PROJECT-BASED LEARNING APPROACH IN A COLLABORATION BETWEEN ACADEMIA AND INDUSTRY

Angelo Martins

Informatics Department, Porto Polytechnic – School of Engineering (ISEP)
Information Systems and Computer Graphics Unit, INESC- TEC

Alexandre Bragança, Nuno Bettencourt, Paulo Maio

Informatics Department, Porto Polytechnic – School of Engineering (ISEP)

ABSTRACT

The traditional disciplinary academic environment and learning practices do not provide a rich enough environment for deep learning of software development practices. Thus, to provide a richer learning environment, in the 2015/2016 school year, an interdisciplinary project-based learning (PBL) pilot approach was introduced where all the courses of the same semester focus on a complex software project provided by a software house. This paper describes the motivation, the concept and presents some qualitative results.

KEYWORDS

CDIO, PBL, Deep Learning, Standards: 3, 5, 7, 8, 9, 11.

INTRODUCTION

Market requirements for Software Engineering (SE) graduates have been changing at a very fast pace. One of the reasons for such is due to SE being a relatively young subject. In fact, many ICT programs pay little attention to the industrial software development best practices, focusing instead on programming languages, algorithms and trendy subjects like, for instance, Artificial Intelligence. Yet, the software is pervasive in modern society and there is a huge demand for software industry professionals, *i.e.,* Software Engineers. In this respect, Europe alone is demanding hundreds of thousands (Hüsing, 2015) of such professionals. Like any other engineering subject, learning SE requires some practice in a real or simulated environment. CDIO is a natural choice for the design, implementation and operation of a SE program.

The Informatics Engineering programs at Instituto Superior de Engenharia do Porto (ISEP) have over 10 years' experience in the application of CDIO. Both the Bologna 1st cycle (LEI) and the master (2nd cycle) have an EUR-ACE accreditation. Moreover, the master was also accredited by ABET in 2017 and both are highly regarded programs by the industry. LEI is the

largest Computer Science / Informatics Engineering program in Portugal with over 200 graduates per year, which are sought after by both national and international companies.

Even so, LEI's program management recognized that the traditional academic environment and learning practices do not provide a rich enough environment for deep learning of SE practices. Following CDIO standard 5, LEI has a 4-week long design-implement course in each semester, providing students with a short team-based product/system-oriented development experiences designed and operated by faculty. Moreover, an iterative approach is used, as it is now common in the industry, but the short time-span doesn't allow for more than 2 or 3 iterations with limited scope. It's too short, too fast, so that it doesn't foster reflective observation the way it should. It was evident that some of the outcomes were not going further than the "apply" level (Bloom level 3).

It would be interesting to provide a learning environment where the students could face the kind of requirements they face in professional practice and, very important, that they had the opportunity (time) to face and learn with the consequences of their choices. Thus, to provide this richer learning environment, in the 2015/2016 school year, an interdisciplinary project-based learning (PBL) pilot approach was introduced where all the courses of the same semester rely on a single complex software project provided by a software house. All students' activities and assessment should be in the scope of the development of this project. Internally, this pilot is called CDIO Integrated Learning (CDIO-IL).

CONTEXT

In order to better understand the motivation behind the CDIO-IL approach, it is important to introduce the reader to the professional software development area, which is a fairly new branch of engineering. Until the 70s, computers were very expensive centralized machines with limited capacity. They were used in science and in big business to solve specific tasks so that the number of professionals programming computers was small and many of them had another background (e.g. Math, Engineering, Business).

During the seventies and eighties, there was a boom in cheap computing, especially with the introduction of IBM PC clones running MS-DOS. For the first time, millions of PCs were produced yearly and introduced to small businesses and home users. The need for software developers rose accordingly, resulting in the worldwide creation of specialized higher education programs with names such as Computer Science (CS) and Computer Engineering (ACM, 2019). Later, Software Engineering. Even so, computers were mostly disconnected from each other or connected only in local networks. Software was composed of large standalone applications that were commonly distributed in a physical form. Cooperative software development was managed as a centralized hierarchical activity. As such, higher education programs focused on standalone applications programming and individual programming practices.

The advent of the commercial Internet and the World Wide Web in the nineties, changed the way how software is developed and distributed and how computers are used, i.e. connected. Almost all software is made collaboratively by teams, many of them encompassing different continents. Software also becomes increasingly interdependent, i.e. relying on services provided by other software to achieve its mission. The centralized hierarchical approach no longer can be applied in such a distributed and decentralized environment in continuous evolution. The weekly build no longer applies.

One current trend in software development is Continuous Integration that Fowler (2006) defines as "[…] a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including tests) to detect integration errors as quickly as possible".

This kind of environment requires software developers to have good "classical" design and programming skills, but also requires them to have practice in coordinating development with other team elements. "Competence" and "practice" are the key success elements here, as this agile development methodology is not viable without those throughout the whole team. To stress this, one well known agile methodology is actually named "scrum" (Suntherland, 2014) after the scrummage in rugby, which requires the cooperation of all players.

Higher education programs in the computing area must prepare the students to work in this collaborative/cooperative, complex and demanding environment. IT engineering programs can gain a lot from adopting the CDIO framework (Costa et al., 2012), as it goes much further than pure technical requirements and promotes a much broader view of the engineering world's requirements (e.g. sections 3 and 4 of the CDIO Syllabus). Furthermore, in IT programs it is easier to fully implement the CDIO framework, as software development doesn't have the same physical, time and cost limitations than other engineering subjects (Martins et al., 2013).

Edström & Kolmos (2012) present an introduction to PBL and CDIO, comparing the two approaches and concluding that "[…] CDIO and PBL can be productively combined. There is no need to make a choice between the two approaches, for an institution that plans to create an innovative engineering curriculum equipping the graduates for engineering practice, problem solving and innovation." Furthermore, the authors state that PBL can be particularly useful in the CDIO design-implement courses.

Design-implement courses in LEI

LEI is a Bologna 1st cycle program with 6 semesters, being the last semester mostly dedicated to the capstone project/internship (18 ECTS). The first 5 semesters have 16 weeks of classes: 12 weeks for traditional disciplinary courses and 4 for a design-build course (LAPR1 to LAPR5). At the end of each semester, there are 4 weeks exclusively for projects assessment and exams. The structure was adopted in 2007 and was inspired in a computing program at DTU (Denmark).

LEI is structured in two learning processes:

- Software Engineering, aiming at providing software development skills (Figure 1);
- Networks and computer systems.

The LAPR2 to LAPR5 design-build courses aim at introducing and practicing the continuous integration (CI) methodology and teamwork using an iterative and incremental approach. The technical requirements of the projects are fully aligned with the disciplinary subjects learned during the first 12 weeks of the semester. These courses are a key component of LEI, as they allow students to practice and enhance their skills in larger projects. Nevertheless, we believe that 4 weeks is too short to fully simulate an agile/iterative development approach. Also, in such a short period it is not possible to fully explore real-life conditions like evolving architecture, evolving requirements, etc. Based on this, the LEI's program management team decide to explore the possibility of the LAPR courses to become a semester-long complex projects

strongly interwind with the semester's disciplinary courses, i.e. a "mix of Aalborg style PBL and CDIO disciplinary approach" (Edström & Kolmos, 2012). This possibility is described in the remaining paper.

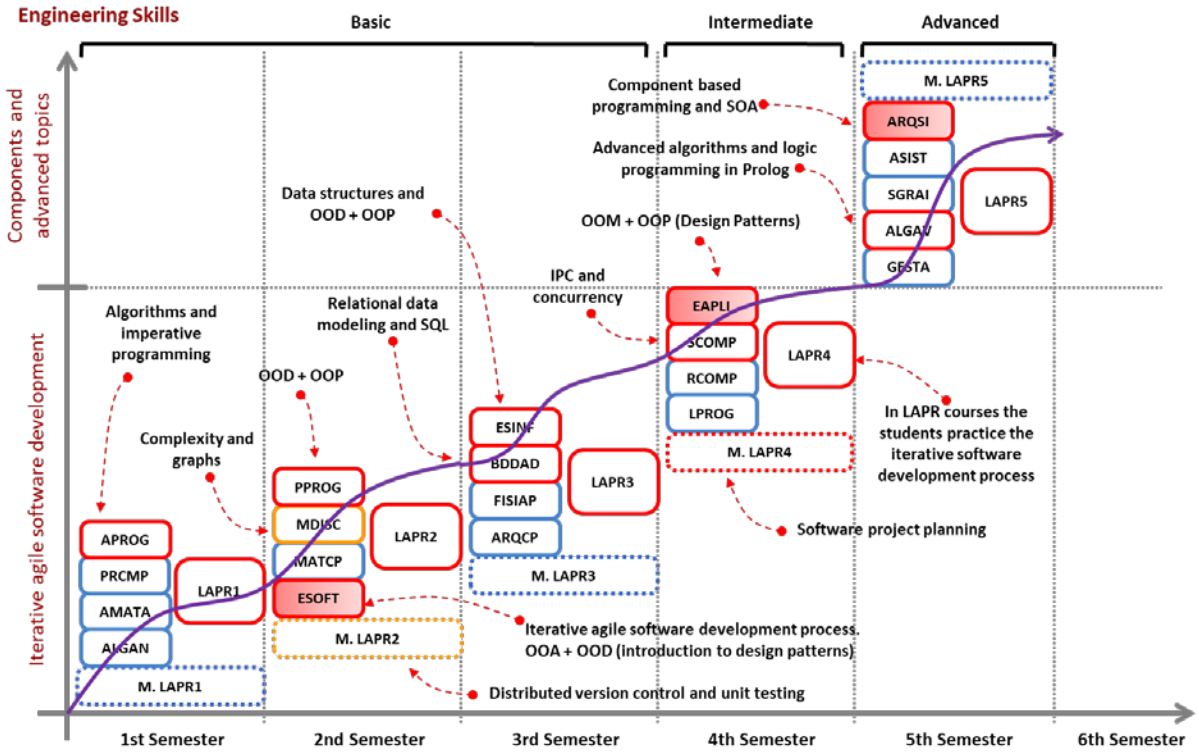

Figure 1 – Software Engineering learning process

DESIGN CONSIDERATIONS OF THE CDIO INTEGRATED LEARNING APPROACH

The introduction of the so-called CDIO Integrated Learning (CDIO-IL) approach aimed at providing students with a richer learning environment without a major structural change to the program. LEI has no elective courses, but students may choose to enroll in fewer courses than they are allowed, thus taking more time to graduate. The strong coupling between assignments from all courses of the semester makes this approach ill-suited for students that are not enrolled in all these courses. Also, there is a sizable number of students that have failed one or two courses, so that they are enrolled in one semester, but have also courses from previous semesters. The CDIO-IL was regarded as an elective track for a limited number of students and the pilot aimed at assessing if the integrated and the pure disciplinary approaches could coexist in the same program.

The disciplinary component versus PBL

LEI has 5 or 6 courses per semester, split by 4 or 5 disciplinary courses and a LAPR design-build course as depicted in Figure 1. There is pedagogical consensus in the program that resulted in the definition of common rules and pedagogical patterns that should be used by all courses. This avoids personal drifts and enforces consistency (Martins et al, 2016). School-wide pedagogical rules try to promote continuous assessment and projects, but most courses do have final exams. In some of LEI's Math and Management courses, the final exam may

have a weight of over 50%. However, on all other disciplinary courses, final exams have a weight between 30% and 50% of the course grade.

The initial idea was that the final assessment through an exam would be the same for all tracks. This is relevant for accreditation purposes, as it reduces variation within the program. As such, the learning process would have to include a relevant component focusing on outcome assessment by written exam (Bloom level 3, max). There is a paradox here: one wants to promote deep learning in CDIO-IL and, nevertheless, decide to use the same assessment tool for all students, namely a tool that, by nature, is not suitable to assess higher levels of learning. It was a dangerous compromise, as the common exam would necessarily fail one of the tracks.

The continuous assessment component of all courses in the semester would be assessed in the context of an interdisciplinary project. The project is implemented in a scrum-like approach so that the semester is divided in 2-weeks sprints. For each sprint, project requirements are given as a set of user stories, and all teams must implement and demonstrate them at the end of the sprint in a special class called “sprint review”. The division of work in the team is the whole responsibility of the team itself, i.e. they are self-managed teams as prescribed by scrum.

The sprint review is an important moment because it includes the students, all teachers and the Product Owner, i.e. the internal client of the project. At this class, the students present and demonstrate their work and are provided live feedback. Further technical feedback is given during regular classes, as well as individual and group assessment. The sprint review tries to echo what happens in the industry and is a major contribution to the students’ communication, teamwork and business skills. It is a key differentiator from the disciplinary track.

The CDIO-IL approach requires careful design and planning of the project’s user stories, as they are the semester assignments and drive all students’ work and learning. One can state that the CDIO-IL learning process is user story driven. Therefore, the user stories are the result of a collaborative effort of the teaching staff and the Product Owner. Courses and project’s requirements are taken into account and a set of user stories are selected for the next sprint, the so-called Sprint Backlog. This is a major deviation from the scrum, where the sprint backlog is defined by the development team, but without it, the project management of multiple teams would be unmanageable. Individual courses lose relevance in this interdisciplinary approach (otherwise it won’t work at all), but there are an increasing breadth and depth of the courses’ subjects and practices in the project implementation.

The general overview of the approach is presented in Figure 2. The project is formally included in the semester’s LAPR course, which also includes all application activities related to the other courses of the semester. Pure disciplinary content is dealt with the respective course. Each course’s grade results from the final assessment by written exam and the course’s application component in the project.

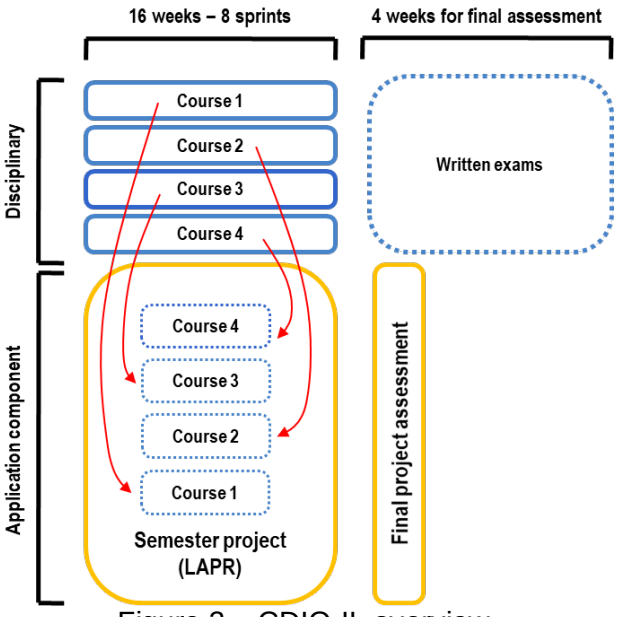

Figure 2 – CDIO-IL overview

The industry component

On one hand, developing software for academic purposes tend to be slightly different from developing professional software houses. Academic software projects are frequently started at the beginning of the semester and are "disposed" when the assessment takes place. Students barely feel the weight of producing low quality software, because they do not feel the burden and the long-run cost of maintaining it. On the other-hand, industry software houses know the long-run weight and cost of maintaining low quality software. Yet, these two realities barely meet during academic years.

For the CDIO-IL approach, in each semester, a local industry software-house is invited to host and promote a software project that could potentially become a fully commercial product. Because software-house collaborators are given access to the software project code, they inspect students code in order to advise them how to do better, according to industry standards, and students start learning that producing higher quality software is actually something that is not only perceived but pursued by software-houses.

Furthermore, students and software-houses become committed in the process. While students know they are being watched for what they produce and therefore tend to show their better skills in order to impress, software-houses also gain in doing some free content recycling and sometimes disposing of bad habits that have accumulated during the years.

In spite of the project being proposed by the software house, this is done in close collaboration with the teaching staff, which are also responsible for writing and scheduling the user stories. This is paramount as the interests of the software house may not be fully aligned with the requirements and the planning of the courses. It is important to keep the software house engaged in the project, but the learning process is the top priority. We have found that most problems can be solved with a little imagination in user story writing.

IMPLEMENTATION - THE CDIO-IL PILOT

In the 2015/2016 school year, CDIO-IL pilot class was deployed in the 3rd semester (2nd year, 1st semester) with a maximum of 32 students. It was proposed that these students should be enrolled in the pilot for 3 semesters. In the last semester, they would have the capstone project/internship, which is out of the scope of this approach. This class would share lectures with the other classes, as well as final assessment by exam on most courses, but lab classes would have to be adapted to the PBL approach: 50% would focus on the regular exercises of the disciplinary track, and the other 50% would focus on the project development.

The semester's project management was achieved by using an agile scrum-like methodology. The semester was structured in two-weeks sprints, where an element from a software house and a faculty member act as co-Product Owners (PO). The project would last 2 semesters and it was to be developed in parallel by the 4 teams, i.e. the user stories were the same for all teams. For the 3rd semester of the pilot, another project from another company would be used, as it must be aligned with the semester's requirements.

The CDIO-IL class was given an exclusive room that would be available to the class 24/24, 7 days a week. Each group would work on an island and have a 3-meter whiteboard, as depicted in Figure 3. Furthermore, we asked some companies to donate high quality puffs with their logos, in order to create a lighter and informal environment.

Figure 3 – CDIO-IL classroom

Starting with 32 students, 8 abandoned the pilot in the first week. These 8 students were good students and very competitive and they wanted to create a team between themselves. This was regarded as a danger for healthy teamwork. So, 4 teams of 6 students were created.

At the end of the first semester, 10 more students abandoned the pilot because they were not happy with their results in the exams. They liked the approach, in spite of the heavy workload and the challenges, but they felt that it didn't prepare them well for the final exams. So, in the

2nd semester of the pilot, there was only a large team of 14 students working as the development team.

In the 3rd semester of the pilot, some of the students that had left the pilot at the end of the 1st semester applied to reenter the pilot and a few more were also added to test if there was a substantial gap in the skills of students from the two tracks. The pilot was run with 3 groups of 8 students.

In the 2016/2017 school year, another CDIO-IL pilot class was deployed, this time in the 4th semester (2nd year, 2nd semester) with a maximum of 28 students and a duration of two semesters. The pilot's rules were basically the same, except for some refinements related to balancing disciplinary and PBL components in lab classes. It was clearly defined that all assessment in lab classes had to be related to user stories. Quizzes, lab tests, etc., were not allowed. This resulted from some teaching staff's resistance to the PBL approach.

Finally, in the 2017/2018 school year, another CDIO-IL pilot class was deployed, also in the 4th semester (2nd year, 2nd semester) with a maximum of 28 students and a duration of two semesters. The pilot's rules changed considerably, in an effort to align the final assessment with the PBL learning process in the project development. Therefore, final written exams were replaced by individual discussion/reflection on the courses' subjects and the project. This was a departure from the initial objective of the pilot, i.e. test if the disciplinary and CDIO-IL approach could exist in the same program. In fact, this 3rd pilot was more akin to experiment with a new independent CDIO-IL based program.

The 1st semester of the 3rd pilot run as planned and it was probably the "smoothest", in the sense that there was a very strong alignment between the courses and the project. It was possible to explore most areas of disciplinary knowledge in the project, almost every time going much deeper. The fact that one of the key courses in the semester (EAPLI) also decided to switch the final written exam for a final individual discussion/reflection in the disciplinary track may have had a positive impact in the pilot.

Unfortunately, the independent final assessment of the 3rd pilot was deemed by the school management "too different from the disciplinary track" so that it could be a risk to the program's current accreditations (national and EUR-ACE). Also, LEI's management team changed, and the new management opted to try to align the second semester of the 3rd pilot with the disciplinary track: the project would be the design-build experience of the semester (LAPR course), running the whole semester (16 weeks instead of 4), and the students would be integrated in the disciplinary track in the other courses of the semester.

RESULTS

The CDIO Integrated Learning (CDIO-IL) approach poses substantial operational and organizational challenges, but the feedback from companies who hosted these students in their capstone project/internship was extremely positive. The students of the 3rd pilot have yet to have their internships. Most of the students earned very good marks in their capstone project/internship, well above the program's average. Just a couple of them failed to reach the program's average, but more than half a dozen reached the typical top mark of 19/20 (20/20 is a rarity, about 0.2% of the students). It must be stressed that the students enrolled in the 3 pilots were not selected by their grades, but by their will to participate. Therefore, the typical

averaged grade before enrolling in the pilot was just 0.5 to 1 point in 20 above the overall average grade of all other students.

Regarding the impact of the approach on final exams' grades, one could say it was neutral. Their grades were in line with their counterparts. This led to some frustration in the students, as they felt that their hard work during the semester didn't pay off in the exams. It is not an unexpected result as exams require a specific set of skills, more often memorization and speed. The students in the disciplinary track train the whole semester the kind of small exercises that show up in exams. It is already very positive that CDIO-IL students can match their performance.

Regarding project work, most students gave their best and went much further than their colleges. They worked hard and enthusiastically, and we believe the external companies' participation was a decisive factor. The companies seem to be much more effective at motivating students than faculty and the students loved to have frequent contact with the companies. Also, companies tried to recognize the students' effort by providing summer internships, etc. For example, the team with the best project in one of the semesters was offered a trip to the retail summit in London (September 2018).

CONCLUSION

This paper presents a brief description of the application of a methodology that tries to extend the CDIO disciplinary approach with the use of PBL in a semester-long design build project course. Over three years, three pilots were deployed with some variations in the methodology, especially in the assessment.

The approach seems to have a very positive results regarding the students' software development competence and skills, teamwork and other key professional skills like knowing how to interact with a client. The technical quality of the work done and the students' maturity also has improved. There was no evidence of improvement in exams' results, which is also within expectations.

On the other hand, it resulted more difficult than expected to implement the pilots. Faculty's mindset was the biggest hurdle and it took some time to change it. The complexity and effort of creating the project user stories should not be undervalued. It requires a lot of cooperation between teachers and the company. In the end, there must be a teacher acting as co-Product Owner that is responsible for integrating all requirements and writing the user stories. This co-PO must have a very good knowledge of all courses in the semester.

One key objective of the pilots was to assess if it was possible to have two different approaches simultaneously in the same program so that the students could choose the one that suits them best. Modern program accreditation is outcome-based, so the dual-track approach is not a problem if one can assure that all students meet the required set of outcomes. Nevertheless, it is possible that auditors may find it a bit odd. It can be a risk. On the other hand, the coexistence of the two tracks was the biggest problem for teachers, some of them finding it difficult to manage two very different sets of students.

The experience of the pilots was very valuable. We are using the CDIO-IL approach in a post-graduation intensive program which aims to requalify graduates from other areas to software development. The results of the first edition were very good and we are currently in the second

edition with 51 students. A new 1st cycle completely based on the "pure" PBL version of the CDIO-IL approach is also being planned. Another quite interesting side effect of the application of this approach is that we have been asked by a large software company to help them redesign their internal training programs. This is very relevant, as in the software area there is the perception that academia is well behind industry regarding software development practices.

Finally, and not least important, this approach has contributed to the enhancement of faculty professional competence (Standard 9). This is a difficult subject in most schools adopting CDIO because professional competence enhancement is seldom aligned with career advancement. In this case, the teaching staff involved faced the same engineering challenges as the students and had to practice and improve their CDIO competences. Sometimes one doesn't introduce more engineering practice because of the potential lack of engineering skills by faculty, but probably it should be the other way around: introduce engineering practice that faculty will adapt.

REFERENCES

ACM. (2019, January 3) Curricula Recommendations. Retrieved from: https://www.acm.org/education/curricula-recommendations

Costa, Antonio, et al, (2012) "CDIO@ISEP: A Stairway to Heaven (A CDIO Contribution to EUR-ACE Certification)", Proceedings of the 8th International CDIO Conference, Queensland University of Technology, Brisbane, July 1 - 4, 2012

Edström, K., Kolmos, A. (2012). Comparing Two Approaches for Engineering Education Development: PBL And CDIO. 8th International CDIO Conference, Queensland University of Technology, Australia

Fowler, M. (2006) Continuous Integration. Retrieved (2019, January 20) from https://martinfowler.com/articles/continuousIntegration.html

Hüsing, Tobias, Korte, Werner B., Dashja, Eriona. (2015). e-Skills in Europe: Trends and Forecasts for the European ICT Professional and Digital Leadership Labour Markets (2015-2020). Empirica Gesellschaft für Kommunikationsund Technologieforschung mbH, Bonn, Germany. Retrieved (2018, December, 15) from http://eskills-lead.eu/fileadmin/LEAD/Working_Paper_-_Supply_demand_forecast_2015_a.pdf

Martins A., Costa A., Ferreira E., Rocha J. (2013). Assessing 6 Years of CDIO in a Computer Engineering Program. 9th International CDIO Conference, MIT, USA, 2013

Martins, A. et al (2016) Pedagogical Patterns as a Facilitator for Change. Proceedings of the 12th International CDIO Conference, Turku University of Applied Sciences, Turku, Finland, June 12-16, 2016

Sutherland, Jeff (2014), Scrum: The Art of Doing Twice the Work in Half the Time, Currency, ISBN-13: 978-0385346450

BIOGRAPHICAL INFORMATION

Alexandre Bragança, PhD. is an Auxiliary Professor in the Informatics Engineering Department at ISEP - Instituto Superior de Engenharia do Porto, Portugal. Between 2010 and 2018 he was in charge of the LAPR4 software design and implementation course at ISEP, where agile approaches were applied. He has more than 20 years of industry experience managing software development teams and projects.

Ângelo Martins, PhD. is an Auxiliary Professor of Computing at ISEP - Instituto Superior de Engenharia do Porto, Portugal. He has been involved in CDIO since 2008 and is deeply involved in integrated learning, especially in the area of agile software development. Between 2008 and 2018 he was the program manager of the Informatics Engineering program, which is EUR-ACE accredited. He led the creation and is the program manager of the Software Development post-graduation program (SWitCH), which aims to retool graduates with different backgrounds to software development. He is the scientific coordinator of the Information Systems and Computer Graphics unit of INESCT TEC.

Nuno Bettencourt, PhD. Is an Auxiliary Professor in the Informatics Engineering Department at ISEP - Instituto Superior de Engenharia do Porto, Portugal. He actively promotes and participates in initiatives adopting an integrated learning approach, especially in the area of software development. Between 20014 and 2018 he was a member of the Informatics Engineering 1st cycle program management team.

Paulo Maio, PhD. is an Auxiliary Professor in the Informatics Engineering Department at ISEP - Instituto Superior de Engenharia do Porto, Portugal. He actively promotes and participates in initiatives adopting an integrated learning approach, especially in the area of software development. He has been also involved in joint I&D projects between academia and software industry, namely regarding software re-engineering activities. Since 2015, he is assistant-manager of the Master in Informatics Engineering program, which is both EUR-ACE and ABET accredited.

Corresponding author

Ângelo Martins
Instituto Superior de Engenharia do Porto
Rua Dr António Bernardino de Almeida, 431
4200-072 Porto, Portugal.
+351 917334270
amm@isep.ipp.pt
angelo.martins@inesctec.pt
Skype name: angelomartins