Software Engineering Education: Challenges and Perspectives

Sofia Ouhbi
Dept. Computer Science & Software Eng.
CIT, UAE University
Al Ain, Abu Dhabi, UAE
sofia.ouhbi@uaeu.ac.ae

Nuno Pombo
Instituto de Telecomunicações
Universidade da Beira Interior
Covilhã, Portugal
ngpombo@ubi.pt

Abstract—**The software engineering community celebrated, in 2018, the 50th anniversary of what is considered to be the official start of the profession of software engineering. Software engineering is a young and promising discipline which is still under development and improvement. This is reflected when teaching software engineering in higher education. The aim of this study is to investigate the challenges and perspectives of software engineering education. To do so, a questionnaire study was conducted. 21 software engineering faculty and experts in teaching software engineering related courses participated in this study. The questionnaire contained demographic questions, questions related to students' engagement and to different methodologies adopted by respondents in the classroom. Results showed that the majority of respondents found engaging students in software engineering courses to be the biggest challenge they faced in the classroom. Almost half of the participants found difficulties designing practical activities for students. Results also revealed that the problem-based learning approach is the most used in software engineering lectures, followed by gamification techniques and role-playing which are new trends used to engage students. Moreover, the majority of the participants considered that the adoption of new teaching methodologies in the classroom produced high impact in the students' learning experience. Based on the outcomes of this questionnaire study, a conceptual model to engage students in software engineering courses is proposed. For future work, complementary studies should be implemented to evaluate the proposed model in a real-world scenarios including its effect on the achievement of learning outcomes.**

Index Terms—**Software Engineering, Teaching Methodologies, Students' engagement, Education**

I. Introduction

Software engineering education aims to bring together theory and practice, so that the learner may develop a deep understanding on cornerstone concepts and principles, along with skills and competences to solve real-world problems. Thus, it is not merely an academic concern about teaching relevant topics, but also a responsibility to shape skilled individuals adequate for the industry demand. Moreover, due to the fact that the software industry is developing rapidly and becoming transnational [1], several challenges are facing the qualification of software engineers, capable to develop products according to the international industrial standards into overseas markets.

Despite the celebration of its 50th anniversary in 2018, software engineering is a young and promising discipline which is still under development and improvement, which is also the case for software engineering education. In the last years, several pedagogical approaches had been implemented aiming at improving learners' motivation and engagement [2]. The increasingly adoption of technology in the classroom disrupted traditional models in terms of lecturer-learner interaction, group dynamics, content delivering, and learning experience, just to mention a few. The rationale behind these pedagogical approaches lies in the attempt to produce innovative solutions that may enable place-based or remotely personalized and self-paced learning, in which, the lecturer may arrange appropriate learning activities to promote student thinking and further enhancing their cognitive levels and problem solving abilities.

Software engineering education requires both hard and soft skills. Technical competencies such as requirements engineering, modelling, programming, and testing are desirable. The ability to communicate, to cooperate, to connect pieces of knowledge to discover solutions, and to think critically are also strongly recommended. Since soft skills develop over time and require practice, it is promising that its awareness and learning may be provided in advance throughout a students' coursework and infused into software engineering curriculum. In the recent years, new approaches such as the formation of interdisciplinary teams composed by software engineering students and entrepreneurship students [3], or the stimulus for prototyping adoption through the end-user experience driven [4, 5], were mentioned in the literature.

The instruction of theoretical concepts, and practical skills in an individual-centered learning environment is an ideal teaching strategy for students [4], which is why several body of knowledge on the software engineering context were proposed. The most relevant ones are the Guide to the Software Engineering Body of Knowledge (SWEBOK) [6], and the ACM/IEEE Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [7]. Both guides aim to meet the challenge of rapidly changing landscape of software engineering, and to aggregate the diversity, and complementary of new emerging domains that interconnect with software engineering.

The overall objective of this study is to investigate the challenges facing academic professionals in teaching software engineering and to propose a conceptual framework to improve software engineering courses delivery in the classroom. The remainder of this paper is organized as follows: Section II introduces related works, including some examples and scenarios with a special focus on students engagement in the classroom; in Section III, the questionnaire deployed in this study is presented; Section IV presents the results obtained from the questionnaire study; Section V discusses the results and presents our proposed framework; and Section VI concludes the paper and presents future works related to this study.

II. Background

Recent pedagogical trends such as the students' empowerment, engagement, and motivation replaced the old-fashioned models to deliver software engineering lectures. This paved the way for adoption of technologies in the classroom which may represents a double challenge. On the one hand, unique and/or complementary technologies must be wisely used to shape an effective and profitable learning tool; capable to enhance students' soft and hard skills, instead an additional source of distraction. On the other hand, learning contents and other materials may be delivered not only in a digital format, but also through different education scenarios such as traditional place-based, online, or blended (combining place-based and online). On top of that, modern students expect a personalized and self-paced learning. Software engineering education is expected to prepare students ready to step directly into software developer positions and succeed.

In [8], authors highlighted the gap between academia and industry as a motivation for the introduction of an educational program. The main goal is to provide an immerse learning experience that may guide students for complementary skills achievement required by the curriculum and by industry. Congruently, [9] also focused in the gap between academia and industry in which the balance between hard and soft skills through realistic projects was recommended on the learning process. In [10], authors proposed a framework; called Essence, to describe the commonality and the diversity of software engineering practices. The ultimate goal is to enhance the learning process by means of providing tools that may enable students to learn and to understand the diversity of software engineering later in their career.

Moreover, in [11], authors proposed a Software Engineering Competency Assessment Tool (SECAT), to assess the achievement of educational learning outcomes and the acquisition of competences related with either hard or soft skills. In [12], authors presented the challenge for the Global Software Engineering (GSE) education highlighting the most relevant Global Teaming Model (GTM) practices. In this study a more active role of SMEs in the education process is recommended. While, in [13], a trans-domain competencies and complementary learning approach is proposed by mans of combine technical specifications and business.

In [14], the validation of software engineering student's learning within their rich professional context is proposed for the adult higher education at a distance. This study highlighted the adaptability of teaching software engineering techniques based on the continuous assessment of problem-oriented practices. In [15] authors reinforce this concept, proposing real-world software development projects in order to enhance students' skills.

In [16], the authors, based on 10-years experience in teaching software engineering, highlighted that soft skills (such teamwork) were critical for success. However, large class sizes challenges for group dynamics and/or limits the adaptability of teaching software engineering. In [17], the authors proposed to use videos in software engineering education, and in [18] the authors highlighted the relevance in teaching software measurement in software engineering courses. Moreover, [19] proposed a learning process to encourage students to avoid multitasking and focus on learning multi-step tasks. The main purpose is to adopt athletic training principles (such as time optimization) in software engineering education.

III. Materials and Methods

This study aims to investigate the challenges and perspectives of software engineering education. In line with this, we designed a research questionnaire, presented in Table I, to get an overall understanding about the different approaches used to teach software engineering and challenges facing the respondents.

The questionnaire was sent to 28 software engineering faculty and experts in teaching software engineering related courses in early September 2019 and was available online for a period of two weeks. An invitation to participate in this survey was sent by the authors to colleagues and authors of papers identified in proceedings of an international software engineering conference. A total of 21 participants answered the online questionnaire, which was created using Google Forms and composed of 9 items segmented into two sections as presented in Table I.

The first section of the questionnaire was composed of four sociodemographic background questions to capture factors that might influence respondents' answers. The second part of the questionnaire contains 5 closed and short questions related to the topic of this paper. The survey was conducted anonymously. An estimated completion time of the online questionnaire was 5 minutes.

Data visualization was done using Excel to represent answers from respondents.

IV. Results

Only 21 out of 28 of potential respondents have participated in our study. The 75% completion rate maybe due to the fact that the invitation was sent via email. Although a second email was sent to remind invitees at the beginning of the second week, the number of responses did not change.

A. Demographics

Table II presents an overview of the respondents' demographics.

Table I
QUESTIONNAIRE. *Acronym: Software Engineering (SE)*

Section 1: *Demographic questions*
- Age:
- Gender:
- Job title:
- Country of Work:
Section 2: *Questions related to teaching software engineering courses*
What are the SE related courses you have taught?
- Software Eng Fundamentals
- Software Quality
- Requirements Engineering
- Software Architecture and Design
- Software Project Management
- Software Construction
- Software Testing
- Others: (Please specify)
How long have you been teaching SE related courses?
- 1-2 years
- 3-5 years
- 6-9 years
- More than 10 years
What are the challenges that you have faced in teaching SE courses?
- Engaging students
- Designing practice activities
- Finding adequate textbooks
- Finding adequate technology, and tools
- Others: (Please specify)
What are the approaches that you have integrated in your lecture to improve SE related courses?
- Gamification techniques
- Role playing
- Flipped classroom
- Problem based learning
- Others: (Please specify)
Do you think that this/these approaches made a difference in the learning experience of students?
- Yes, but it was a reduced impact
- Yes, it made a big difference
- No

Table II
BACKGROUND INFORMATION OF RESPONDENTS (N=21)

Characteristic	Value (%)
Age	
Younger than 30	0 (0%)
31-40	10 (48%)
41-50	8 (38%)
51-60	3 (14%)
Older than 60	0 (0%)
Gender	
Male	17 (81%)
Female	4 (19%)
Job title	
Full Professor	1 (4.8%)
Professor	8 (38.1%)
Associate Prof.	2 (9.5%)
Assistant Prof.	3 (14.3%)
Lecturer	3 (14.3%)
Software Engineer	2 (9.5%)
Research Assistant	2 (9.5%)
Country of Work	
Portugal	10 (47.6%)
Spain	5 (19%)
UAE	2 (9.5%)
Morocco	2 (9.5%)
Germany	1 (4.8%)
Finland	1 (4.8%)

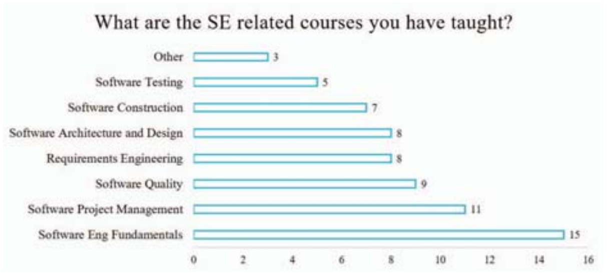

Figure 1. Results of research question 1. *Acronym: Software Engineering (SE)*

B. Research questions

Fig. 1 presents the results of the first research question. The majority of respondents have taught *Software Engineering Fundamentals* course, which is an introductory course for the principal concepts of Software Engineering, followed by *Software Project Management*. Less than half of the respondents have had experience teaching specialized software engineering courses: 43% have taught *Software Quality*, 38% have taught *Requirements Engineering*, same percentage taught *Software Architecture and Design*, 33% have taught *Software Construction*, while only 24% have taught *Software Testing* course. Three respondents listed other courses: *Information Systems Security Management* course; *Algorithms and Data Structures*, which is a Computer Science course; and another course about multidisciplinary projects involving software development.

Fig. 2 shows that the majority of the respondents have taught software engineering courses for more than 6 years. Almost one third of participants has had experience teaching software engineering courses for more than 3 years and less than 5 years, while 19% has had less then two years experience teaching these courses.

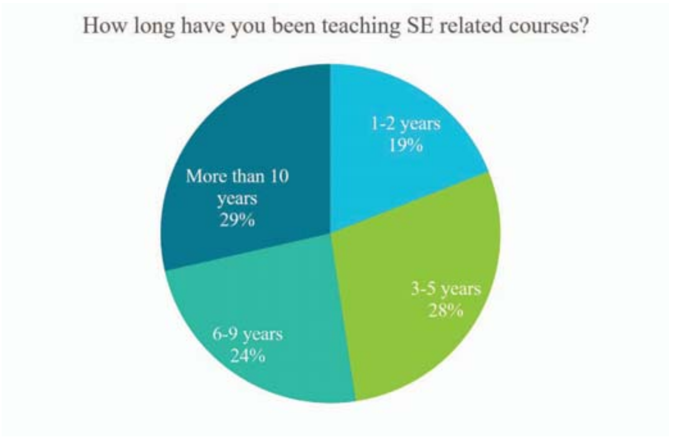

Figure 2. Results of research question 2. *Acronym: Software Engineering (SE)*

Fig. 3 presents the challenges that have faced the respondents in teaching software engineering courses. The main challenge faced is engaging students in the classroom followed by the challenge to design practice activities for the software engineering courses. Five participants found it challenging to find adequate textbooks and the same number found difficulties finding adequate technology, and tools to incorporate in software engineering courses. Three participants suggested other challenges: (i) create assessments/exams that go beyond memorization; (ii) there is a big discrepancy between textbooks and practical life, students tend to do well on the test but in practice they are lost; and (iii) too much material to cover.

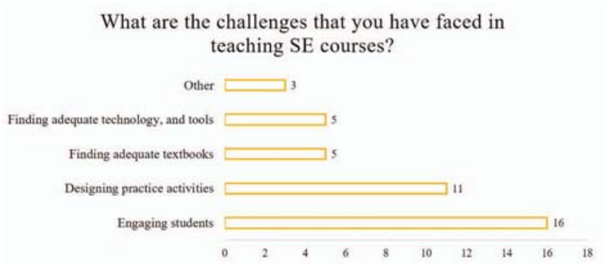

Figure 3. Results of research question 3. *Acronym: Software Engineering (SE)*

Table III links between the challenges and number of years of experience teaching software engineering courses.

Table III
NUMBER OF YEARS OF EXPERIENCE VS CHALLENGES FACED

Years of experience	Challenges
1-2 years	Engaging students, Designing practice activities, Too much material to cover.
3-5 years	Engaging students, Designing practice activities, Finding adequate textbooks, Create assessments/exams that go beyond memorization, Finding adequate technology, and tools.
6-9 years	Engaging students, Designing practice activities, Finding adequate technology, and tools, There is a big discrepancy between textbooks and practical life. Students tend to do well on the test but on practice they are lost.
More than 10 years	Engaging students, Designing practice activities, Finding adequate textbooks, Finding adequate technology, and tools.

Fig. 4 presents approaches adopted by the respondents to improve their software engineering courses. The majority uses *problem-based learning* approach, 43% integrate *gamification techniques* in their courses, 29% use *role playing*, same percentage use the *flipped classroom* approach, while 19% use different approaches: serious games, audience response systems, and storytelling using real-world problems and examples from industry.

Table IV shows the relation between the respondents years of experience and the approach adopted to improve teaching software engineering course.

Fig. 5 presents the respondents' opinion about the efficiency of the approaches adopted in their classroom. 71% of respondents have found that the implemented approaches have significantly improved their teaching of software engineering courses. Only 24% have experienced a slight improvement in

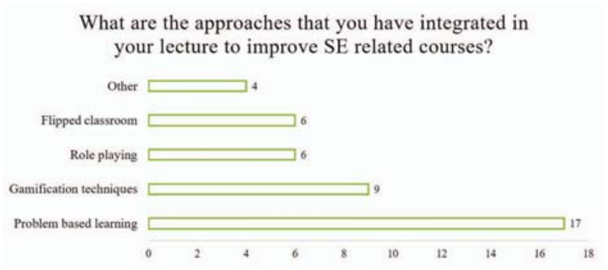

Figure 4. Results of research question 4. *Acronym: Software Engineering (SE)*

Table IV
YEARS OF EXPERIENCE VS APPROACH TO IMPROVE SOFTWARE ENGINEERING COURSES

Years of experience	Approaches adopted
1-2 years	Role playing, Flipped classroom, Problem based learning, serious games, audience response systems
3-5 years	Role playing, Problem based learning, Gamification techniques, Storytelling
6-9 years	Gamification techniques, Flipped classroom, Problem based learning
More than 10 years	Role playing, Gamification techniques, Flipped classroom, Problem based learning, Interactive systems for audience response

their courses. One respondent stated that it depends on the students but generally adopting new approaches improves the learning experience.

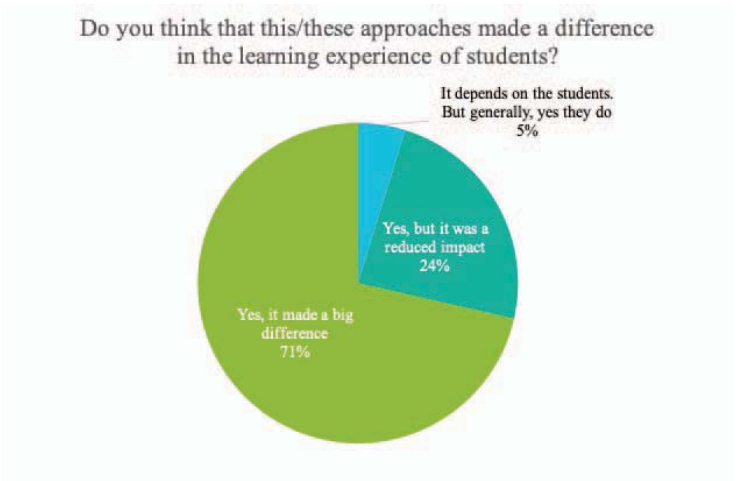

Figure 5. Results of research question 5

Table V shows the relationship between the approaches and their efficiency according to the respondents.

V. DISCUSSION

A. Principal findings

Engaging students is one of the main challenges in higher education [20], and is considered critical for students achievement in a course [21]. Engaging students in software engineering courses is particularly challenging as these courses

Table V
APPROACHES ADOPTED VS THEIR EFFICIENCY

Approach adopted	**Feedback on their efficiency**
Gamification techniques, Problem based learning, Storytelling	Reduced impact
Role Playing, Gamification techniques, Flipped classroom, Problem based learning, Interactive systems for audience response, serious games	Big difference
Problem based learning	It depends on the students

contain heavy theoretical notions with a few practical work in the classroom. For this reason many software engineering researchers have started focusing on approaches to engage students [22, 23].

Bridging the gap between the classroom and the real world is also one of the biggest challenges that instructors face in software engineering courses [24]. It is challenging to design practical work and finding adequate technologies and tools to integrate in the classroom. Numerous tools used in the real world by software engineers are not free and not suitable for education [25]. However, instructors can use a list of open source tools provided by Toth [26], which can be used to give students practical software engineering experience.

One of the challenges in teaching software engineering is finding adequate textbooks. Although guides like SWEBOK [6] exist but it is difficult to design a whole course based on the definition available only in them. Often instructors tend to combine more than one source to prepare their lecture and find it difficult to rely only on one book. The traditional textbooks available for software engineering related courses are usually lengthy books, for instance this requirements engineering book [27] is 800-page long.

The majority of the approaches when adopted in the classroom had significantly improved the teaching experience of the respondents. A traditional approach is problem-based learning, which was the mostly used by the participants. Adopting this approach by instructors can be challenging as it requires them to shift from the position of knowledge provider to facilitator of learning [28]. However, its efficiency has been demonstrated in helping students understanding practical aspects of software engineering [29, 30].

Role playing efficiency has been demonstrated for requirements engineering courses in particular [31, 32, 33] and in software engineering courses in general [34]. The efficiency of using gamification in software engineering courses has also been studied in [35, 36]. However, a main challenge in using gamification is to decide which gamification elements to use in the classroom [37], this is why the results of our questionnaire found that gamification can have a slight impact in some cases compared to other cases where it can have a significant impact improving software engineering courses.

The effectiveness of using flipped classroom, also known as inverted classroom, in software engineering courses has been studied in [38, 39, 40]. The main challenge of this approach for instructors is the increase in workload compared to the traditional approaches [41]. While for students, challenges are experienced in the out-of-class activities [42].

Other approaches were also listed by participants such as the use of interactive system for audience response, such as Mentimeter [43], or OMBEA [44]. Audience response systems are used to allow students to "answer electronically displayed multiple choice questions using a remote control device" [45]. The use of such systems in software engineering courses has been studied in [46, 47], and is found it increases classroom dynamic and improves interaction with students [48].

Combining more than one technique shows better results however it might be time consuming and requires much more effort from the instructor. Experience also plays an important role in teaching software engineering, instructors with more than ten years of experience teaching software engineering courses are the most respondents to use more than one technique in their classroom. This might be explained by the fact that they had more time over the years to integrate new approaches to teach software engineering than the rest of respondents.

B. *Proposed framework*

Based on the finding of the questionnaire and the literature studied on the efficiency of each approach, we propose the framework shown in Fig. 6 which combines SE courses, teaching approaches and the rational for its adoption. On the one hand, seven software engineering courses are included in the framework: Requirements Engineering, Software Project Management, Software Quality, Software Testing, Software Construction, Software Architecture & Design, and Software Engineering Fundamentals. On the other hand, five teaching approaches are recommended, namely: Role Playing, Gamification, Problem-based Learning, Flipped Classroom, and Automatic Response Systems. Due to its simplicity, Automatic Response Systems may be applied in every course as a tool to empower audiences by means of real-time interaction. In addition, Flipped Classroom is proposed as an approach to ease the delivering of software engineering principles. Moreover, Gamification, and Problem-based Learning are suggested to deliver a practical experience on students. Finally, Role Playing is suggested to deliver a complementary overview on a specific topic.

VI. CONCLUSION

This study presented the results of a questionnaire to investigate the challenges facing instructors of software engineering courses. 21 respondents gave the challenges they face and the teaching approaches they have tried to improve software engineering courses. The main challenge is engaging students in the classroom while the main approach used in courses is problem-based learning. Based on the questionnaire's results, a conceptual framework is proposed to link each course with a technique that can be used to improve the learning experience of students. For future work, we intend to investigate more the applicability of the framework to make further adjustment in it.

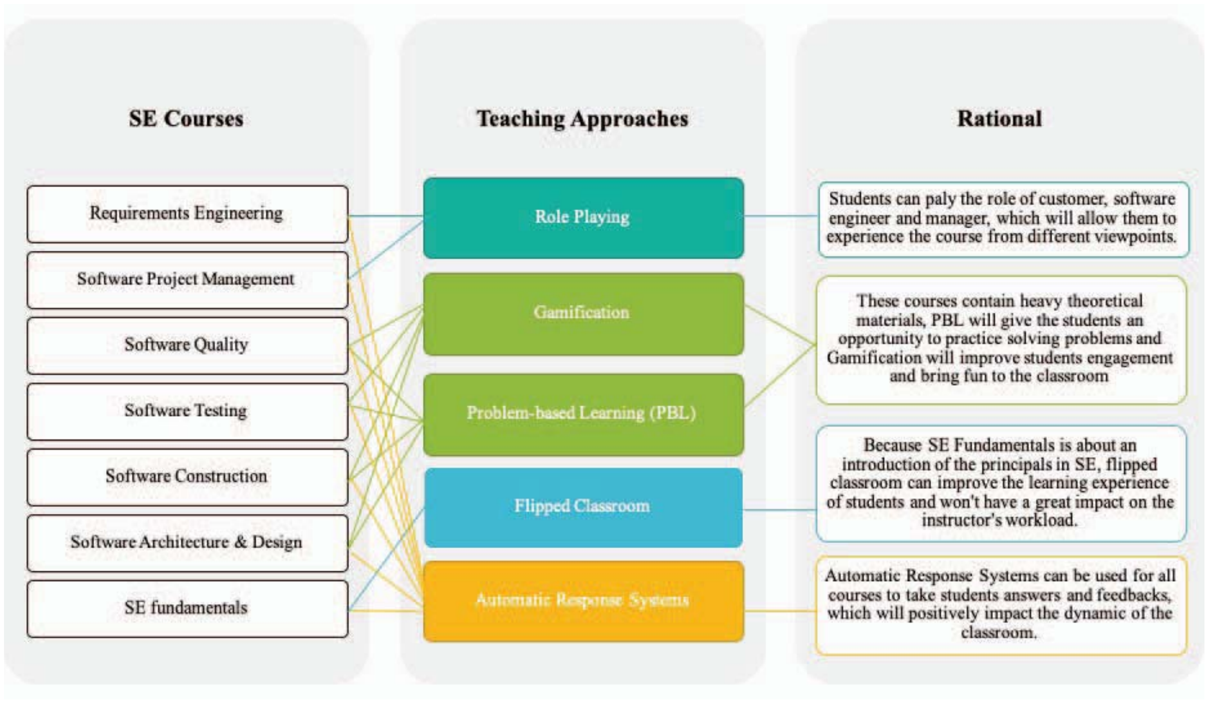
SE Courses
Teaching Approaches
Rational
Requirements Engineering
Software Project Management
Software Quality
Software Testing
Software Construction
Software Architecture & Design
SE fundamentals
Role Playing
Gamification
Problem-based Learning (PBL)
Flipped Classroom
Automatic Response Systems
Students can paly the role of customer, software engineer and manager, which will allow them to experience the course from different viewpoints.
These courses contain heavy theoretical materials, PBL will give the students an opportunity to practice solving problems and Gamification will improve students engagement and bring fun to the classroom
Because SE Fundamentals is about an introduction of the principals in SE, flipped classroom can improve the learning experience of students and won't have a great impact on the instructor's workload.
Automatic Response Systems can be used for all courses to take students answers and feedbacks, which will positively impact the dynamic of the classroom.

Figure 6. Framework to integrate approaches in software engineering courses. *Acronym: Software Engineering (SE)*

ACKNOWLEDGMENTS

The author affiliated with the *Instituto de Telecomunicações* would like to acknowledge that this work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020 (Este trabalho é financiado pela FCT/MCTES através de fundos nacionais e quando aplicável cofinanciado por fundos comunitários no âmbito do projeto UIDB/EEA/50008/2020).

REFERENCES

[1] Philip G. Altbach and Jane Knight. “The Internationalization of Higher Education: Motivations and Realities”. In: *Journal of Studies in International Education* 11.3-4 (2007), pp. 290–305. DOI: 10.1177/1028315307303542. eprint: https://doi.org/10.1177/1028315307303542. URL: https://doi.org/10.1177/1028315307303542.

[2] N. Pombo, N. Garcia, and P. Alves. “How to Get a Badge? Unlock Your Mind : Motivation through Student Empowerment”. In: *2019 IEEE Global Engineering Education Conference (EDUCON)*. Apr. 2019, pp. 115–119. DOI: 10.1109/EDUCON.2019.8725146.

[3] Kevin Buffardi, Colleen Robb, and David Rahn. “Tech Startups: Realistic Software Engineering Projects with Interdisciplinary Collaboration”. In: *J. Comput. Sci. Coll.* 32.4 (Apr. 2017), pp. 93–98. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=3055338.3055355.

[4] I. Matias et al. “Scaffolding students on connecting STEM and interaction design: Case study in Tallinn University Summer School”. In: *2018 IEEE Global Engineering Education Conference (EDUCON)*. Apr. 2018, pp. 979–987. DOI: 10.1109/EDUCON.2018.8363336.

[5] N. Pombo et al. “Design and evaluation of a decision support system for pain management based on data imputation and statistical models”. In: *Measurement* 93 (2016), pp. 480–489. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2016.07.009. URL: http://www.sciencedirect.com/science/article/pii/S0263224116303682.

[6] IEEE Computer Society, Pierre Bourque, and Richard E. Fairley. *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. 3rd. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014. ISBN: 0769551661, 9780769551661.

[7] The Joint Task Force on Computing Curricula. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Tech. rep. New York, NY, USA, 2004.

[8] Scott Heggen and Cody Myers. “Hiring Millennial Students As Software Engineers: A Study in Developing Self-confidence and Marketable Skills”. In: *Proceedings of the 2Nd International Workshop on Software Engineering Education for Millennials*. SEEM ’18. Gothenburg, Sweden: ACM, 2018, pp. 32–39. ISBN: 978-1-4503-5750-0. DOI: 10.1145/3194779.3194780. URL: http://doi.acm.org/10.1145/3194779.3194780.

[9] D. Oguz and K. Oguz. "Perspectives on the Gap Between the Software Industry and the Software Engineering Education". In: *IEEE Access* 7 (2019), pp. 117527–117543. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2936660.

[10] P. Ng and S. Huang. "Essence: A framework to help bridge the gap between software engineering education and industry needs". In: *2013 26th International Conference on Software Engineering Education and Training (CSEE T)*. May 2013, pp. 304–308. DOI: 10.1109/CSEET.2013.6595266.

[11] Y. Sedelmaier and D. Landes. "A multi-perspective framework for evaluating software engineering education by assessing students' competencies: SECAT — A software engineering competency assessment tool". In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. Oct. 2014, pp. 1–8. DOI: 10.1109/FIE.2014.7044331.

[12] Sarah Beecham, Tony Clear, and John Noll. "Do We Teach the Right Thing?: A Comparison of Global Software Engineering Education and Practice". In: *Proceedings of the 12th International Conference on Global Software Engineering*. ICGSE '17. Buenos Aires, Argentina: IEEE Press, 2017, pp. 11–20. ISBN: 978-1-5386-1587-4. DOI: 10.1109/ICGSE.2017.8. URL: https://doi.org/10.1109/ICGSE.2017.8.

[13] A. Adorjan and M. Nunez-Del-Prado. "Fostering 21 Century Learning and Innovation Competencies Through Students' Online Collaborative Activities in Software Engineering Courses". In: *2018 IEEE World Engineering Education Conference (EDUNINE)*. Mar. 2018, pp. 1–4. DOI: 10.1109/EDUNINE.2018.8450987.

[14] Jon G. Hall and Lucia Rapanotti. "Masters-level Software Engineering Education and the Enriched Student Context". In: *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. ICSE '15. Florence, Italy: IEEE Press, 2015, pp. 311–314. URL: http://dl.acm.org/citation.cfm?id=2819009.2819058.

[15] J. Vanhanen, T. O. A. Lehtinen, and C. Lassenius. "Teaching real-world software engineering through a capstone project course with industrial customers". In: *2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*. June 2012, pp. 29–32. DOI: 10.1109/EduRex.2012.6225702.

[16] S. Heckman, K. Stolee, and C. Parnin. "10+ Years of Teaching Software Engineering with iTrust: The Good, the Bad, and the Ugly". In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. May 2018, pp. 1–4.

[17] A. Alaboudi and T. D. LaToza. "Supporting Software Engineering Research and Education by Annotating Public Videos of Developers Programming". In: *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. May 2019, pp. 117–118. DOI: 10.1109/CHASE.2019.00035.

[18] M. Villavicencio and A. Abran. "The necessary software measurement knowledge in software engineering education from the practitioners' point of view". In: *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. Apr. 2012, pp. 1–5. DOI: 10.1109/CCECE.2012.6335058.

[19] E. Hill, P. M. Johnson, and D. Port. "Is an Athletic Approach the Future of Software Engineering Education?" In: *IEEE Software* 33.1 (Jan. 2016), pp. 97–100. ISSN: 1937-4194. DOI: 10.1109/MS.2016.15.

[20] Stephen John Quaye, Shaun R Harper, and Sumun L Pendakur. *Student engagement in higher education: Theoretical perspectives and practical approaches for diverse populations*. Routledge, 2019.

[21] Mitchell M Handelsman et al. "A measure of college student course engagement". In: *The Journal of Educational Research* 98.3 (2005), pp. 184–192.

[22] Mehmet Kosa et al. "Software engineering education and games: a systematic literature review". In: *Journal of Universal Computer Science* 22.12 (2016), pp. 1558–1574.

[23] Sofia Ouhbi et al. "Requirements engineering education: a systematic mapping study". In: *Requirements Engineering* 20.2 (2015), pp. 119–138.

[24] J Huffman Hayes. "Energizing software engineering education through real-world projects as experimental studies". In: *Proceedings 15th Conference on Software Engineering Education and Training (CSEE&T 2002)*. IEEE. 2002, pp. 192–206.

[25] Klaus Alfert, Jörg Pleumann, and J Schroder. "Software engineering education needs adequate modeling tools". In: *17th Conference on Software Engineering Education and Training, 2004. Proceedings.* IEEE. 2004, pp. 72–77.

[26] Kal Toth. "Experiences with open source software engineering tools". In: *IEEE software* 23.6 (2006), pp. 44–52.

[27] Klaus Pohl. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[28] John R Savery. "Overview of problem-based learning: definition and distinctions, the interdisciplinary". In: *Journal of Problem-based learning*. Citeseer. 2006.

[29] Joanna C Dunlap. "Problem-based learning and self-efficacy: How a capstone course prepares students for a profession". In: *Educational Technology Research and Development* 53.1 (2005), pp. 65–83.

[30] George G Mitchell and James Declan Delaney. "An assessment strategy to determine learning outcomes in a software engineering problem-based learning course". In: *International Journal of Engineering Education* 20.3 (2004), pp. 494–502.

[31] Sofia Ouhbi. “Evaluating Role Playing Efficiency to Teach Requirements Engineering”. In: *2019 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2019, pp. 1007–1010.

[32] Richard Berntsson Svensson and Björn Regnell. “Is role playing in Requirements Engineering Education increasing learning outcome?” In: *Requirements Engineering* 22.4 (2017), pp. 475–489.

[33] Peng Liang and Onno De Graaf. “Experiences of using role playing andwiki in requirements engineering course projects”. In: *2010 5th International Workshop on Requirements Engineering Education and Training*. IEEE. 2010, pp. 1–6.

[34] Tyson R Henry and Janine LaFrance. “Integrating role-play into software engineering courses”. In: *Journal of Computing Sciences in Colleges* 22.2 (2006), pp. 32–38.

[35] Oscar Pedreira et al. “Gamification in software engineering–A systematic mapping”. In: *Information and software technology* 57 (2015), pp. 157–168.

[36] Bilal Sercan Akpolat and Wolfgang Slany. “Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification”. In: *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. 2014, pp. 149–153.

[37] Manal M Alhammad and Ana M Moreno. “Gamification in software engineering education: A systematic mapping”. In: *Journal of Systems and Software* 141 (2018), pp. 131–150.

[38] Gerald Gannod, Janet Burge, and Michael Helmick. “Using the inverted classroom to teach software engineering”. In: *2008 ACM/IEEE 30th International Conference on Software Engineering*. IEEE. 2008, pp. 777–786.

[39] Ashish Amresh, Adam R Carberry, and John Femiani. “Evaluating the effectiveness of flipped classrooms for teaching CS1”. In: *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2013, pp. 733–735.

[40] Pang Nai Kiat and Yap Tat Kwong. “The flipped classroom experience”. In: *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. 2014, pp. 39–43.

[41] Thomas Wanner and Edward Palmer. “Personalising learning: Exploring student and teacher perceptions about flexible learning and assessment in a flipped university course”. In: *Computers & Education* 88 (2015), pp. 354–369.

[42] Gökçe Akçayır and Murat Akçayır. “The flipped classroom: A review of its advantages and challenges”. In: *Computers & Education* 126 (2018), pp. 334–345.

[43] *Mentimeter. URL: https://www.mentimeter.com/ Accessed on Feb 2020.*

[44] *OMBEA. URL: https://www.ombea.com/ Accessed on Feb 2020.*

[45] Robin H Kay and Ann LeSage. “Examining the benefits and challenges of using audience response systems: A review of the literature”. In: *Computers & Education* 53.3 (2009), pp. 819–827.

[46] José A Sánchez et al. “An Approach for Automated Software Engineering Competence Measurement: Model and Tool”. In: *International Journal on Information Technologies & Security* (2017).

[47] Valentin Razmov and Richard Anderson. “Pedagogical techniques supported by the use of student devices in teaching software engineering”. In: *ACM SIGCSE Bulletin*. Vol. 38. 1. ACM. 2006, pp. 344–348.

[48] Alf Inge Wang. “The wear out effect of a game-based student response system”. In: *Computers & Education* 82 (2015), pp. 217–227.