Combining Agile and DevOps to Improve Students' Tech and Non-tech Skills

Telcio Cardoso[a], Rafael Chanin[b], Alan R. Santos[c] and Afonso Sales[d]
School of Technology, Pontifical Catholic University of Rio Grande do Sul, Zip Code 90619-900, Porto Alegre, RS, Brazil

Keywords: Software Engineering Education, Computer Science Education, Computer Science Curricula, DevOps, Scrum, Challenge Based Learning.

Abstract: The goal of this ongoing study is to understand how the development of soft skills is approached in the existing computer science and software engineering curricula. Based on the findings and improvement opportunities, we propose a high-level course structure to be used as a framework for those higher education organizations who want to support the development of students technical and non-technical skills. With this proposal, we believe computer science and software engineering students would be better prepared for the most recent IT market requirements, fostering the development of the 21st century's main competencies. In order to develop the course structure presented in this study, we conducted a literature research, which showed the development of students' soft skills still requires improvement and more efficient approaches. Our solution proposal combines DevOps, Scrum and Challenge Based Learning approaches into one single course, which uses Agile DevOps culture and values, along with continuous feedback, to promote students' soft skills development.

1 INTRODUCTION

According to the Computer Engineering Curricula 2016[1], industry leaders in general understand that soft skills are essential for hiring in a computing-related position (on Information Technology Curricula, 2017). Additionally, a common sense in the industry indicates that soft skills and technical skills have similar importance (on Information Technology Curricula, 2017). The Computer Science Curricula 2013 (Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, 2013) argues that students will acquire some important soft skills and personal attributes through general college activities (*e.g.* patience, time management, work ethic, and an appreciation for diversity), and others through a specific curriculum. The ACM Cybersecurity Curricula 2017 (on Cybersecurity Education, 2018) emphasizes the importance of developing technical skills and soft skills in order to be ready to the transition from academic environments to a career within a corporation, organization, academic institution or even an entrepreneurial environment.The ACM Information Technology Curricula 2017 (on Information Technology Curricula, 2017) argues that the ability to connect to co-workers in a convincing manner will be extremely important in the future. Still according to the ACM Information Technology Curricula 2017 (on Information Technology Curricula, 2017), college and universities are well positioned to teach IT technical skills, however, they encounter challenges when teaching non-technical and soft skills.

Considering the importance of soft skills for computer science and software engineering students' career and personal development, we conducted a literature research in order to understand how soft skills are taught in higher education courses. Based on our study results, we proposed a course to complement the current Computer Science 2013 curricula guidelines, providing a more prescriptive structure to evaluate and develop soft skills, using DevOps and Agile practices as foundation to improve such skills.

1.1 Problem Statement

Hazzan (Hazzan and Har-Shai, 2014) argue that a simple web search reveals that most of the issues

[a] https://orcid.org/0000-0002-6388-2448
[b] https://orcid.org/0000-0002-6293-7419
[c] https://orcid.org/0000-0001-8323-3472
[d] https://orcid.org/0000-0001-6962-3706

[1] ACM Education Curricula Recommendations: https://www.acm.org/education/curricula-recommendations

Cardoso, T., Chanin, R., Santos, A. and Sales, A.
Combining Agile and DevOps to Improve Students' Tech and Non-tech Skills.
DOI: 10.5220/0010401302990306
In *Proceedings of the 13th International Conference on Computer Supported Education (CSEDU 2021) - Volume 1*, pages 299-306
ISBN: 978-989-758-502-9

associated with software development processes are connected to people and that their origin is rooted in cognitive, social and managerial aspects rather than technical or technological aspects. In this sense, software continuous delivery approaches, such as Scrum and DevOps, are widely used in the software industry, from startups to established companies. By being not-siloed based, in order to be successful, teams that use such approaches may require higher-levels of soft skills from their team members.

Soft skills are a very important requirement in the software development industry and such importance is increased by new continuous delivery approaches such as DevOps, but also on the collaboration among different specialists and stakeholders (Kuusinen and Albertsen, 2019).

Even understanding that soft skills are important, and students will gain some of them over the computer science course, there is not a clear guideline on how to implement and develop such skills and neither a clear course structure to evaluate and improve such skills.

Soft skills are an important part of students' development. However, they are generally neglected in the computer science and software engineering curricula in the majority of the higher education institutions, which understand the importance of such skills, however, lack of a more prescriptive method to foster and measure the development of such skills. Considering the importance of soft skills to the students personal and professional live, the main question this study aims to answer is:

Can we develop a new course to foster computer science and software engineering. students' soft skills development, preparing them to the market, by using industry emerging practices?

In order to further understand the usage of this combination of Scrum and DevOps in the development of students' soft skills, we have performed a literature research and presented a course structure proposal to support the development of students' soft skills in higher education. The course teaches students using an active learning framework: Challenge Based Learning (Nichols et al., 2016). Our literature research indicates that the development of soft skills for higher education students of computer science and software engineering courses is still an area to be improved and lacks of more prescriptive course structures.

The remainder of this paper is organized as follows. Section 3 presents the research method applied in this study, and in Section 4 we present relevant findings from the selected studies. In Section 5, we present the proposed course and its structure. Finally, in Section 6 we conclude the paper with some final thoughts and future work.

2 BACKGROUND

2.1 Challenge Based Learning

Teaching students can be done in multiple ways. Traditionally, learning is mostly based on lectures, a teacher-centered approach which usually provides low levels of interaction. On the other hand, active learning is an approach that proposes high levels of interaction and stimulates students to perform not only low-order cognitive tasks, such as reading and writing, but also high-order ones, including debating, analysing and decision making (Fetaji and Fetaji, 2009; Ahmad and Gestwicki, 2013; Gestwicki and Ahmad, 2011).

There are several active learning methodologies that have been used in an educational setting. Problem Based Learning, Project Based Learning, Task Based Learning and Challenge Based Learning are just a few examples of these frameworks. *"The foundations of experiential learning can be found within the history of most cultures, but were formally organized and presented by David Kolb drawing heavily on the works of John Dewey and Jean Piaget"* (Santos et al., 2015a). Challenge Based Learning (CBL) (Nichols et al., 2016) is a learning framework based on solving real world challenges.

The CBL process begins with the definition of a *big idea*, which is a broad concept that can be explored in several ways. The big idea has to be engaging and important to students. Once the big idea is chosen and the *essential question* is created, the *challenge* is defined. From this point, students must come up with the *guiding questions* and *guiding activities and resources*, which will guide them to develop a successful solution. The next step is *analysis*, which will set the foundation for the definition of the *solution*. Once the solution is agreed upon, the *implementation* begins. Finally, *evaluation* is undertaken in order to check out the whole process and verify if the solution can be refined.

2.2 DevOps

Commonly in medium and large companies, software development and IT operations teams are structured in silos and the collaboration between these

teams use to occur when operational resources are required from software development teams in order to release new applications or functionalities, or, when new released applications somehow compromise the integrity of environments managed by IT operations teams. Therefore, we have software development teams aiming to deliver software to end-users as fast as possible, however, such need for speed and faster deliveries, sometimes, may compromise the integrity of IT operations environments and, IT operations teams, in the opposite side, aiming to ensure integrity and stability of production environments, what may conflict with software development teams goals.

In order to handle these conflicting goals between software development and IT operations teams, sustaining the value delivery pace required by the market, new practices were required. At the 2008 Agile conference in Toronto, Canada, Patrick Debois and Andrew Schafer presented a session to share their experience and research on applying Agile principles to IT infrastructure, what would be later give origin to the term *DevOps*, coined by Debois.

Husssani (Hussaini, 2014) defines DevOps as an acronym for Development (Dev) and Operations (Ops) of information technology systems and applications. Swartout et al. (Swartout, 2012) argues that DevOps is a way of working that encourages the collaboration between Development and Operations teams towards the same goal.

An important aspect to take into consideration regarding DevOps is the fact that the movement is not restricted to just a few IT operations teams. As Davis et al. (Davis and Daniels, 2016) argue, there is no one definitive list of teams or roles which should be involved or how and, any team within the organization should be considered in order to be more effective, which includes security, support, QA, legal, among other roles.

3 RESEARCH METHOD

In order to understand how students' soft skills are developed by Computer Science and Software Engineering courses in higher-education organizations, we performed a literature research.

The search terms were defined based on the main terms related to the subject of this research and structured in terms of population and intervention (Kitchenham, 2004). Based on these terms, the final search string used in this study is presented in Table 1.

We have evaluated papers, articles, journals and magazines, written in English language, with more than 3 pages and published between 2010 and 2020. The search string has been used to find relevant information at the ACM and IEEE online databases. Our research resulted in 23 studies, from which 3 studies have been considered relevant. Additionally, we have complemented our initial research with a web search, which resulted in 11 additional relevant studies which have their contribution added to our study.

Table 1: Search String.

Type	String
Population	software engineering OR computer science
Intervention	AND soft skills AND devops

4 FINDINGS

The following paragraphs provide details of the most relevant contribution from the studies we selected.

Hazzan et al. (Hazzan and Har-Shai, 2014) conducted a study at the Israeli Computer Science department, the largest CS department in Israel. The study describes the experience related to a 14 weeks course and was designed for both CS and SE students, first, to expose them to a variety of soft skills, and second, to enable them to acquire these skills gradually by experiencing them and reflecting upon them. The course structure was based on the CM/IEEE-CS Computer Science Curricula 2013 recommendations. The authors argue that over the course, the following skills were mentioned by at least 25% of students as being important for teamwork: listening skills, different aspects of teamwork, interpersonal communication, giving feedback, flexibility, time and pressure management, knowledge transfer, and patience. The study also provided some interesting insights regarding the soft skills mentioned by the students as the ones they would like to improve, such as Time Management, Presentation Skills, Creativity, among others. The authors still argue that students repeatedly expressed the idea that soft skills should be learned by expressing and experiencing them.

Sedelmaier (Sedelmaier and Landes, 2014) conducted a study to evaluate which soft skills are important for software engineers and proposed a body of skills (SWEBOS). The authors argue that SWEBOS treats skills in a merely descriptive fashion. The authors still argue that there is no guideline as how to break down required competencies, be it technical or non-technical, into teaching goals that can be addressed in university education for future software engineers. The study concluded the top three soft skills in software engineering are comprehension

of the complexity of software engineering processes, awareness of problems and understanding of cause-effect relationships, and team competence including communication skills.

Ahmed (Ahmed et al., 2012) surveyed jobs advertised on the following online portals: *workopolis.ca* (North America), *eurojobs.com* (Europe), *monsterindia.com* (Asia), and *seek.com.au* (Australia). The results presented the main soft skills required by the software market in North America, Europe, Asia and Australia in 2012. The main soft skills were interpersonal skills, analytical and problem-solving, team playing, organizational skills, fast learning, ability to work independently, innovative, open and adaptable to change.

In the study conducted by Carter (Carter, 2011), employment ads for software engineer of 50 companies were evaluated in order to identify the main soft skills required by employers. The study also applied a survey at Point Loma Nazarene Mathematical, Information and Computer Science university departments to understand the importance of soft skills from student's perspective. The study argues that communication appears to be the number one desired skill across the board, but employers explicitly ask for employment-ready computer science and engineering students to have other skills as well. The authors also mention that dedicated soft skill courses have proven to be extremely unpopular with both students and professors alike. The students feel that they are being presented with information that they already know. The study also provided details about a capstone course the university provides, which uses Project-Based Learning to practice and develop the soft skills required by the employers.

Matturro (Matturro et al., 2015) conducted a set of surveys with software development team members and team leaders from the Uruguay software industry and concluded that leadership, communication skills, customer orientation, interpersonal skills, and teamwork are the most valued for team leaders, while analytic, problem-solving, commitment, responsibility, eagerness to learn, motivation, and teamwork are the most valued ones for team members.

Kerzazi (Kerzazi and Adams, 2016), performed an empirical analysis of online job postings to determine and compare the main tasks of release and DevOps engineers, globally and across countries. According to authors, automation is the most important activity across the three roles, as articulated in job posting description data, and that the release engineer role combines the top activities of the DevOps and more traditional build engineer roles. The role of release engineer seems to combine the most important activities of DevOps and more traditional build engineers, either because of incorrect choice of role name or roles taking up more responsibilities than would be expected from them. Still according to the authors, although not an explicit variable in our analysis, build/DevOps/release engineers are needed by any kind of company, regardless of whether it is a startup or an established one (based on the job ad description). For start-ups, the responsibilities are more guided towards supporting development, followed by build, continuous integration, and fast delivery (building an effective pipeline of releases).

Valentin (Valentin et al., 2015) proposed a methodology to improve students' soft skills by using Scrum in a systematic project development experience. The methodology was applied to research groups from different areas. According to the authors, the Scrum framework offers several opportunities for students exercise different skills by means of well-structured events, roles, and artifacts. Writing, oral presentation, leadership, transparency, organization, and pace were the improvements most reported by the involved people.

Bastarrica (Bastarrica et al., 2017) developed a capstone course to support the development of critical soft skills to succeed in software development project, introducing some agile practices into an existing course, however, lacked details regarding which soft skills were evaluated and how to evaluate the improvements.

Andersson (Andersson and Logofatu, 2018) proposed an approach to improve the Master students' soft skills through a Mathematics Update course. An interesting insight provided by this study is related to the usage of discussion groups to align concepts about soft skills among the students. Another interesting approach mentioned by the authors is the self-evaluation survey, which allows the student to self-evaluate in order to have a baseline and better understand how they are improving over the course.

Thurner (Thurner et al., 2016) identified four reasons why students struggle with their soft skills and competencies during the first semesters and designed a learning project that addresses technical as well as non-technical competencies in an integrated way. One of the insights provided by this study is related to the practice of self-reflection as a way to improve self-awareness on students regarding their soft skills.

In the study conducted by Brabov (Bobrov et al., 2020), experiences regarding teaching DevOps in the Industry and in the Academia were shared by the authors. The authors argue that DevOps experienced significant success in the industrial sector, but still requires attention in higher education. The au-

thors presented the knowledge structure used over the course which was mainly focused on the Deployment Pipeline. The course used a PBL (Problem-Based Learning) approach and the students worked in groups to solve challenges they received. One of the lessons learned in the study is related to the importance of soft skills in DevOps. The authors argue that soft skills capabilities are a must for future software engineers working expected to work in a DevOps-oriented organization. The authors still argue that by working in groups students get involved in a context where problems may arise, allowing them to learn soft skills to deal with such as problems.

Similar to PBL, there are different active learning methodologies including Challenge-Based Learning (CBL). Santos (Santos et al., 2015b) present a new learning and software development framework that combines Agile methodologies with CBL. CBL[2] is a learning framework developed by educators working with Apple Inc., which has been implemented in a wide variety of educational and corporate settings (Santos et al., 2015b).

Kuusinen (Kuusinen and Albertsen, 2019) investigated how to organize the structure of multidisciplinary and complex architecture courses by designing and executing a Continuous Delivery (CD) and a DevOps oriented course. The authors argue that the main motivation to perform the study was related to inability of the educational system to adapt to the market needs and match the available skills with all existing jobs. The authors proposed a study topics structure mainly focused on the technical aspects related to CD and DevOps. After designing the structure, the course ran over two weeks, with 15 students participating on it. The authors argue the students were particularly happy about learning both practical and academic skills. The biggest complaint was the workload of 125 study hours over two calendar weeks. Moreover, content-wise the students wished that the practical and theoretical days would be mixed. Regarding infrastructure and tools, overhead of keeping up with the industrial state-of-the-art at universities is often too high. Still regarding the toolset used, the authors argue that in the continuous deployment era, universities need to explore feasible ways to keep up with the technological development in a way that allows them to concentrate on fundamentals and underpinning theories that help the students in lifelong learning, but at the same time also keep the employability of a newly graduated in mind.

Buffardi (Buffardi et al., 2017) argue that educational programming assignments have limited scope and rigid design and introduce a *Tech Startup* approach to teaching Agile software development in a software engineering course leveraging collaboration and entrepreneurship aspects. According to the authors, such experience should help students to develop soft skills, specially communication, once students will be contact with non-technical stakeholders as well as class colleagues.

Melegati (Melegati et al., 2019) study investigated how software engineering students learned startup development methodologies and the challenges and benefits of such learning process. According to the authors, such learning process allows the development of several soft skills and the importance of business concepts.

5 COURSE PROPOSAL

Based on the problem stated in Subsection 1.1 and the lessons learned from the previous studies evaluated in Section 4, we propose a course structure, which uses DevOps, Scrum and Challenge Based Learning (CBL) approaches to help with the development of soft skills and technical skills, which are in demand in the information technology market as presented in Figure 1.

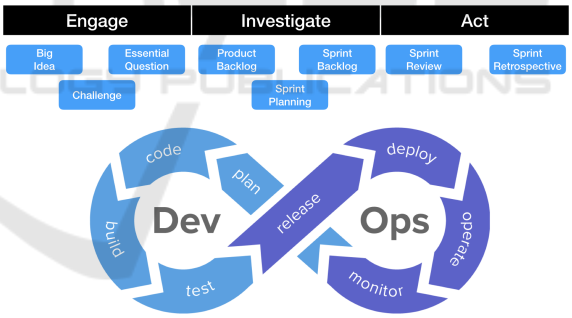

Figure 1: Course Structure Proposal.

The usage of DevOps as an integrative approach is especially important in the computer science and software engineering context, once it allows the joint of several areas, such as Cloud Computing, Networks, Operational Systems, Database Management and Software Development, among others. The usage of Scrum is proposed because, along with DevOps, it fosters the communication and collaboration among team members, over extensive documentation and processes. The usage of Challenge Based Learning is recommended based on the characteristics of this framework, which provides a powerful learning and problem-solving approach on real problems. The purpose of the course structure described in the fol-

[2]Details and samples about CBL are also available at http://www.challengebasedlearning.org

lowing section is to provide a high-level guideline to be used by institutions who are willing to improve students' soft skills and technical skills.

5.1 Course Structure

The aim of our proposal is to provide a high-level framework using the lessons learned from the studies previously evaluated and the industry emerging practices. Before we provide details about the proposed course structure, it is important to emphasize that some characteristics of the structure proposed, such as duration, could be adapted according to each institution needs and the number of students attending to the course. Table 2 provides a macro description of the course structure.

Table 2: Course Structure.

Phase	Description	Duration
Sprint 1	CBL: Macro Challenge - Engage/Investigate	8 hours
Sprint 1	CBL: Student Reflection	30 minutes
Sprint 1	Peer Feedback Session (Professor)	30 minutes
Sprint 1	Peer Feedback Session (Colleague)	30 minutes
Sprint 2	CBL: Macro Challenge - Act - DevOps Fundamentals	8 hours
Sprint 2	CBL: Student Reflection	30 minutes
Sprint 2	Peer Feedback Session (Professor)	30 minutes
Sprint 2	Peer Feedback Session (Colleague)	30 minutes
Sprint 3	CBL: Macro Challenge - Act - Building Deploy Environments	8 hours
Sprint 3	CBL: Student Reflection	30 minutes
Sprint 3	Peer Feedback Session (Professor)	30 minutes
Sprint 3	Peer Feedback Session (Colleague)	30 minutes
Sprint 4	CBL: Macro Challenge – Act - Building the CI/CD Pipeline	8 hours
Sprint 4	CBL: Student Reflection	30 minutes
Sprint 4	Peer Feedback Session (Professor)	30 minutes
Sprint 4	Peer Feedback Session (Colleague)	30 minutes
Sprint 5	CBL: Macro Challenge - Act - Adding Test Automation to the Pipeline	8 hours
Sprint 5	CBL: Student Reflection	30 minutes
Sprint 5	Peer Feedback Session (Professor)	30 minutes
Sprint 5	Peer Feedback Session (Colleague)	30 minutes

At the beginning of the course, students should be grouped in teams and each team should define a real-world problem to be solved with a software based solution. In order to ensure that different areas of computer science or software engineering will be explored over the course, some requisites should be defined by the professors at the beginning of the course. In every *sprint*, the students will be given an introductory class about a specific DevOps subject and, following the CBL framework, will work together to plan and deliver what has been planned. Elements of the DevOps subject discussed at the beginning of each sprint are expected to be implemented by each team, as part of the software solution to be delivered at the end of the sprint.

At the end of each sprint, the students should perform a CBL reflection, individually, in order to reflect about their own learnings and improvement opportunities, regarding the soft skills they have practiced over the sprint.

Hattie (Hattie and Timperley, 2007) argue that feedback can be a great approach to support students' learning and achievements. In order to provide feedback to each one of the students regarding their soft skills and improvement opportunities, professors are encouraged to schedule a 30 minutes meeting after the end of each sprint with each student. For several reasons, face-to-face feedbacks are recommended. Such approach could be challenging to the professors based on the number of students. However, we encourage the engagement of more than one professor in the course, in order to ensure the feedbacks will be provided with enough details to allow improvements on each individual's soft skills. As this course is a joint of several areas, the participation of other professors in the soft skills evaluation process is highly recommended.

Sadler (Sadler, 2010) argues that peer assessment is also the most natural way to provide tacit and explicit knowledge transfer. Based on Sadler's findings, we also consider having a peer assessment approach, where, at the end of each sprint, each student should be assessed by their peers, which includes professors and other students, on the soft skills agreed to be evaluated over the course.

Regarding the soft skills to be developed, Table 3 provides a list of important soft skills we identified in the previous studies evaluated as well as their characteristics.

Additionally, Table 4 describes the main rituals which could be used to evaluate the students and each one of the skills to be developed.

It is important to emphasize that the skills selected to be evaluated in our course proposal could

Table 3: Soft skills evaluation.

Soft Skill	Description
Team work	Cooperative, gets along with others, agreeable, supportive, helpful, collaborative
Communication	Oral speaking capability, written, presenting, listening, clear and structured speech
Problem Solving	Ability to frame a problem and define a structured course of action
Decision Making	Ability to take decisions in a timely matter, rationally, using the available information

Table 4: Rituals.

Soft Skill	Ritual
Team work	Sprint Planning, Backlog Grooming, Tasks Execution, CBL Engage/Investigate/Act
Communication (verbal)	Daily Scrum, Sprint Retrospective, Sprint Planning, Backlog Grooming, CBL Reflection
Communication (written)	User Stories, Behavior Driven Development Specifications, Knowledge Transfer, CBL Reflection
Problem Solving	Tasks Execution, Researching Solutions, CBL Engage/Investigate/Act
Decision Making	Sprint Planning, Backlog Grooming, Tasks Execution, Researching Solutions, CBL Engage/Investigate/Act

be adapted according to the institutions and industry needs.

5.2 Course Evaluation

In order to evaluate the effectiveness of the course structure proposed, questionnaires should be applied at the beginning and after the conclusion of the course. The comparison between both questionnaires' responses, along with feedbacks provided by peers and professors over the course, will provide means to evaluate the progress and development of the students at the end of the course, as well, should provide valuable insights to promote improvements to the course structure.

6 CONCLUSION AND FUTURE WORKS

This study proposed a framework that takes advantage of cultural aspects from emerging software development and active learning practices such as DevOps, Scrum and CBL to support the development of Software Engineering and Computer Science students in higher education courses. The development of this framework was motivated by the need of a more prescriptive guideline to develop such skills and due to the market needs (Ahmed et al., 2012; Carter, 2011). The usage of Challenge Based Learning elements as part of this framework was also motivated by some of the studies we analyzed (Thurner et al., 2016; Santos et al., 2015b) . This study is still a work in progress, therefore, a more extensive research on the literature could bring additional insights to complement our proposal. As a future work, we aim to apply the solution proposed on an experimental course in order to validate the outcomes and propose changes based on the lessons learned.

REFERENCES

Ahmad, K. and Gestwicki, P. (2013). Studio-based learning and app inventor for android in an introductory cs course for non-majors. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 287–292, New York, NY, USA. ACM.

Ahmed, F., Capretz, L. F., and Campbell, P. (2012). Evaluating the demand for soft skills in software development. *IT Professional*, 14(1):44–49.

Andersson, C. and Logofatu, D. (2018). Using cultural heterogeneity to improve soft skills in engineering and computer science education. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 191–195.

Bastarrica, M. C., Perovich, D., and Samary, M. M. (2017). What can students get from a software engineering capstone course? In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 137–145.

Bobrov, E., Bucchiarone, A., Capozucca, A., Guelfi, N., Mazzara, M., and Masyagin, S. (2020). Teaching devops in academia and industry: reflections and vision. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 1–14. Springer International Publishing.

Buffardi, K., Robb, C., and Rahn, D. (2017). Learning agile with tech startup software engineering projects. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*,

ITiCSE '17, page 28–33, New York, NY, USA. Association for Computing Machinery.

Carter, L. (2011). Ideas for adding soft skills education to service learning and capstone courses for computer science students. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, page 517–522, New York, NY, USA. Association for Computing Machinery.

Davis, J. and Daniels, R. (2016). *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media.

Fetaji, M. and Fetaji, B. (2009). Analyses of mobile learning software solution in education using the task based learning approach. In *Information Technology Interfaces, 2009. ITI '09. Proc. of the ITI 2009 31st Int. Conf. on*, pages 373–378.

Gestwicki, P. and Ahmad, K. (2011). App Inventor for Android with Studio-based Learning. *Journal of Computing Sciences in Colleges*, 27(1):55–63.

Hattie, J. and Timperley, H. (2007). The power of feedback. *Review of Educational Research*, 77(1):81–112.

Hazzan, O. and Har-Shai, G. (2014). Teaching and learning computer science soft skills using soft skills: the students' perspective. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 567–572.

Hussaini, S. W. (2014). Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 178–183.

Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, New York, NY, USA.

Kerzazi, N. and Adams, B. (2016). Who needs release and devops engineers, and why? In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, pages 77–83.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33.

Kuusinen, K. and Albertsen, S. (2019). Industry-academy collaboration in teaching devops and continuous delivery to software engineering students: Towards improved industrial relevance in higher education. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 23–27.

Matturro, G., Raschetti, F., and Fontán, C. (2015). Soft skills in software development teams: A survey of the points of view of team leaders and team members. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 101–104.

Melegati, J., Chanin, R., Wang, X., Sales, A., and Prikladnicki, R. (2019). Perceived benefits and challenges of learning startup methodologies for software engineering students. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 204–210, New York, NY, USA. Association for Computing Machinery.

Nichols, M., Cator, K., and Torres, M. (2016). *Challenge Based Learning Guide*. Digital Promise, Redwood City, CA, USA.

on Cybersecurity Education, J. T. F. (2018). *Cybersecurity Curricula 2017: Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity*. Association for Computing Machinery, New York, NY, USA.

on Information Technology Curricula, T. G. (2017). *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. Association for Computing Machinery, New York, NY, USA.

Sadler, D. R. (2010). Beyond feedback: Developing student capability in complex appraisal. *Assessment & Evaluation in Higher Education*, 35:535–550.

Santos, A., Sales, A., Fernandes, P., and Nichols, M. (2015a). Combining Challenge-Based Learning and Scrum Framework for Mobile Application Development. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'15)*, pages 189–194, Vilnius, Lithuania.

Santos, A. R., Sales, A., Fernandes, P., and Nichols, M. (2015b). Combining challenge-based learning and scrum framework for mobile application development. In *Proceedings of the 2015 ACM conference on innovation and technology in computer science education*, pages 189–194.

Sedelmaier, Y. and Landes, D. (2014). Software engineering body of skills (swebos). In *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 395–401.

Swartout, P. (2012). *Continuous Delivery and DevOps: A Quickstart Guide*. Packt Publishing.

Thurner, V., Schlierkamp, K., Böttcher, A., and Zehetmeier, D. (2016). Integrated development of technical and base competencies: Fostering reflection skills in software engineers to be. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 340–348.

Valentin, E., Carvalho, J. R. H., and Barreto, R. (2015). Rapid improvement of students' soft-skills based on an agile-process approach. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.