Teaching Agile Software Engineering Practices Using Scrum and a Low-Code Development Platform – A Case Study

José Carlos Metrôlho[1,2], Fernando Reinaldo Ribeiro[1,2], Pedro Passão[2]
[1]R&D Unit in Digital Services, Applications and Content
[2]Polytechnic Institute of Castelo Branco
Castelo Branco, Portugal
e-mail: metrolho@ipcb.pt, e-mail: fribeiro@ipcb.pt, e-mail: pedropassao@ipcb.pt

Abstract— **Following the recent trends in software engineering regarding the growing adoption of agile methodologies and low-code development platforms, and considering the results of surveys, we carried out on students, alumni and some IT companies, we adapted the software engineering teaching of a computer engineering course to the needs and new trends of the IT industry. The Scrum methodology and the OutSystems low-code development platform were used in a project-based learning approach for teaching agile software engineering practices. This approach was complemented with the presentation and discussion of several topics during the theoretical classes, lectures given by professionals from IT companies and study visits to an IT company that uses agile methodologies and low-code platforms. This approach aims to enhance the technical skills, namely development skills on a widely used low-code platform and other software engineering skills, but also to reinforce some non-technical skills of students like teamwork and communication, today highly valued by IT companies. The first results are quite positive.**

Keywords- agile methodologies; education; Low-code platforms; software engineering; Scrum; teaching.

I. INTRODUCTION

Several approaches have been used for teaching software engineering. The way they propose to do it differs. However, regardless of the proposed approach, there are some aspects that already seem to be well accepted and that seem to be a common trend for several approaches: there is an effort to make the teaching of software engineering as close as possible to what is done in IT companies; Most strategies try to provide students with practical experience in a software engineering project using methodologies and tools also used in IT companies; and there is a growing concern on empowering students with the non-technical skills required in a software project. To achieve this, it is important to be aware of the needs and trends of the market. It is important to understand how the main concepts of the software engineering subject are assimilated by the students and understand the point of view of the companies which employ and develop activities in this area.

Following the recent trends in software engineering, with regard to the growing adoption of agile methodologies and low-code development platforms, and considering the results of surveys carried out in some IT companies [1], we made some changes in the teaching of the software engineering subject. In this paper, we describe an experience in teaching software engineering. An agile development methodology and a low-code development platform were used in a project-based learning approach. This approach aims to enhance the technical and non-technical skills of students, today highly valued by IT companies, without, of course, neglecting other methodologies and topics related to software engineering.

The remainder of this paper will be as follows: Section II presents a brief review of related work; Section III presents a background about agile development and low-code development platforms; in Section IV, we present an overview of our methodology for teaching undergraduate software engineering using Scrum and a low-code development platform; Section V presents some lessons learned and challenges faced and finally, in Section VI we present some conclusions and we outline the future work.

II. RELATED WORK

Several approaches and strategies have been followed to provide students with the best training in software engineering. Some of them are more theoretical, more focused on the study of theory, concepts, methods and methodologies, while others are more practical, fostering practical experimentation to students, and often carried out in collaboration with companies. Some are more traditional, in the sense that they privilege traditionally used practices, others are more avant-garde and encourage contact with the most innovative practices and new market trends. All of them aim to give students the appropriate knowledge and skills for their professional activity in software engineering. However, the way they propose to do it differs.

Emulating the workplace using distributed software development projects, involving various courses or institutions, is an approach proposed by several authors (e.g., [2][3]). The Distributed and Outsourced Software Engineering course [2] proposed teaching software engineering using globally distributed projects. The projects were developed in collaboration with eleven universities in ten different countries providing students with the experience of working with different cultures, native languages and time zones. This approach also helped to alert students to the importance of understanding 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

 ISBN: 978-1-61208-827-3

committed to their peers. In [3], students work on real distributed open-source projects as full members of software development teams. Students use the same software development processes as regular team members and are provided with explicit mentorship from mentors from each project. With this approach students integrate and apply the skills they have learned in their courses and they develop and improve their technical communication skills in a real development setting.

The use of simulations and gamification to provide students with a variety of experiences that would not be possible in an academic environment, is an alternative proposed by other authors (e.g.,[4]–[6]). Usually, these approaches propose to gamify some phases of the software life cycle and some tasks associated to each of them. The goal is to increase the user's engagement, motivation and performance when carrying out specific tasks. However, these approaches also have some disadvantages. After two periods of teaching using Scrum with gamification to learn and train agile principles, Schäfer [5] identified some lessons learned. Gamification is motivating and helps to bring together participants with different experiences in project teams.

Several project-oriented approaches have been proposed in several software engineering training programmes (e.g., [7]–[10]. A project-based learning experience based on the formation of small heterogeneous teams was presented in [7]. Through a strategy of role rotation and documentation transfer, all students perform different tasks and face different challenges throughout the project. This is the case they decided not to include any external stakeholder. In [8], software engineering concepts are taught using the Scrum framework in real life projects. The requirements are discussed with external customers during a kick-off meeting. During the project, students work together as self-organized teams. They chose a project management and team coordination process and they are only asked to use some core tools that are needed to monitor the projects.

From a different perspective, the teaching of software engineering has been adapting to new developments and trends namely the agile methodologies. This topic has deserved the attention of many authors who have published several studies that address this subject. Usually, teaching agile methodologies has focused on teaching a specific method like Scrum (e.g., [11]–[13]) or XP (e.g., [14] [15]). A project-based learning approach using the Scrum framework in real life projects is presented in [11]. The module starts with a kick-off where external stakeholders introduce their topics, students apply for their preferred topics and the supervisors define the teams of 5–7 persons. After 3 weeks, the students must provide a project proposal which has to be presented and defended face-to-face against customer comments. The project proposal requires the definition of a clear aim of the project, as well as a backlog of requirements with an estimation and prioritization of the relevant user stories. The projects are run in sprints, with a final presentation and the hand-over of the results. An outline of the literature related to Scrum in software engineering courses [16] shows that providing students with practical experience is of vital importance when teaching Scrum in software engineering courses. It also states that most Scrum courses require students to work in teams in order to develop a non-trivial software project or practice simulation games. A study on the impact of using agile methods in software engineering education [17] concluded that using Agile practices would positively influence the teaching process and that they could stimulate communication, good relationships among students, active team participation, and motivation for present and future learning.

In fact, several approaches have been used for teaching software engineering. However, and as mentioned in [18], it is not clear which should be the best approach do follow because there are different perspectives on the different proposed approaches. Some of them propose to emulate the workplace using distributed projects or using simulations and games to simulate different scenarios. Others propose project-based learning where students can train the various stages of project development, following different methodologies, and develop non-technical skills. However, regardless of the approach followed, some aspects seem to already be well accepted and seem to be a common trend for several approaches. There is an attempt to bring teaching closer to business reality. Many of these strategies include providing students with hands-on experience in a software engineering project using methodologies and tools that are also used in IT companies. At the same time, many of these strategies have also focused on empowering students with the non-technical skills required in a software project. It is also true that more traditional approaches, in which students take on a more passive role and that place a higher priority on teaching students to follow instructions and rules, do not produce the intended results. Most current approaches, for teaching software engineering, try to help students develop their own ideas and strategies. They promote project-based learning and they try to engage students in the problem definition, design process and system thinking.

III. AGILE DEVELOPMENT AND LOW-CODE PLATFORMS

The growing spread of agile software development methodologies, the increasing attention they have attracted and their growing adoption by IT companies, seem to ensure that they will play an important role in the future. Some recent surveys demonstrate the importance and the high level of adoptions of these methodologies. A survey presented in the 14th annual state of agile report [19] shows that 95% of respondents report that their organizations practice agile development methods. Accelerating software delivery, enhancing ability to manage changing priorities, increasing productivity, and improving business alignment are the top reasons stated for adopting Agile. Scrum and related variants are the most common agile methodologies used by respondents' organizations (referred by 58% of the respondents). Another survey [20], which involved 3300 IT professionals, mentions an even higher percentage, stating that Scrum and related variants are used in 76% of companies. Additionally, some studies have demonstrated

 ISBN: 978-1-61208-827-3

the greater satisfaction of companies and professionals who have adopted these methodologies. For individual professionals, they found that agile development seams to led to greater satisfaction mainly because of collaborative practices and business influences [21]. Another study [22] points out several benefits that were identified by companies that adopted agile methodologies namely: improving project monitoring and tracking, improving interaction and collaboration and fosters sharing knowledge.

Another trend that has been noted is the growing adoption of low-code development platforms by IT companies. The State of Application Development [20] refer that 41% of respondents said their organization was already using a low-code application platform and, a further 10% said they were about to start using one. This growing interest is also corroborated by the Low-Code Development Platform Market [23]. It reports that the global low-code development platform market size is projected to grow from USD 13.2 billion in 2020 to USD 45.5 billion by 2025, at a Compound Annual Growth Rate of 28.1% during the forecast period. The top reported reasons for adopting agile [20] are the ability to manage changing priorities, project visibility, business alignment, delivery speed/time to market and team morale. These reasons are in line with the advantages that are usually associated with the use of low-code development platforms: They comprise many of the same tools functionalities that developers and teams use to design, code, deploy and manage an application portfolio [24]; A significant part of the job can be done through a drag-and-drop interface and although developers may still need to do some coding this is just for specific tasks [25]; They are able to accelerate the delivery of new software and applications, allowing to update and deliver new features in short time periods, they allow build apps for multiple platforms simultaneously, and cross-platform support and data integration capabilities have already been built and can be customized easily [26]. In fact, these platforms have become quite popular and are currently spread across many companies around the world. A report from Forrester [27] evaluated the 13 most significant low-code platforms suppliers and identified Microsoft, OutSystems, Mendix, Kony and Salesforce as leaders.

Another important aspect is that low-code platforms have often been associated with agile development methodologies. The adoption of agile development methodologies, platforms and tools has been a way of improving the ease and speed at which applications can be developed. But, as referred in [28], there is still room for improvement and in particular when it comes to education and training, management commitment and staffing. There is also a need for greater involvement from the wider business, which agile and the use of tools such as low-code both encourage while, at the same time, enhancing developer productivity. The State of Application Development [20] revealed that companies that have adopted low-code have an 8% higher organizational agility score compared to those not using low-code. They also refer that this result seems to be related to the fact that a highly mature agile culture helps organizations maximize the benefits of low-code development platforms by combining the fast decision-making of agile with fast development speeds. However, to maximize agile teams' performance with a low-code platform, there are some aspects that must be followed with particular attention. Some of these aspects are identified in the document Adapting Agile to Build Products with Low-Code: Tips and Tricks [29] and are related to: the difficulty for teams in maintaining a sufficient backlog of user stories ready for development due to the faster development speed; the difficulty of new teams in low code to achieve the necessary quality from the beginning of the process; the significant difference in development velocity between co-dependent teams; the need for a strong product owner who is engaged, empowered and responsive; and the need for collaboration between developers and business analysts from the start of the development cycle, especially for complex user stories.

IV. Overview of our Approach

The software engineering subject is part of the second year of a computer science course (undergraduate course). It is a subject that has a semester load of 30 hours for theoretical classes and 45 hours for laboratory classes. The focus of our approach is to combine theory and practice and ensure that the topics covered remain appropriate to whatever the needs of employers and current trends in the area of software engineering are. A project-based approach is used in practical classes for teaching Software engineering.

The teacher of theoretical classes presents the concepts and methodologies and promotes discussion about them. In these classes, several aspects related to the software development cycle are taught and discussed. Students are provided with an introduction to several software development methodologies namely Waterfall, Extreme Programming, Scrum, Spiral, Rapid Application Development, Rational Unified Process, Feature Driven Development, Behaviour Driven Development, etc. Other topics analysed include quality and metrics in software engineering, requirements analysis, software design, implementation, testing, configuration management, among others. In addition to the presentation and discussion of several topics during the classes, other initiatives are organized and implemented, namely lectures given by professionals from software development companies and study visits to software development companies. These initiatives provide students with the contact and interaction with real software engineering projects with real stakeholders. They are carried out in the final weeks of the semester, so by that time the students have already acquired significant knowledge that will then allow them to get the most out of them.

In practical classes, students acquire some practice of software engineering through the process management, specification, design, implementation and validation of a

software application, as a project for teams. Scrum is the adopted agile software development methodology. The teacher has experience with Agile methodologies and holds a professional certification in the adopted low-code platform. He was able to provide support during the initial learning phase of application development on the low-code platform, but also to support the various teams of students during the scrum sprints of development of their projects. The teacher acts as a product owner. Each team member has a specific role (e.g., Scrum Master, developer, etc.). Each team develops a different project. We have used Scrum because several employers of companies in the software development area, with whom we have had contact, use agile methodologies [1], namely Scrum and because our graduates have told us that it is clearly one of the methodologies they use most [30]. In addition, we also aim to improve students' teamwork, and this methodology is one that fits well with this goal. These skills of teamwork have been highly valued by employers and therefore they deserve to be worked on in this subject as well. We have been using Scrum in practical classes for years and the recent survey [1] only reinforced it and that is why we continue to use it.

In past editions of this subject, the projects were related to the development of games (using Unity) or even to continue work started earlier in other subjects of the course (developed in java). The new trends and the feedback we obtained in a survey [1] led us to, in the previous academic year, choose to introduce the development of projects in practical classes using a low-code platform. This is an area of great demand by our students' employers, so with this approach we also wanted to provide new skills at the level of coding competence. In other words, the survey we carried out [1] was clear as to the importance of coding skills, but also of other aspects such as requirements analysis and development methodologies. So, on the one hand, with this new approach we give students new coding skills using one of these development platforms widely used by several recruiting companies in the software development area. On the other hand, due to the characteristics of these low-code platforms, it allows us to emphasize and work with students on other different and important aspects of the development of software projects, such as requirements analysis, project design, project management project, development methodology, quality assurance, testing, planning, etc. When students complete the entire course, they obviously have much more comprehensive skills because in other subjects they learn to program in various other languages and paradigms (Java, PHP, Html, SQL, etc.). This subject of software engineering is not a programming subject but a subject in which the coding stage is only part of a whole. The whole concerns the cycle of software development and therefore it is also important to address and emphasize what is not so addressed in other subjects of the course. Namely the importance of development methodologies, planning, requirements analysis, software quality, testing, maintenance, documentation, etc. Considering this reality, it seemed to us that the use in this subject of a low-code platform in practical classes could bring advantages, and after having implemented it, we remain convinced.

To keep students motivated, the themes and objectives of the projects could be defined by the teams of students or alternatively by carrying out themes proposed by the teacher of the practical classes. With this new approach, projects include the development of web and mobile applications using a low-code platform. In practical classes Scrum is the development process used. The teacher of the practical classes monitors weekly the evolution of each of the projects. This monitoring allows for the assigning of grades between teams but also being able to differentiate the grades of each element of a team. Monitoring is weekly, during contact classes with students. The student teams, in addition to the weekly class time, also work outside of classes. Tasks are all registered in Trello, allowing the teacher and the whole team to have a permanent record of the progress of the respective project. Trello is used for project management and to track progress on tasks.

During the semester, the project evolves over several sprints (of two weeks), in which the teacher (acting as product owner) evaluates with the respective team what was achieved in the previous sprint and what should be the sprint backlog of the sprint that follows.

The final grade of the subject, in terms of the practical part, results from an intermediate evaluation of each student based on the work presented in the middle of the semester and from a second evaluation made at the end of the semester. In these two stages, a demo is made by each team, resulting in grades and feedback given by teachers to the various teams. The grades result from the application of parameters related to various aspects of the various phases of the project's development and the Scrum methodology. Some of the parameters are: Requirements analysis, software development process (e.g. roles, artefacts, timings, hits and misses), task scheduling, modelling (e.g. user stories, storyboards), implemented features, conclusions and future work, user interface, documentation, and final presentation and discussion. In the past academic year, due to Covid-19, classes were provided using video conferencing for teacher-student or teacher-team interaction. The fact that low-code platforms provide several online teaching materials (webinars, tutorials, examples, etc.) was also useful to successfully overcome the limitations mentioned above. This complementary material helped all teams to quickly and timely assimilate necessary knowledge about development in the adopted platform, in order to implement their projects.

We also noticed that the learning and adaptation to the use of low-code platform by students was overall very good. The developed projects resulted in applications with practically all user stories implemented and validated. The students in their final reports addressed aspects about the various stages and timings of the work developed, as far as software engineering is concerned. Some of the projects

 ISBN: 978-1-61208-827-3

resulted in web applications with good user interfaces. Throughout the semester, we verified a high activity and motivation by practically all students. All projects resulted in functional applications, some of which reached quality close to the maximum score.

The low-code platform that we used in practical classes was the OutSystems. We choose this platform because it is a platform widely used by software development companies in Portugal and because we have had a collaboration protocol with that company for several years, under which we have accessible software licenses. Another important fact in the choice is that this platform can easily coexist with agile methodologies such as Scrum [31] and it is one of the leaders in the low-code market [27].

V. LESSONS LEARNED AND CHALLENGES FACED

Even considering the entropy caused by the effects of Covid-19 (videoconference classes and student/team meetings also via videoconference), in the end it resulted in good results from both the theoretical and practical parts. The inclusion of the low-code platform in practical classes, allowed students to develop web applications, and to develop new skills in one of the low-code platforms widely used in software development companies. Additionally, and very importantly, this approach allowed us to meet the findings of the survey that was carried out on IT companies [1]. It allows to strengthening students with other skills related to software engineering like development methodologies, requirements analysis, project management, schedules, testing, etc. As mentioned before, this subject is not focused on coding, for that there are several others in the course where several programming skills are covered. We also believe that this approach contributes to the improvement of the non-technical skills of students, namely teamwork and communication.

It is also important to consider that Low-code platforms have some advantages and are suitable in the context of this subject of software engineering. However, they do not replace the need for the knowledge covered in other subjects to prepare our students for a wider range of knowledge, about other approaches and technologies that are also very useful and often necessary.

VI. CONCLUSION AND FUTURE WORK

After listening to several stakeholders with the aim of keeping the themes and methodologies taught in the subject of Software engineering updated, we share in this paper an update done recently. This update consisted in making the projects developed in the practical classes using Scrum and a low-code platform. This decision was to reinforce students development skills (on a low-code platform currently highly used in the labour market) and lead students to a greater focus on other software engineering skills (teamwork, communication, requirements, software quality, schedules, documentation, among others). The results achieved were positive, and the feedback from the students was very rewarding. In a survey conducted at the end of the semester, on a scale of 0 to 6, students rated the overall satisfaction in relation to the subject with 5.4.

In the future, we will continue to be attentive to stakeholder feedback, to keep materials and methodologies updated in order to prepare students as best as possible and close to what is followed in the software development industry.

REFERENCES

[1] J. C. Metrôlho and F. R. Ribeiro, “Holistic Analysis of the Effectiveness of a Software engineering Teaching Approach,” *Int. J. Adv. Softw.*, vol. 12, no. 1 & 2, pp. 46–55, 2019.

[2] 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, doi: 10.1145/1984665.1984673.

[3] 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, doi: 10.1145/2591062.2591160.

[4] 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, doi: 10.1145/2896958.2896965.

[5] U. Schäfer, “Training scrum with gamification: Lessons learned after two teaching periods,” in *2017 IEEE Global Engineering Education Conference (EDUCON)*, 2017, pp. 754–761, doi: 10.1109/EDUCON.2017.7942932.

[6] W. Ren, S. Barrett, and S. Das, “Toward Gamification to Software engineering and Contribution of Software Engineer,” in *Proceedings of the 2020 4th International Conference on Management Engineering, Software engineering and Service Sciences*, 2020, pp. 1–5, doi: 10.1145/3380625.3380628.

[7] B. Pérez and Á. L. Rubio, “A Project-Based Learning Approach for Enhancing Learning Skills and Motivation in Software engineering,” in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 309–315, doi: 10.1145/3328778.3366891.

[8] 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, doi: 10.1109/EDUCON.2018.8363442.

[9] M. L. Fioravanti *et al.*, “Integrating Project Based Learning and Project Management for Software engineering Teaching: An Experience Report,” in

Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 2018, pp. 806–811, doi: 10.1145/3159450.3159599.

[10] M. Gordenko and E. Beresneva, "A project-based learning approach to teaching software engineering through group dynamics and professional communication," in *Actual Problems of System and Software engineering. Proceedings of the 6th International Conference Actual Problems of System and Software engineering*, 2019, pp. 278-288.

[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, doi: 10.1109/EDUCON.2018.8363442.

[12] G. Wedemann, "Scrum as a Method of Teaching Software Architecture," in *Proceedings of the 3rd European Conference of Software engineering Education*, 2018, pp. 108–112, doi: 10.1145/3209087.3209096.

[13] I. Bosnić, F. Ciccozzi, I. Čavrak, E. Di Nitto, J. Feljan, and R. Mirandola, "Introducing SCRUM into a Distributed Software Development Course," 2015, doi: 10.1145/2797433.2797469.

[14] J. J. Chen and M. M. Wu, "Integrating extreme programming with software engineering education," in *38th International Convention on Information and Communication Technology, Electronics and Microelectronics*, 2015, pp. 577–582, doi: 10.1109/MIPRO.2015.7160338.

[15] B. S. Akpolat and W. Slany, "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification," in *27th Conference on Software engineering Education and Training*, 2014, pp. 149–153, doi: 10.1109/CSEET.2014.6816792.

[16] V. Mahnic, "Scrum in software engineering courses: An outline of the literature," *Glob. J. Eng. Educ.*, vol. 17, no. 2, pp. 77–83, 2015.

[17] S. Al-Ratrout, "Impact of using Agile Methods in Software engineering Education: A Case Study," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 1986–1991, doi: 10.1109/CoDIT.2019.8820377.

[18] 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, doi: 10.1109/MS.2017.12.

[19] Digital.ai, "14th annual state of agile report," 2020. https://stateofagile.com/ (accessed Aug. 20, 2020).

[20] OutSystems, "State of Application Development Report," 2019.

[21] M. Kropp, A. Meier, C. Anslow, and R. Biddle, "Satisfaction, Practices, and Influences in Agile Software Development," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software engineering*, 2018, pp. 112–121, doi: 10.1145/3210459.3210470.

[22] F. Kamei, G. Pinto, B. Cartaxo, and A. Vasconcelos, "On the Benefits/Limitations of Agile Software Development: An Interview Study with Brazilian Companies," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software engineering*, 2017, pp. 154–159, doi: 10.1145/3084226.3084278.

[23] Marqual IT Solutions Pvt. Ltd (KBV Research), "Global Low-Code Development Platform Market By Component By Application By Deployment Type By End User By Region, Industry Analysis and Forecast, 2020 - 2026," Report, 2020. [Online]. Available: https://www.kbvresearch.com/low-code-development-platform-market/.

[24] OutSystems, "Low-Code Development Platforms," 2019. https://www.outsystems.com/low-code-platforms/ (accessed Jul. 30, 2020).

[25] C. Boulton, "What is low-code development? A Lego-like approach to building software," *CIO (13284045)*, 2019. http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=134645048&site=eds-live (accessed Aug. 07, 2020).

[26] J. Idle, "Low-Code rapid application development - So, what's it all about?," *Platinum Business Magazine*, pp. 52–53, 2016.

[27] J. R. Rymer and R. Koplowitz, "The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2019," 2019.

[28] I. Media, "Agile is as agile does. Understanding the role of agile development and low-code solutions in the delivery of digital transformation." Incisive Media, 2018.

[29] T. Huff, "Adapting Agile to Build Products with Low-Code: Tips and Tricks," 2019. https://www.outsystems.com/blog/posts/adapting-agile-to-low-code/ (accessed Jul. 28, 2020).

[30] J. Metrôlho and F. Ribeiro, "Software engineering Education: Sharing an approach, experiences, survey and lessons learned," in *Thirteenth International Conference on Software engineering Advances*, 2018, pp. 79–84.

[31] T. Huff, "Agile and Scrum: Understanding the Differences," 2019. https://www.outsystems.com/blog/posts/agile-and-scrum/ (accessed Jul. 12, 2020).

 ISBN: 978-1-61208-827-3