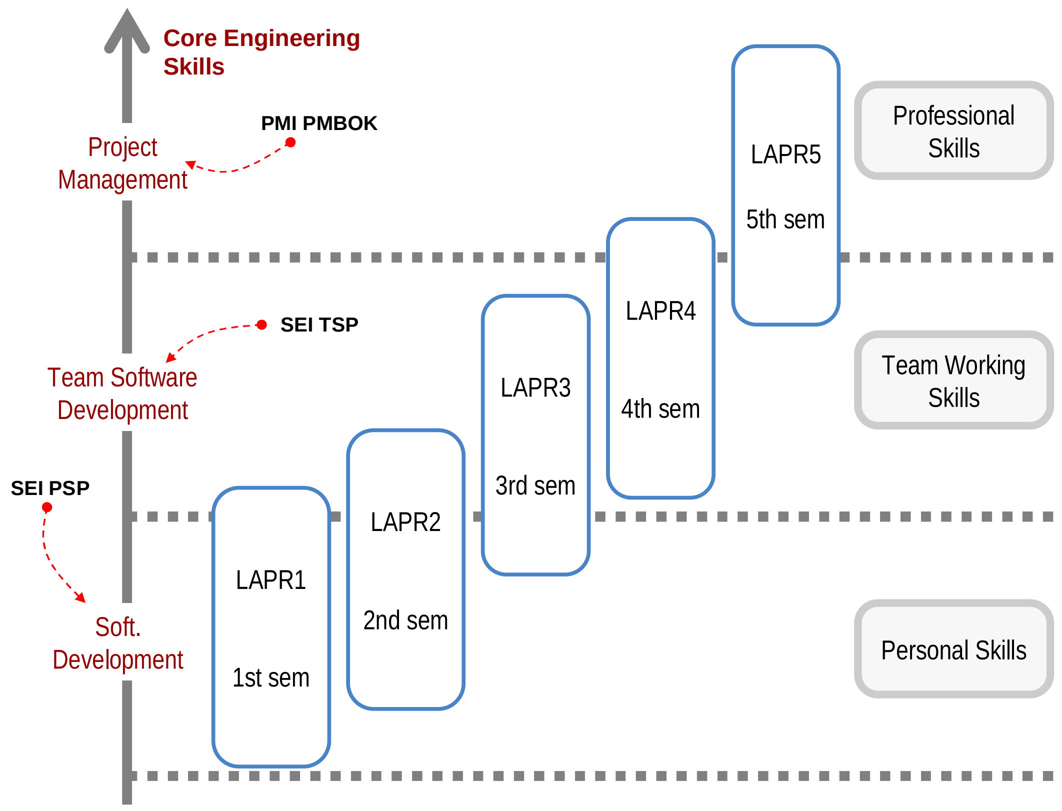

Figure 1. Lab.-Project (LAPR) courses in the curriculum

The first two courses LAPR1 and LAPR2 focus on basic software development skills, using the widely known Software Engineering Institute (SEI) Personal Software Process (PSP) as reference [3]. In the 3rd and 4th semesters, the focus moves steadily to team working skills, while in the 5th semester the students are presented with the "system view" of the development process. To this effect, project management is the core subject of LAPR5, preparing the students for the internship/capstone project in the last semester.

The first four LAPR courses have two components, a 12 week (1,5 hours/week) class based module where a set of competences are introduced (time management, team working, writing skills, law, etc.), and a 4 week intensive project development module (19 to 24 hours/week) where the students (groups) have to implement a software project.

The multidisciplinary aspect of the LAPR courses arises from the fact that the project of each LAPR is based on the subjects thought on the other courses of the same semester. Furthermore, most of the teaching staff involved in the LAPR courses is picked up among the teaching staff of those courses, acting mostly as tutors, counselors or even simulating the role of clients, as is the case of LAPR5.

LAPR5 differs a little from the other LAPR courses, as there is no 12 week competence module; just a week crash course on project management concepts and the project is centered on the implementation of real-like software project under the guidance of project management specialists.

The LAPR courses structure

A basic common structure was defined for the five LAPR courses in the LEI-ISEP program:

- Competence module. In LAPR1 through 4, this module is lectured in the first 12 weeks of the semester and represents about 20% of the course's classes and 35% of the course's credits. In LAPR5 the module and project mostly overlap in time. The subject of the module is related to the course's objectives.
- Project. The project represents 65% of the course's credits and it takes place in the last 4 weeks of the semester (a semester has 16 weeks), in an intensive mode (somewhere between 16 and 24 hours/week, depending on the particular LAPR course). There is only one big project and it is the same project for every student. The students are grouped in groups of 2 or 3 (LAPR1, LAPR2), 3 or 4 (LAPR3 and LAPR4) and 5 (LAPR5).

Though LAPR is an autonomous course, the project's subject and objectives are defined in close coordination with the semester's other courses. Whenever possible, the teaching staff involved in a LAPR course is also involved in the semester's other courses. This is important, because the teacher's sole role is to be a technical advisor/tutor.

LAPR1

LAPR1 belongs to the first semester of LEI-ISEP and the students only have a limited set of competences in computer programming. Nevertheless, coding is just one of the tasks involved in software development and a lot can be learned with a simple example, especially the discipline and the method that should be applied in any engineering process.

Following state of the art recommendations [4], a detailed eight page guide was provided to steer the students/groups in this first project. Along with the guidelines and rules, penalties were defined, to encourage the students not to break the rules. For example, in the case of a group breakup during the project, a 30% penalty on each student's project mark would be applied to every group member. The chance of any of the students get approved on the course would them be rather slim. This is important because in real life environment project teams don't breakup due to futile reasons. Of the over 90 groups of this year's edition, there was only one breakup. In the last year's edition, in a striking contrast, there were over a dozen breakups.

This year's project theme was the development of a Java console application to make matrix operations (product, addition, transpose, inversion, determinant, etc.). For data input/output was mandatory the use of well formed HTML files, the matrixes being represented as HTML tables. A standard file format was defined and testing examples provided to the students.

Along with the base requirements, which accounted for 80% of the project's final mark, a set of optional/elective requirements were provided, each worth 10%. These included: the use numbers in any numerical base between 2 and 16; automatic HTML file structure validation/debugging; step by step determinant calculation and report in a HTML file; etc. The groups could implement as many as elective requirements as they wished, but they would get marks only on two of them.

Project/group monitoring and assessment

Each of the 93 groups was provided a tutor (one of the course's 23 teachers), being mandatory a formal weekly meeting between the tutor and the group. The minutes of this meeting should then be written on the group's logbook and signed by everyone present. The purpose of this meeting was to plan the week ahead and to help solve eventual problems. Technical problems were not required to be addressed in this meeting, as the students would always have immediate access to a teacher in the 24 hours/week project classes.

The software development process, although in a simplified implementation, was the course's main objective, thus the group's tutor was a key element during the project and in the group's final assessment. The group's assessment was based on the developed software application, the group's project report, the group's logbook and the tutor report. The application was demonstrated to a 3 element jury, which included the tutor, and the jury directly quizzed every element of the group on some of the application's details. A 30 minute slot was scheduled for each group's assessment.

Of the initial 93 groups, 81 delivered a project. Only 5 of these projects failed to get the minimum mark (10/20) and the average score was 13.6. Two groups got 19/20. Though the assessment was group based, an individual project mark was given to every student, based on his performance during the project presentation and the tutor's records. About to half of the students got different individual and group project marks. This mark differentiation within the group is likely to increase in next course editions.

LAPR3

LAPR3 intends to give a more global and complete view of software development than its predecessors. So, it tries to cover all the software developments phases in a more explicit way, to include distinct domains than the more obvious to the students. Because almost all software projects are team projects, group work, communication and presentation are also included and learned during LAPR3.

Like the other LAPR, LAPR3 have two components, a 12 week (2 hours/week) class based module where competences are introduced and a 4 week intensive project development module (19 to 24 hours/week) where the students (groups) have to implement a software project.

The first module aims to develop competences in group working and communication. Communication includes development of public presentations. All these competences are explored in the project developed during the second module. The LAPR3 project was developed in groups of three students in order to implement group work and communication inside the team and with others, like teachers and colleagues.

LAPR3 project is based not only on the subjects thought on the other courses of the same semester, Databases; Computer Architecture; Information (data) Structures and Physics, but also in the computing subjects learnt in previous curricular years, like software engineering concepts and the topics covered during the first 12 weeks. LAPR3 project involved 21 teachers, being 17 of them from the teaching staff of those courses, and 4 not from that teaching staff. They act mainly as tutors and simulating the role of clients. Initially there were 286 students.

The teachers try to guide students but project development is left to students. The motivation for this, was to develop the students develop the ability to autonomously conceive and implement systems (a CDIO requirement). That was also achieved through the statement of the problem which possessed some ambiguity, but in a very controlled way as explained next.

The problem presented to the students was about an acrobatic air race. The students were asked to develop a system to manage that kind of competitions. The students were pointed to the rules of a very well known air race. Some other requirements were discriminated, so the students should be responsible to determine all the requirements. For example, the discriminated requirements made reference to the final phase of the competition, but said no word about the qualifying sessions.

The problem statement was presented in two documents, one with the problem itself only, and another document with a list of the tasks and deadlines. The problem document was extensive with 5 pages. The tasks document was 3 pages long. The students were forced to make deliveries every week, as explained in the tasks document. Because we have classes at the end of the day for student-workers, all deliveries were scheduled till Saturday 13h. All deliveries were classified with a mark and students informed about them. There was a strong feedback to students about their first delivery, which contained several documents (e.g. use cases, requirements list with priorities and responsible for each one, data model, etc.)

A complete list of requirements was asked for the final of the first week, but an analysis of the physics component of the problem was asked only for the second week. This difference forced the students to change the initial list of requirements at the end of the second week. Physics were incorporated through air security requirements. Physics teachers act like aeronautic specialists mainly during this second week.

We believe that this was very important to stimulate the students' autonomy. One of the most important characteristics of LAPR3 is to help students develop their autonomy. At what concerns autonomy and complexity, LAPR3 represents a significant leap from its previous LAPR. At the beginning some students did not react very well to all the ambiguity and forced autonomy, but with the development of the project things changed, and, at the end, students claims almost disappear. It is important to refer that some other students react very well from the beginning. At the end of the project we asked students to give their opinions about what could change for next year.

LAPR5

On the 5th semester of the course, students are ready to be exposed to complex systems' building. The LAPR5 module is intended to give students contact with a reality of system of systems as opposed to a single system view they have been exposed to until now. The module works as a public tender where the students must form a group to simulate a company and answer a *Request for Proposals* (RFP). In parallel the students have a theorical-pratical module (30 hours) on project management (as part of the personal/professional skills modules).

This module worked for 4 consecutive weeks *after* the end of the other modules of the semester. During this time students had 16 hours with teacher support dedicated to the Lab Project plus 4 hours dedicated to the personal/professional skills module.

Personal/professional skills module

This is a "regular" module on project management based on the PMBOK (Project Management Body of Knowledge) with focus on the planning processes due to time constraints.

Project

The lab project for this module must make the students aware of the reality that a software system is most of times composed of several elements (e.g., applications, databases) sometimes in heterogeneous environments (be it operating systems and/or programming languages). Furthermore, the students must also gain awareness to the hardware infrastructure that is needed (and the associated operating plans) to run the system, so that students drive away from a single-application, software-only, mindset.

From an operating point of view, this module worked with teams of students with 5 members, each team representing a company answering a Request for Proposals issued by a software development firm (the customer). There was also a group of teachers representing a consulting company that the customer has commissioned to evaluate the proposals and clarify the requirements.

The module worked in tight conjunction with the other modules of the semester and the project theme was mainly focused on applying the knowledge of those modules, but students should apply every knowledge they have gained from the beginning of the course. The other modules of the 5th semester are:

- ARQSI – component and web development (asp.net)
- ASSIST – IT administration (windows and linux)
- ALGAV – introduction to artificial intelligence (mainly searching and planning)
- SGRAI – computers graphics (OpengGL) and Human-Machine Interfaces

The consulting team had a teacher from each module (the module's main responsible) that would meet with all the students teams on the beginning of the week. This session was mainly for the teams to pose questions regarding the scope and requirements of the problem and not to solve technical implementation details. Throughout the week, the students had tutorial lessons with teachers from each of the areas to help with technical problems.

This year the project theme was about Airport Management Systems and the RFP requested:

- A prototype implementation of an integrated system with
    - Gate scheduling component – online/offline scheduling engine for the airport gates
    - Front office gate scheduling viewer and what-if simulator – allows the airport manager to view the current status of the gates and prepare for necessary reschedules due to flight delays, gate breakdowns, etc.
    - Back office administration – simple master table administration web application
    - Schedule publication web service – anonymous web service for publishing current information on scheduled flights and authenticated web service for publishing detailed scheduling information
    - Flight incident simulator – 3D simulation for use by the airport fire department to simulate best fire fighting strategies in case of airplane fire and building evacuation
- An infra-structure plan for running the proposed system along with operating and contingency plan

Results

The weekly joint session was important in keep everybody (including teachers) in sync to what respects the expected output of the projects. Nonetheless, students didn't quite grasp all the requirements and most of them presented some modules which didn't comply with the customer's expectation. This was mostly due to misunderstanding of the module objectives since this was the first time this students had this kind of module mechanics. Some students still had the mindset of single application, no software engineering practices which translated, for instance in the flight incident simulator module, in a flight simulator which would be of no use to a fire department chief (the intend user of that module).

The same thing happened to some extent in what regards the integration of the system; for instance, the flight incident simulator had no integration with the other modules in most of the teams even though the simulator was intended to simulate the same airport that was configured in the rest of the applications and as such should use information about gate layouts and flight schedules to run the simulation.

All in all, this was a positive experience mainly because of exposing the students to:

- Team work
- Time pressure
- Integrated systems
- Infra-structure planning

In what respects evaluation the students were evaluated by a jury of 5 teachers in what respects the lab project and a regular multiple-answer written exam for the project management module. The lab project evaluation was divided among different criteria such as documentation, presentation, functionality, technical solution.

Conclusion

In this paper we presented a practicing approach to learn the software development process in a higher education software and computer engineering program in Portugal. A set of multidisciplinary courses, one in each semester, were aligned to provide a coherent package, starting with a basic introduction to software development in the first semester of the program and finishing with the simulation of a market-like *Request for Proposals* in the fifth semester. With this five semester training, the authors believe the student can then face the capstone project/internship with confidence and solid fundamentals. In theory, the software development process can be taught in a couple of lecture-based courses, with lectures and a few hands-on classes. But will then the students' final competence surpass by far the one they could get by reading a good pair of books on the subject?

The authors believe there is no substitute for life-like experience in learning, with all the hurdles and limitations one can't find in books and in lectures (there are actually to conventional courses devoted to software development process in the LEI-ISEP program). For example, early design choices may lead to dead ends or product limitations. In all LAPR courses students can negotiate penalties with teachers (within predefined limits), in order to avoid major rewriting/redesign of their product.

It should be mentioned that the use of multidisciplinary project based courses is still regarded with some lack of confidence by some conventional course teachers. This lack of confidence is aggravated by the difference in student approval rates between project and lecture-based courses. There are certainly some students that are able to successfully complete project courses on the expense of their group's colleagues. This is a fact of life and one can also argue that cheating also exists in lecture-based courses. The tutor's role in the LAPR courses was designed to alleviate some of these problems, but nothing can eliminate them completely.

Finally, a very interesting *side effect* of these project courses is its impact on the teaching body. Course teachers were seldom used to cooperate with colleagues from other courses, resulting in what could be described as *islands of teaching*. The use of multidisciplinary courses greatly improves teacher cooperation, but at limited level if the actual projects don't really mix subjects from different areas. The one big project approach used in LAPR courses and the teacher pool for group tutoring works *wonders* in terms of teacher cooperation and even personal relationship, especially between teachers of different departments (Maths, Physics) which in many cases hardly knew each other.

References

[1] Association for Computing Machinery, Curricula Recommendations http://www.acm.org/education/curricula-recommendations

[2]. Karl-Frederick Berggren et al., CDIO: An International Initiative for Reforming Engineering Education, www.cdio.org/papers/papers.html

[3] Software Engineering Institute, Carnegie Mellon, http://www.sei.cmu.edu/tsp

[4] J. Malmqvist et al., Lessons Learned from Design-Build Test-Based Project Courses, Presented at Design-2004, Dubrovnik, Croatia, May 2004.

Corresponding author
Angelo Martins, PhD
Departamento de Informática – Instituto Superior de Engenharia do Porto (DEI ISEP)
Rua António Bernardino de Almeida
4250 Porto, Portugal

Phone: +351 228340500
amartins@dei.isep.ipp.pt

Angelo Martins is the LEI-ISEP program director and the responsible for the LAPR1 and LAPR2 courses. He is an electrical and electronic engineer with a PhD in industrial automation. He has a 15 years teaching experience at ISEP.