Holistic Analysis of the Effectiveness of a Software Engineering Teaching Approach

José Carlos Metrôlho, Fernando Reinaldo Ribeiro
R&D Unit in Digital Services, Applications and Content
Polytechnic Institute of Castelo Branco
Castelo Branco, Portugal
metrolho@ipcb.pt, fribeiro@ipcb.pt

***Abstract*—To provide the best training in software engineering, several approaches and strategies are carried out. Some of them are more theoretical, learned through books and manuals, while others have a practical focus and often done in collaboration with companies. In this paper, we share an approach based on a balanced mix to foster the assimilation of knowledge, the approximation with what is done in software companies and student motivation. Two questionnaires were also carried out, one involving students, who had successfully completed the subject in past academic years (some had already graduated, and others are still students), and other questionnaire involving companies, in the field of software development, that employ students from our school. The analysis of the perspectives of the different stakeholders allows an overall and holistic) view, and a general understanding, of the effectiveness of the software engineering teaching approach. We analyse the results of the questionnaires and share some of the experiences and lessons learned.**

Keywords- agile methodologies; education; software engineering; teaching; teamwork.

I. INTRODUCTION

One of the biggest challenges in teaching software engineering is empowering students with the knowledge and skills they need to be well prepared to face the labour market. This includes providing students with technical skills but also providing them with the non-technical skills associated to the software engineering process. It is also known that the teaching of software engineering cannot be limited to the presentation of concepts and methodologies as a set of abstract concepts. In our previous paper, presented in ICSEA [1], we analysed how the main concepts of the software engineering subject are assimilated by the students and if they are applied in the labour market. We also learn that it is important for students to develop practical projects to complement their education. This is corroborated by other authors who point out that wherever possible, software engineering teaching should be adequately complemented with the practice of software engineering projects so that the students can assimilate and understand them successfully [2]–[4]. Additionally, it is important to consider the growing importance of human factors in the software development process [5] and consequently the role that some of them play in the software engineering process, namely: communication, coordination, collaboration, trust, expert recommendation, program comprehension, knowledge management and culture.

Several approaches and strategies have been proposed and used to improve the teaching and learning of software engineering. They all ensure the importance of giving students hands-on experience. However, the way they propose to do so differs greatly.

This paper describes an experience in teaching Software Engineering, of a Computer Engineering program, using a project-based approach. This approach is enriched with the collaboration of software houses giving the students a real-world experience of software engineering project development. In our paper [1] we tried to understand how the main concepts of the subject are assimilated by the students and if they are applied in the professional life of our past students. In this paper we extend our approach to include the point of view of the companies that employ our students. The opinion of these companies, which employ and develop activities in this area (Software development), is very valuable. It may represent significant contributions for the improvement of the teaching-learning process and for better integration of the students in the labour market. To reach this goal, we conducted a questionnaire with a group of employers. We chose 5 companies that have recruited our graduates in Portugal. This set of companies is obviously reduced compared to the universe of thousands of software development companies worldwide. However, the collected data is a starting point for analysing what is considered relevant to these partners. Although we graduate students for a larger universe of companies, the collected data are useful as indicators of aspects that we must take into account, while obviously not neglecting other methodologies and topics related to Software Engineering.

The results of this questionnaire allow us to improve the definition of the topics on which the lecture should focus and to keep the syllabus updated. Additionally, it helps us to understand the strengths and weaknesses of the employees who graduated from our school (soft skills and technical skills). This analysis of the perspectives of the different stakeholders allows an overall and holistic view and a

general understanding of the effectiveness of the software engineering teaching approach.

The remainder of this paper will be as follows: Section II presents a brief review of related work; in Section III we present an overview of our project-based approach for software engineering; Section IV provides a brief description of the questionnaires that were conducted to achieve feedback from former students and employers; in Section V we present the results and analysis of the questionnaire; Section VI presents some lessons learned and challenges faced and finally, in Section VII we present some conclusions and we outline some of the future work.

II. Related Work

To provide the best training in software engineering, several approaches and strategies have been proposed. Some of them are more theoretical, more focused on the study of theory through books and manuals, while others have a more practical focus and often done in collaboration with companies. Nowadays, it seems to be a well-accepted fact that the software engineering training should not be strictly focused on the theoretical study of concepts and methodologies. It is important to provide students with hands-on experience in a software engineering project and provide them with the non-technical skills in a software project. It is important to promote hands-on ability training and the rapprochement between teaching and practice. Additionally, the recent diffusion of agile methodologies in software development brings many difficulties and challenges to software engineering teaching. In this context, several authors refer that current approaches to teaching software engineering are outdated and lack authenticity [6], [7]. However, as referred in [6], it is not clear which should be the best approach and there are different perspectives with different proposed approaches. Some authors (e.g., Clear and Damian [6][8]) suggest that the best approach is to emulate the workplace through distributed software development projects, through cross-university or cross-course courses, others (e.g., [9]–[11]) suggest involving students in a project where they have the possibility to experience team work and understanding in the practice of the theoretical concepts dealt with in the course and others (e.g., [12]–[14]) argue for the use of simulations and games to provide students with a variety of experiences that would not be possible within the constraints of an academic environment. Next, a brief analysis of some works that have been proposed for each one of the perspectives identified before is presented.

The emulation of the workplace through distributed projects or cross-university courses was approached and experienced by some authors. The DOSE [8], a Distributed and Outsourced Software Engineering course, followed an approach to teaching distributed software engineering centred in a distributed software development project. They experienced teaching software engineering using a geographically distributed software project involving various countries with different cultures, native languages and time zones. This approach gives the students the opportunity of facing the challenges of distributed software development and helps them understand typical software engineering issues, such as the importance of software requirements for specifications, or the relevance of adequate system design. However, they also identify some time scheduling inconveniences, and difficulties in keeping teams committed to their peers. The Undergraduate Capstone Open Source Projects (UCOSP) program [15] ran for ten terms over six years providing for over 400 Canadian students from more than 30 schools. After this period, the authors identified some lessons they had learned: Students work on real distributed open-source projects as full members of software development teams; they use the same software development processes as regular team members and are provided with explicit mentorship from volunteer mentors from each project; students integrate and apply the skills they have learned in their courses in a real development setting; students develop and improve their technical communication skills in a real development setting.

A project-oriented approach is followed in several software engineering training programmes. Its purpose is to teach students the theoretical and the practical aspects of developing software systems in a team environment giving students a chance to experience a work scenario that is closer to a real-world experience. A Project-Based learning in software engineering Lab, teaching through an e-Portfolio approach is described in [10]. In this approach, the e-Portfolio allows students to carry out a software project, addressing each phase collaboratively with other students and obtaining appropriate feedback from instructors. The e-Portfolio includes a single problem statement for the development of a complete software project comprising of a set of deliverables. To support the implementation, they chose the Moodle Platform. To assess the students' e-portfolios, various rubrics were implemented by scoring and weighting the sections and categories for every deliverable to be evaluated. Another project-based learning approach for teaching software engineering concepts is described in [11]. Their goal is to teach software engineering concepts using the Scrum framework in real life projects. Projects usually have a capacity of about 1000 workhours. To make the projects more relevant real customers were incorporated. They bring in requirements from industry and present their topics during a kick-off meeting. During the project, students work together as self- organized teams (5-7 elements). They chose an appropriate project management and team coordination process and they are only asked to use some core tools that are needed to monitor the projects.

A game-based learning methodology of teaching software engineering is presented in [13]. They suggest a methodology of two-fold use of learning games for teaching software engineers. Students, experienced in programming, develop learning games, and then they use the games that are developed for teaching the next generation of students. Students developing games learn the software development life cycle phases including testing, deployment and maintenance, they contact with customers (teachers of corresponding subjects act as customers) and users (students, learning these subjects). In their approach, they find both advantages and disadvantages. As advantages, they identify the increasing students' motivation and revealing their

creativity. The main problems observed include difficulty in organizing team work especially for students of early years and lack of time for coordinating them. Schäfer [14] describes some lessons learned after two teaching periods in using Scrum with gamification to learn and train the agile principles. They found that their approach has both advantages and disadvantages. Gamification is motivating and helps to bring participants with different backgrounds together in project teams. As drawbacks, they refer to the importance of having a real external stakeholder or customer defining a project goal externally in a Scrum learning project.

There are different approaches and strategies that may be followed to provide students with the best training in software engineering. All of them agree that the theoretical study of concepts and methodologies should be complemented with hands-on experience in a software engineering project. This would allow students to be provided with a better understanding of the theoretical concepts and to equip them with the non-technical skills in software projects. However, the way different approaches propose to provide the students with the practical experience is very different. Some of them suggest emulating the workplace through distributed projects, which may involve several entities and thus provide interesting experiences in software engineering. Others suggest a project-oriented course where students can practice requirements analysis, project management, development methodologies and teamwork. Another recommendation is using simulations and games to simulate distinct scenarios in software engineering teaching and training.

However, regardless of the approach or strategy, it is necessary to understand whether students have acquired the knowledge and skills they need for the performance of their duties, and whether they apply them in their professional activity in software engineering. To understand this, it is a holistic analysis (i.e., analysis of the big picture involving several stakeholders, namely students, alumni and employers, about the teaching methodology implemented in this subject in recent years) that is important since it allows for an understanding of the vision of the different stakeholders that may be involved in the software engineering teaching process: teachers, students, graduates and employers.

III. Overview of our Approach for Software Engineering

In this case a project-based approach was adopted for teaching Software Engineering. This subject is part of the second year of a computer science course (undergraduate course). It is a subject that has 5 ECTS and where the semester load is 30 hours for theoretical classes and 45 hours for laboratory classes. The focus of the adopted approach was to combine theory and practice.

One teacher is responsible for the subject management and theoretical lectures. In these classes, the teacher presents the concepts and methodologies and promotes discussion about them. Students are also provided with an introduction to some software development methodologies namely Waterfall, Extreme Programming, Scrum, Spiral, etc. Other topics analysed include quality and metrics in software engineering, software design, implementation, testing, configuration management, among others. In the assessment, this theoretical part has a weight of 40% for the final grade; the remaining 60% is from the practical component.

Another teacher is responsible for the practical classes. In these classes, students acquire some practice of software engineering through the specification, design, implementation and validation of a software application, as a project for teams of 4-6 students. Scrum is the adopted agile software development methodology. The teacher acts as a product owner. Each team member has a specific function (e.g., Scrum Master, Designer, etc.). Each team develops a different project. However, all the projects are focused on the development of a game from a software engineering perspective. This is important to maintain the students motivated and engaged with the project. The first deliverable is revised to accommodate feedback from the product owner. Trello is used for project management and to track progress on tasks.

A. Additional Realism

One class of the subject has been taught by professionals from software house companies. In this class, software development processes like Feature Driven Development (FDD) and Behaviour Driven Development (BDD) were approached and some of their practical aspects are discussed.

Another important initiative to enable students to get in touch with practice in software engineering is a one-day visit to the premises of another software house company. This company (Outsystems) is well-known for the software development platform they hold and that is used by many software companies worldwide. Their platform is a low-code platform for rapid application development. It is especially designed for developing applications in the context of agile projects. During this journey, students were able to have closer contact with some Scrum activities (namely Daily Scrum, Sprint, Sprint Execution) and contact with some Scrum Roles (Scrum Master, Development Team). Professionals explain to the students what they are doing, and which technologies and tools are used to support their activities. Students also had a brief session about software cost estimation.

These events are very important since they provide students with the contact and interaction with real software engineering projects with real stakeholders. They help to improve the understanding and the assimilation of the concepts learned in the classes of the subject.

B. Student evaluation

The student evaluation comprises both theoretical and practical evaluation. The theoretical evaluation is a written

exam over the course material. The exam consists of 10 questions chosen from the list of 30 questions that were made available to the students at the beginning of the semester. Most questions are reflexive questions about software engineering subjects. With this approach, the intent is to avoid students wanting to memorise the matters learned along the semester (15 weeks). Also, it is desirable that students learn and acquire knowledge for a long-life period, mainly to be used after graduation on their job integration experience. In section V, some gathering data that wants to evaluate results about the achievement to this goal of our approach will be presented.

For the practical evaluation, throughout the semester, during the 15 working weeks, students´ working teams develop the product on 6 sprints (sprints here are defined as having 2 weeks each). The teacher (i.e., product owner) meets with each team at the end of the sprint to evaluate the work in progress, the achievements and the goals for the next sprint. The team works in class (3h/week) and out of class. Halfway through the semester, after sprint 4, and at the end of the semester, after sprint 7, each team has an assessment session where both teachers are present to evaluate different parameters. Some of the parameters are: clear goals, state of the art, requirements (functional and non-functional), software development process (roles, artefacts, timings, hits and misses), team member´s description (roles, skills) task scheduling (monitoring using Trello tool), modelling (user stories), implementation (code), budget (estimated based on the lesson learned during the visit to the company referred to on the previous section of this paper), conclusions (pros and cons) and future work, literature used and citation on the final report, and final presentation and discussion.

One of the achievements that students sometimes realize is learning from mistakes. For instance, if they do not communicate within the team the achieved results are poor, when compared with other more cohesive teams. On the other hand, in collaboration with the "Scrum Master" of the team, a deeper evaluation can be done to eventually assign different grades to the members of the team.

IV. UNDERSTANDING STAKEHOLDERS' PERSPECTIVE

In order to gauge the post-retention cognitive load, a questionnaire of former students was conducted in order to obtain feedback on the importance of the subject to their current professional activity (of those who finished the course and work in the area), and also to know if the knowledge transmitted in the theoretical classes remains. For this last component, the questionnaire included questions that had already been used in the theoretical evaluation of the subject. The answers were evaluated with the same evaluation criteria, graded in a scale of 0-20. The questions were selected from the same set of 30 questions referred to in Section III-B. Respondents were informed that the results were for a study. They were also informed that the goal of the study was to understand if the concepts and knowledge acquired in the Software Engineering subject remained present. The questionnaire was also used to gather insights about the usefulness of the subject for each graduate's the practical life. Thus, questions about aspects that may be used in the day to day of their professional activities in the companies where they currently work, were included in the questionnaire.

Also, in order to get feedback from employers about issues that are important to graduate students starting their professional activity, a questionnaire for employers was conducted. The questionnaire included questions about the development processes used in the company (traditional, agile, etc.); the importance of software engineering contents to the company's activity; soft skills and technical knowledge that have more importance to the company; and a question about topics or issues that, in their opinion, should be considered in software engineering subjects. The answers were analysed and will be presented in further sections. Respondents were informed that the results were for a study. The questionnaire was also used to gather insights about issues that must be included in future editions of this subject. Thus, questions about aspects that may be important in the day-to-day activities in the companies were included in the questionnaire.

A. Questionnaire of Former Students' Description

This questionnaire was designed to be directed towards our objectives and be filled in quickly and simply. Some questions were answered in free text (case of questions of theoretical knowledge) and others are multiple choice questions (e.g., used software methodologies). The questionnaire was organized in three parts: Questions about the current professional activity of the respondents; theoretical questions about software engineering; and space for feedback on the importance of topics in their current professional life (for those who had already finished the course).

As examples of questions, we asked if the graduated students were working. If yes, we asked about the actual tasks in their companies (Planning, Requirements analysis, Design, Code, Quality control, Tester, Project management, other), the used methodologies (Waterfall, Scrum, XP, Prototyping, Spiral, FDD, Lean, RUP, other, none). About the theoretical questions we asked about the fundamentals of Software Engineering, Software Quality, Verifications vs Validation, traditional vs Agile, team dimensions and roles, among other questions and feedback.

B. Questionnaire of Employers' Description

This questionnaire was also designed to be directed towards our objectives and be filled in quickly and simply. Some questions were answered in free text and others are multiple choice questions (e.g., used software methodologies). The questionnaire was organized in distinct parts: Questions to characterize the company activity; questions to characterize topics of importance to the

companies' activity and feedback with contributions for future improvements of the syllabus.

As examples of questions, we asked about the respondent's experience, position in the company, number of students graduated from our school that work/worked in the company, activity of the company (planning, requirements analysis, design, code, quality control, tests, project management, quality assurance, others), used software processes in the company (Waterfall, Scrum, XP, Prototyping, etc.), from the different company's activities what are the most important. We also asked about the soft skills and technical knowledge that are most important to the company activities. And lastly, but not least important, we requested feedback to improve and keep the syllabus updated.

V. Questionnaire Results and Analysis

This section presents the results of the questionnaires answered by the students, graduates and also the results of the questionnaire answered by the employers.

A. *Data Collection/Methodology: students and graduated students*

As a universe of respondents, questionnaires were sent to 97 students. Of these, 56 were undergraduate students (although they had passed in this subject) and 41 graduated.

The questionnaire was done online, using the *LimeSurvey* Webtool.

The response rate was of 24.4% of the graduated students and of 21.4% of the undergraduate students.

It is important to note also that some respondents did not answer all questions.

B. *Results and Analysis: students and graduated students*

Figure 1 shows the activities the respondents (Graduated students) are involved in, in their work. 84% of the respondents are involved in more than one activity. 50% of them are involved in planning, analysis and testing but they are not involved in implementation.

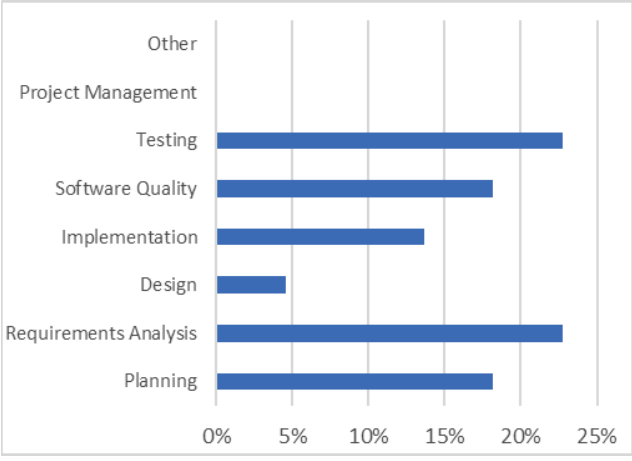

Figure 1. Activities carried out (Graduated students).

Graduated students were also asked to identify the software development methodologies they use in their activities. They were able to identify the methodologies they use considering a list of given methodologies. Results are presented in Figure 2.

More than 70% of the respondents refer that they use the Scrum methodology. This appears to be in line with the results presented in the "12th annual State of Agile report" [16] that refers that 52% of respondents stated that more than half of the teams in their organizations are using agile practices. And it is also in accordance with the results presented in another questionnaire of more than 2,000 active Scrum and Agile practitioners [17]. This study refers that 94% of agile users use the Scrum approach in their agile practice (78% use Scrum with other approaches).

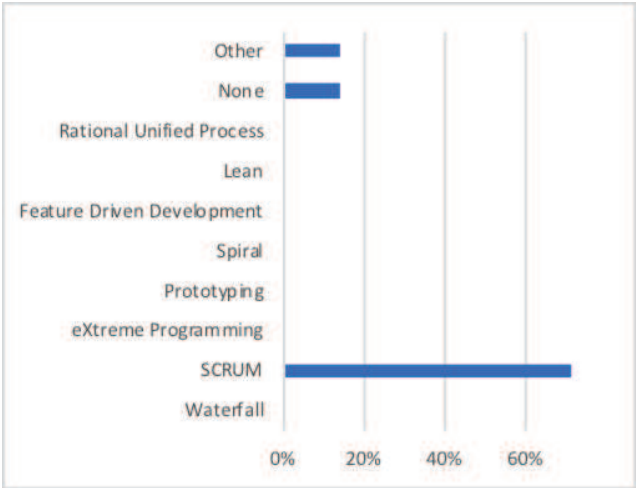

Figure 2. Software development methodologies (Graduated students).

With respect to the importance of the subjects learned, 87.5 percent, of the 8 graduated students that respond to this question, said that the content learned in the course has been considerably useful for their actual professional activity (see Figure 3).

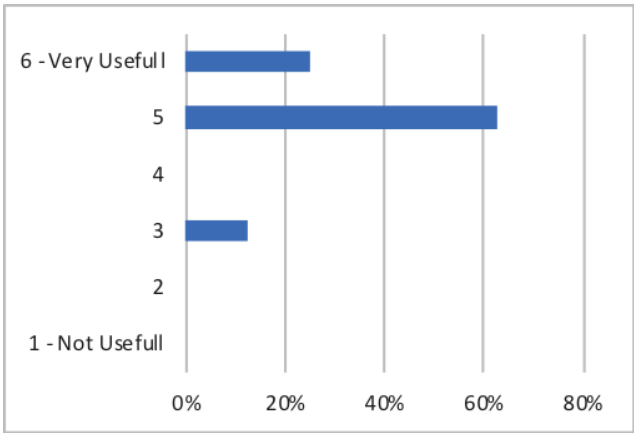

Figure 3. Course content vs professional activity (Graduated students).

The second part of the questionnaire was related to theoretical questions about software engineering. This part was evaluated in a 0-20 scale and we compare these results with the results achieved by the same individual during the course. We consider the individual "maintained" if *(grade achieved in the subject -1.5 ≤ grade achieved in the questionnaire ≤ (grade achieved in the subject +1.5).*

After evaluating the answers of the non-graduated students to the questions, we conclude that there is a majority (58%) that has maintained or increased the result (41% maintained, 17% increased) (see Figure 4).

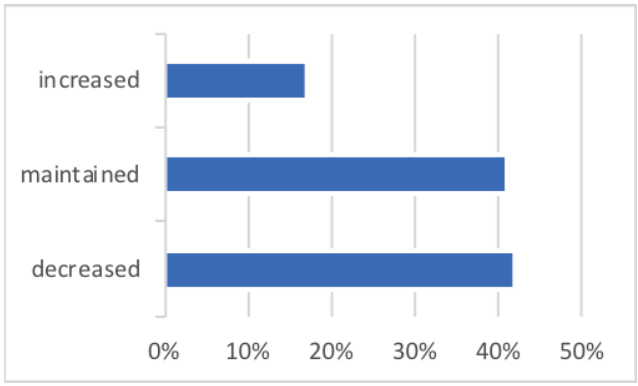

Figure 4. Grades evolution (Students).

In the case of students already graduated, the results, presented in Figure 5, are better (less cases (37.5%) of lowering grades). Despite the long period of time after they attend the course, this is probably a consequence of the practical experience they get in the field of software development.

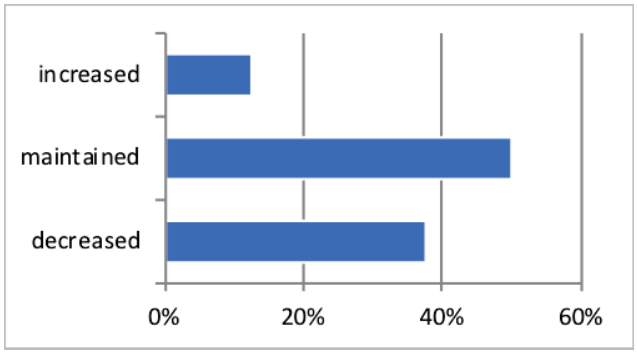

Figure 5. Grades evolution (Graduated students).

In addition, at the end of each semester, a questionnaire is usually conducted in order to obtain knowledge about the students' perception of the importance of the subject for their academic education. This questionnaire addressed four issues: acquisition of knowledge; development of skills, improvement of critical thinking; relevance for academic background. Figure 6 shows the results of the questionnaires (average values) performed in the last 2 years. Each topic was evaluated on a 6-level scale (1 nothing important – 6 very important).

In general, all issues were evaluated very positively, which shows that there is a recognition of the importance of the subject for their education.

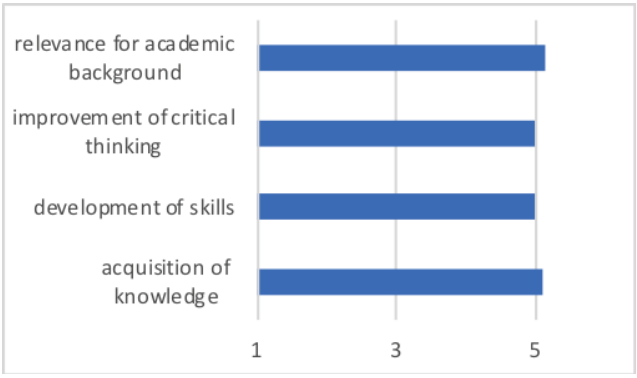

Figure 6. Students' perception of the importance of the subject for their academic education.

C. *Data Collection/Methodology: employers*

We selected 5 companies that have employed graduates of the school in recent years. The selected companies are multinationals, working in the ICT area, and each of them has at least graduates from the school as collaborators. For each company we asked a member, with an intermediate or high responsible position, to fill in a questionnaire about their activities and about the school graduates they currently employ. The questionnaires were completed by representatives of the company that hold leadership positions (Senior Manager, Executive Director, Team Manager, Business & Project Manager, Ecosystem Talent Director). The professional experience, in the ICT area, of these representatives of companies, goes from 13 years to 21 years.

The questionnaire was composed of 4 parts. The first part with general questions about the company and the respondent. The second part with 3 questions about the activity of the company. Two questions on which the respondent had to select from among the various options available, and one question where the respondent had to evaluate various options on a 1-6 scale (1 nothing important – 6 very important) on the importance of software engineering to the company's activity. The third part of the questionnaire consisted of 2 questions about the technical and soft skills of the employees who were graduated by the school. In these questions, the respondent had to evaluate several options on a 1-6 scale (1 very poor – 6 very good). Part 4 includes only one question where respondents were asked to provide feedback or additional input.

The five representatives of the companies replied to the questionnaire.

D. Results and Analysis: employers

Respondents answered that they had already worked, or are currently working, with 7, 15, 20, 30 and 38 employees who graduated from the school (each value correspond to a different company). This number of employees, graduated from the school, is higher than the number of graduates who were questioned (Sections V.A and V.B) because they represent graduates of several years. Although they do not represent the same universe, some of the graduates questioned in Sections V.A and V.B are employees in these companies. Therefore, they may be included in the group referred to herein.

Respondents were asked to fit their area of intervention by considering a list of 8 activities related to software engineering (see Figure 7). Each respondent could select several activities from a list or indicate other activities. Most companies focus on several areas of software engineering. Only areas related to quality assurance and software quality control are not ensured in all questioned companies.

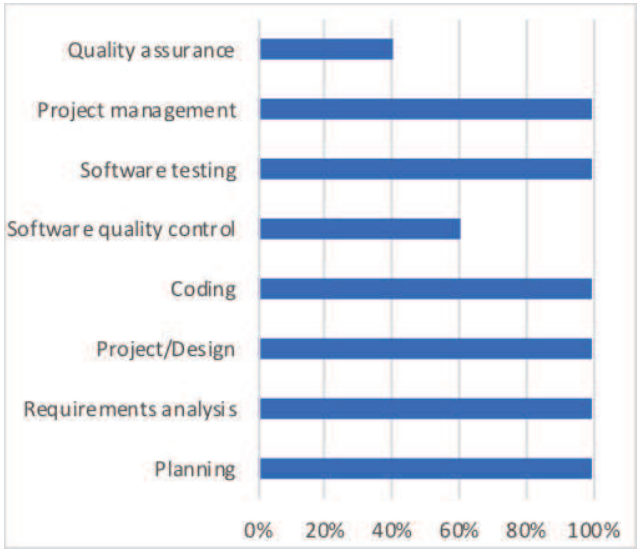

Figure 7. Company's activities.

Another important aspect, regarding the activity of companies, is related to the software development methodologies they use most in their activity. In our classes we teach some software development methodologies namely Waterfall, Extreme Programming, Scrum, Spiral, etc. However, in the practical project, Scrum is the adopted software development methodology. The goal is to provide students with knowledge and practice about the methodologies most companies follow. And it seems to be a wise decision. According to the questionnaire's responses, the Scrum methodology is the one most used by these companies. The waterfall software development methodology is also one of the most used. Figure 8 shows the software development methodologies used by the respondent companies. In this question each respondent could select, from a list, all the methodologies that they used in their projects. They could also add other methodologies.

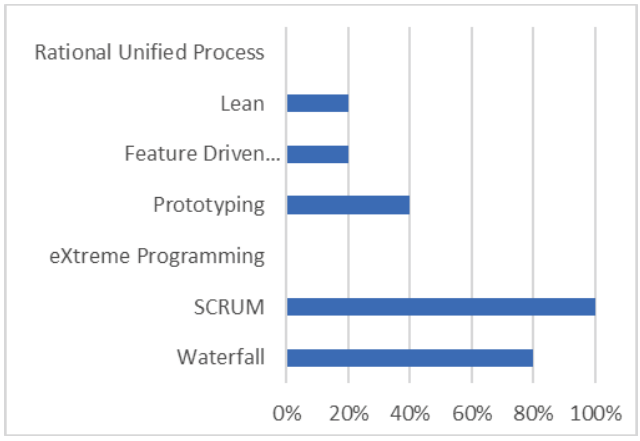

Figure 8. Software development methodologies used in the company.

Respondents were also asked to evaluate the importance of technical knowledge and skills, in 9 areas (from a predefined list) associated with software engineering. The graph, shown in Figure 9, presents the number of respondents that evaluated each area of knowledge and the average value of the importance that these respondents gave to this area of knowledge. It is important to note that some of them did not evaluate all available areas. The knowledge areas that were evaluated as most important were (in descending order of importance): coding, requirements analysis, development methodologies. Some respondents also mentioned user experience, debug and problem solving.

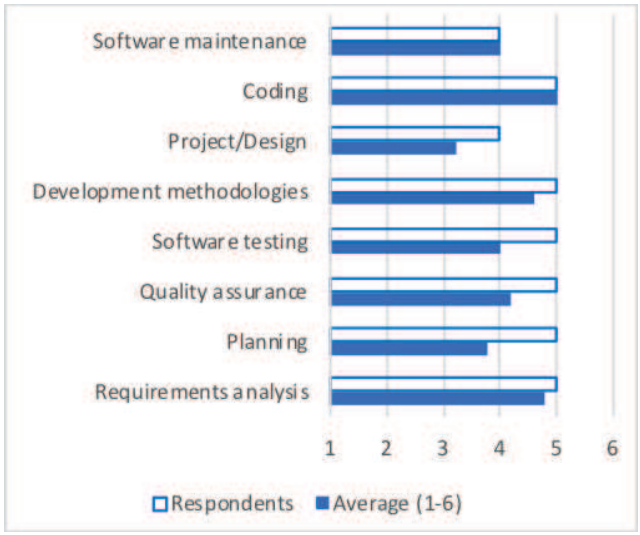

Figure 9. Importance of knowledge learned about Software Engineering for what the activity of the company is.

The next two figures represent the opinion of the representatives of the companies on graduates at school. Similarly, to the previous figure, in these issues some companies did not evaluate all the issues that are available in

the list. Graphs, shown in Figure 10 and Figure 11, show the number of companies that evaluate each skill and the average value of the evaluation that these companies gave to that skill. It is also important to note that companies were asked to make an overall assessment of employees for each competency. However, this does not have an easy answer. Employees have different competencies, work on different projects, and often relate to the respondent in different domains. This was corroborated by the respondents and, in particular, one which stated that "…it is very difficult to assign a general classification to all the employees who were recruited from school courses. Besides being many, they were also in different periods, different courses and as you know, not all are the same."

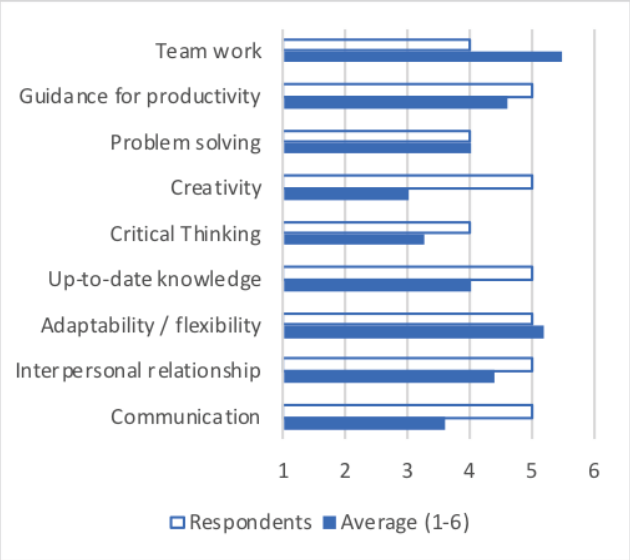

Figure 10. Strengths and weaknesses of the employees who graduated from our school (Soft skills).

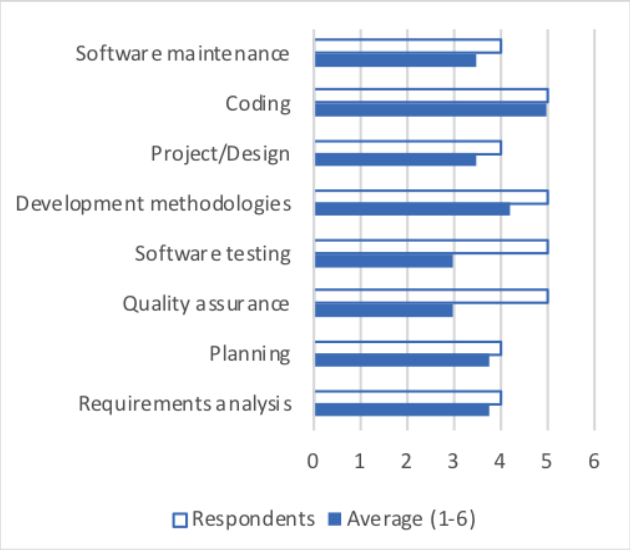

Figure 11. Strengths and weaknesses of the employees who graduated from our school (knowledge and technical skills).

Figure 10 corresponds to the feedback given by the respondents about the soft skills of their employees who graduated from school. The two skills that got higher ratings were related to teamwork and adaptability/flexibility. Both obtained very positive ratings from all respondents. With less positive evaluations arise the creativity and critical thinking.

The graph (from Figure 11) presents the feedback given by the respondents about the technical skills of their employees who graduated from school. Coding stands out for the positive. Software testing and the quality assurance had lower ratings.

The last part of the questionnaire was one open question where respondents were free to provide feedback or additional input. The answers obtained are different because they represent different perspectives and are usually conditioned by the nature of the company and the activities in which it focuses. However, these are valuable inputs as they represent the perspectives and real needs of companies. Below, some comments received:

"... any educational institution should be increasingly adapted to the major market trends, including them in the course programme ... so that the transition to the labour market is simple and contributes to meet the current needs ...".

"… strengthen the most used subjects: Methodologies/Development processes, Implementation/Coding, Maintenance and Software Testing...".

"…strengthen the areas of Software Quality (automatic software testing, AI (artificial intelligence), ... as well as Computer Security…".

However, these comments also show the different perspectives and focus of each of the companies.

VI. Lessons Learned and Challenges Faced

The contributions of this paper are in the form of the lessons learnt, which may be seen as guidance for others looking to approximate the know-how of students to the methods and techniques used by software companies. In summary, these are:

- Students should learn by doing and, wherever possible, software engineering principles should be assessed in the context of practical work, rather than by regurgitating material taught or extracted from textbooks.
- Students must have well defined and known goals. The assessment of the theoretical subjects does not need to be a surprise in the exam.
- Opening classes to external stakeholders (by promoting talks or visiting companies) during the last part of the semester helps students to reinforce knowledge and motivate them to the subjects.
- It is very important to get feedback from past students and evaluate if the transmitted concepts and knowledge are still there, and if it was improved by the work experience in the labour market.
- It is important to choose projects that are of interest to the students and that can motivate them and involve them in their development.

- It is very important to get feedback from companies that employ past students. It may contribute to a better adjustment of the syllabus with the real needs of the labour market. It also provides very important feedback on the technical and behavioural skills of former students.
- It is important to analyse the different perspectives of the various stakeholders: teachers, students, graduates and employers. This allows a holistic analysis and may help to improve the teaching methodologies.

However, during our experience, we faced challenges like:

- Difficulty in maintaining all team members equally motivated and engaged in the same way throughout the entire project development period;
- Keeping all students involved in the project. Some students may drop out, leaving the team during the semester, and affecting the workflow and scheduling of the remaining members of the team;
- Allowing students to experience various roles within the team. It is necessary to find a way to rotate the roles of each one within the team, to avoid any student being too focused on just one role. It is important that everyone experiences a diversity, as broad as possible, of different roles;
- Allowing students to experience different methodologies in real environments. More field trips and contact with companies that use different methodologies, must be promoted to foster more diversity of experiences.
- The representative of the companies that collaborated with us were very cooperative. However, some difficulties in responding to questionnaire questions were identified. This was mainly due to the fact that companies employ several graduates and therefore they try to make an average assessment.

VII. Conclusion and Future Work

Our questionnaire of former students was the starting point of a reflexion about the impact of the approach followed in previous years in the subject of Software Engineering. Based on the results, we think that allowing students to know the pool of questions in advance, fosters the students on important knowledge in the field and to understand these items, that we want students to maintain over a long period of time. The second questionnaire, of employers, give us important feedback to know the most important aspects of software engineering to companies, in the field of software production, where several of our former students are working. The feedback allows us to understand the employer's opinion about our graduated student's training and get contributions to focus our teaching goals in topics that are considered relevant to the future of our students. Using the feedback achieved from the questionnaire given to the employees of our graduated students, we want to improve and maintain updated the contents of this subject. This holistic analysis that includes different perspectives from different stakeholders: teachers, students, graduates and employers, gave us important guidelines to improve the teaching methodologies and syllabus.

Regarding the assessment of students, in future editions of the subject the pool of questions will be increased to improve the effect of randomisation for the next exams. Also, a mix of questions from the pool (~66%) and other questions (~33%), will be used to build the exams and explore the advantages of both approaches. As for the practical component, based on the results, Scrum is still used as a case study since it is one of the most used processes by companies where our graduated students work.

One final remark to reiterate that the study presented here is based on data collected from our students and alumni of the Software Engineering subject and from a group of 5 experienced representatives of multinational companies with whom we interact. This set of companies is obviously reduced compared to the universe of thousands of software development companies worldwide and the opinion of other employers may differ significantly according to their own reality and activity. In any case, the collected data is useful as indicators of aspects that we must take into account, obviously without neglecting other methodologies and topics related to Software Engineering.

We will continue to make all efforts to listen to these types of stakeholders (students, alumni and representatives of companies) and to broaden the universe of respondents, with the aim of keeping the themes and methodologies taught updated.

References

[1] J. Metrôlho and F. Ribeiro, 'Software Engineering Education: Sharing an approach, experiences, survey and lessons learned', in *The Thirteenth International Conference on Software Engineering Advances (ICSEA)*, 2018.

[2] R. Chatley and T. Field, 'Lean Learning: Applying Lean Techniques to Improve Software Engineering Education', in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*, 2017, pp. 117–126.

[3] S. D. Zorzo, L. de Ponte, and D. Lucrédio, 'Using scrum to teach software engineering: A case study', in *2013 IEEE Frontiers in Education Conference (FIE)*, 2013, pp. 455–461.

[4] M. Kuhrmann and J. Münch, 'Enhancing Software Engineering Education Through Experimentation: An Experience Report'. 2018.

[5] C. Amrit, M. Daneva, and D. Damian, 'Human factors in software development: On its underlying theories and the value of learning from related disciplines; A Guest Editorial Introduction to the Special Issue', *Inf. Softw. Technol.*, vol. 56, no. 12, pp. 1537–1542, 2014.

[6] S. Beecham, T. Clear, D. Damian, J. Barr, J. Noll, and W. Scacchi, 'How Best to Teach Global Software Engineering? Educators Are Divided', *IEEE Softw.*, vol. 34, no. 1, pp. 16–19, 2017.

[7] F. Matthes *et al.*, 'Teaching Global Software Engineering and International Project Management - Experiences and Lessons Learned from Four Academic Projects', *3rd Int. Conf. Comput. Support. Educ. CSEDU 2011*, p. 12, 2011.

[8] M. Nordio *et al.*, 'Teaching Software Engineering Using Globally Distributed Projects: The DOSE Course', in *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, 2011, pp. 36–40.

[9] D. Dahiya, 'Teaching Software Engineering: A Practical Approach', *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 2, pp. 1–5, 2010.

[10] J. A. Macias, 'Enhancing Project-Based Learning in Software Engineering Lab Teaching Through an E-Portfolio Approach', *IEEE Trans. Educ.*, vol. 55, no. 4, pp. 502–507, 2012.

[11] A. Heberle, R. Neumann, I. Stengel, and S. Regier, 'Teaching agile principles and software engineering concepts through real-life projects', in *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2018, pp. 1723–1728.

[12] M. Yampolsky and W. Scacchi, 'Learning Game Design and Software Engineering Through a Game Prototyping Experience: A Case Study', in *Proceedings of the 5th International Workshop on Games and Software Engineering*, 2016, pp. 15–21.

[13] O. Shabalina, N. Sadovnikova, and A. Kravets, 'Methodology of teaching software engineering: Game-based learning cycle', *Proc. - 2013 IEEE 3rd East. Eur. Reg. Conf. Eng. Comput. Based Syst. ECBS-EERC 2013*, pp. 113–119, 2013.

[14] U. Schäfer, 'Training scrum with gamification: Lessons learned after two teaching periods', in *2017 IEEE Global Engineering Education Conference (EDUCON)*, 2017, pp. 754–761.

[15] R. Holmes, M. Craig, K. Reid, and E. Stroulia, 'Lessons Learned Managing Distributed Software Engineering Courses', in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 321–324.

[16] VersionOne Inc, '12th annual State of Agile report', 2018.

[17] Scrum Alliance, 'STATE OF SCRUM 2017-2018. Scaling and agile transformation.', 2017.