Experiencing the Sheffield Team Software Project: *A project-based learning approach to teaching Agile*

Olakunle Olayinka
Department of Computer Science
The University of Sheffield
Sheffield, United Kingdom
o.olayinka@sheffield.ac.uk

Mike Stannett
Department of Computer Science
The University of Sheffield
Sheffield, United Kingdom
m.stannett@sheffield.ac.uk

Abstract— **Graduates of computer science and software engineering degrees are often expected by employers to possess various technical skills as well as competencies in project management, testing, teamwork, and other soft skills. Extant literature has identified that these competencies are often not addressed by traditional teaching approaches such as lectures and labs. In this paper, we present a project-based learning approach to teaching agile software development where students work in multicultural teams to develop software for clients. This approach to teaching software development addresses some of the competencies required by employers, and the feedback from students, clients, and tutors are discussed and analysed critically.**

Keywords— Project-based learning, Agile, Teamwork, Software development, Scrum framework

I. Introduction

The ability to work in teams is recognised as a crucial asset both in industry and in academia, and accreditation bodies regard it as an essential component of any meaningful degree programme. Today, software engineering graduates are expected to start on their first job as software engineers or developers and actively contribute to the team [1]. However, university group projects often fail to adequately simulate complex organisational structures and collaboration that make students work-ready [2].

Furthermore, most university curricula focus heavily on the theoretical underpinnings [1][3] of the subject and although these are essential, other relevant practical skills that a student needs to apply in a work environment are equally important [4]. Today, most software projects in organisations are complex, requiring an average team of eight [5] to tackle, and in larger organisations development teams often have thousands of developers [6] working on several projects that interface with each other. For an employee to be productive in such an environment, the need for communication and collaboration is extremely important [7], but how well does the software engineering curriculum today prepare them for such an environment?

To make students work-ready, project-based learning is a pedagogical approach that has been used over the years in disciplines such as creative design [8] and game development [9] to teach the subject matter while also preparing students for life after university. In recent times, this approach has also been applied to computer science and software engineering modules, to teach a combination of technical and non-technical concepts. Also, given the increased demand for companies to review GitHub profiles of prospective graduate developers, universities have increasingly encouraged their students to build up a portfolio of projects by applying this approach to teaching some final year modules [10]. However, some academics have argued that this approach to teaching has limitations and might be inappropriate for teaching subjects where several deliverables and feedback are required [11][12] but others have applied this approach with great feedback from students [13][14].

Over the last two decades, there has been an increase in the adoption of Agile software methodologies by companies of various sizes [15]. The benefit of deploying Agile practices in an organisation cannot be overemphasised as it encourages quicker development and regular dialogue between the business and development teams [16]. As a result, computer science and software engineering degrees now consider including agile software development modules in their curriculum for both undergraduate and postgraduate degrees [17][16]. However, some of these modules are heavily theoretical and lumped with teaching other concepts such as learning a new programming language, understanding coding styles and several others without a strong emphasis on teaching Agile on its own [1].

Some researchers have explored the use of problem-based learning [3], active learning [18], and other approaches to teaching agile software development practises to both undergraduate and postgraduate students with varying results. In our work, we take a project-based learning approach to teaching agile software development in an MSc module. The ideas, though unique, leverage learnings from Genesys Solutions, a Sheffield University module where fourth-year students run their own software house [19], and Software Hut [20]. In this paper, we discuss the goals of the module, as well as the hurdles that had to be overcome while teaching it and useful lessons learned.

Outline. Section II reviews existing literature and related work on the use of project-based learning to teach software engineering and teamwork. Section III gives the background to the module and the module structure; section IV discusses our methods and results, together with evaluations of the modules from various stakeholders' perspectives. Section V preincludessents ideas for improvement, while section VI concludes the paper.

II. Related Work

Research has shown that students learn faster and better when taking an active position in the learning process as opposed to passively listening [21][22]. Of the several approaches to implement active learning, project-based learning has been gaining ground in recent years [23], particularly in engineering disciplines where abstract concepts are taught and theoretical information, without adequate practical experience, is insufficient for the students to understand the subject [12].

Attesting to the challenge faced when teaching software engineering in a traditional classroom setting to undergraduate students, and the need for a more involving approach, [24] developed a "mock software company" for undergraduate students to learn software engineering principles and teamwork. In this module, the students worked for 10 weeks in teams which were subdivided into management and software development teams to develop an Algorithm Animation project. Each year the module is run, the students inherit the project code and artefacts from the previous year, thus providing an opportunity to experience challenges such as refactoring other people's code.

In a similar vein, [25] created a capstone course where students created a "mock" consulting company to work with a corporate sponsor on a project. A large part of the project was to develop an actual product that the corporate clients could use, and throughout the two-semester course the student teams met bi-weekly with corporate sponsors and faculty members. At the end of the course, one corporate healthcare sponsor reported that one team saved them over $100,000 in consulting services. [26] also created a four-semester capstone project called the 'Software Enterprise' that relied on a combination of lectures, problem-centered learning, and project work to expose students to industry practice.

In order to foster industry and academia collaboration and enable students to learn from industry partners acting as more than simply clients, [27] developed an industry-academia team-teaching capstone course for software engineering undergraduate students that ran over two semesters and was coordinated by two instructors - one from academia and the other from industry. Likewise, [28] developed a software engineering course that provided a real-world enterprise simulation that used agile for students to learn the competencies necessary for a software engineering job. In this course, 25-26 students formed a virtual company and competed on developing a specific large-scale software project.

Similarly, [10] adopted a project based learning approach to teaching a software engineering project course to a large cohort of 156 students. The course aimed not only to provide students with relevant theoretical knowledge but also to improve students' employability skills by ensuring the projects were industry oriented. The course was designed to cover agile project delivery, requirements engineering and software architecture and upon execution of this course, the feedback from the students and industry partners was very positive and resulted in an improved satisfaction rate.

However, one main challenge faced during the execution of the course was that the contact hours required for the module were too much for the students. This challenge was also identified by [12], however in their module, it was deemed a mere perception by the students and not the actual reality. Nonetheless this emphasises the need to consider student workload and manageability, when designing modules that use project based learning.

Furthermore, while some academics have taken to project based learning as an approach to tackle the issue of teaching team work and employability skills to their students, [14] argues that many of these competencies are not addressed by regular lectures and project based learning. They accordingly designed a module using the scrum framework to teach software engineering competencies. Their module incorporated real customers from the industry who presented their projects at a kick off meeting; students were allowed to select projects and form their own teams (of 5-7 students). The module has run for more than 8 years and the feedback from all stakeholders has been quite good; particularly with students and customers commenting on the technical and non-technical competencies gained.

To expose the student to true agile projects and practices, [29] developed a "tech start-up model" approach to teaching agile software development to software engineers by leveraging collaboration with an entrepreneurship class. Through a combination of agile and lean start-up practices their module provided an interdisciplinary mix, where the students were able to learn from each other, and enabled the software engineering students to heavily involve users and business people (entrepreneurship students) in their development process.

Because most students enrolling in their software engineering class did not have significant teamwork or project experience, [13] developed a new software engineering module to meet these needs. The module consisted of "local" groups of students from two different universities, one in the US and the other in Germany, so as to simulate a small software company with teams of 4-6 students working in a virtual environment. While this module design did facilitate collaboration and created a multi-cultural environment, it appears to be a complex model to replicate.

Furthermore, as a way to address both technical skill and employability skills required by engineering graduates, [30] developed a course to simulate industry projects. In this course, students worked with industry partners to develop solutions to contemporary, real-world problems such that software developed by students was regularly incorporated into the code base of industry partners. The course was targeted at both 3rd and 4th year students; the 3rd year students worked as members of the team while 4th years acted as team leaders. Feedback from stakeholders has been encouraging and the students acknowledged that the course helped them gain strong professional skills.

Similarly, [31] developed a capstone project course with industrial customers to teach real-world software engineering by simulating full software development projects, and creatively used master's students as project managers and undergraduates as developers. However, while these approaches, modules, and courses have recorded reasonable success, the modules mostly targeted undergraduate students and focused predominantly on teaching students' technical content while working in teams. Thus, the soft skills required for appropriate teamwork were not always emphasised.

Using a case study strategy [32] informed by an inductive approach [33] and qualitative methodology [34], we critically examine the process of creating our own module, teaching the module, and reviewing what students learned. Specifically, this paper aims to address three research questions: RQ1) What approaches have been adopted to teach technical and soft skills required by employers of software engineering graduates? RQ2) What skills do students acquire when working in a multicultural team to develop software? RQ3) What did we want students to come away with, and to what extent did we succeed?

III. Background and Methodology

The Department of Computer Science at Sheffield University has an annual intake of around 210 MSc students, split across six programmes. Of these, just over 100 are studying MSc Advanced Computer Science (ACS) or MSc Computer Science with Speech and Language Processing (CSSLP, a specialised version of ACS). Both programmes are well established and have been running for many years - nonetheless regular reviews are undertaken, both to keep the programmes current and for BCS accreditation purposes. Following the most recent review, it was decided to boost the amount of teamwork by introducing a new second-semester 15-credit Team Software Project module for both ACS and CSSLP students. This was launched last year, running for the first time during the 2018-2019 session.

A total of 104 students were enrolled on the module (20 CSSLP, 84 ACS). In order to keep team sizes manageable, these were split into 18 teams whose members were assigned by the module leaders (fourteen teams had six members, four teams had five members, and each team included at least 1 CSSLP student). There were three different projects, with six teams competing on each. The majority of students taking the module were from non-British backgrounds, and this resulted in much of the teamwork having an unexpected multicultural dimension. While this introduced additional communication overheads for many teams, overcoming this problem had unexpected side benefits for many of the students.

For this first run of the module, all three projects were specified from within the department, and dealt with problems associated with monitoring departmental learning/teaching activities that were directly relevant to the modules taken by the students themselves. It was felt that this would both help motivate the students, and also make it easier for them to understand the relevant organisational context. Project 1 required the development of a system for monitoring the spread of assessment loadings on students across the academic year (this varies from student to student, depending on the modules they have chosen), so as to help year tutors identify and mitigate overloading. Project 2 required teams to extend and improve an existing in-house system for detecting plagiarism of program code. Project 3 asked for a system capable of relating module-level assessments to programme-level learning outcomes, so as to enable automatic year-on-year checking that these outcomes remain fully supported and assessed as new modules are introduced and old ones deleted from the curriculum.

We employed three PhD students as project mentors, one for each project, to provide individual advice to the teams working on their project and attend their weekly stand-up meetings.

The first task for each of the teams was to come up with a team name and choose which project they would like to work on by indicating their first, second and third choice. The actual assignment of teams to projects was done by the module tutors. While it was not possible to allocate all teams their preferred projects, all but four teams were assigned either their first or second choice.

It was explained during the Week 1 presentations that the main focus of the module was on teamwork, and that while we would be looking at the systems they produced, we were keen that teams should manage themselves as effectively as possible. This allowed students with, e.g., excellent writing skills to contribute fully to their teams' outputs by focussing on documentation even when their programming skills were limited (these skills are already assessed in other modules).

Nonetheless, the information we provided at this stage was deliberately incomplete; students had to establish team dynamics for themselves, while at the same time analysing their project requirements, deciding on development techniques, and deciding who would do what. This was always going to be stressful for team members, but we took the time to answer questions promptly as and when they arose, held weekly debriefing meetings with the mentors and allowed teams the whole of Sprint 1 (a period of 3 weeks) to sort themselves out and establish their working arrangements.

At the end of each sprint, the teams were required to produce a short report detailing what they had planned to do during the sprint, what they eventually achieved, the plan for the next sprint and evidence of their regular meetings. This report allowed the module tutors to provide timely formative feedback to each team before the end of the second sprint, thus providing an opportunity for the students to learn from mistakes in the earlier sprints. Table 1 below provides a breakdown of the structure of the module

TABLE I. Structure of the Module

Time	Activity
Week 1	• Introductory lecture • Project briefing session • Meet your team members
End of Week 1	• Indicate Team preferences
Week 2-4	• Sprint 1
End of Week 4	• Submission of progress report
Week 5-7	• Sprint 2 • Client meetings
End of Week 7	• Submission of progress report
Week 8-10	• Sprint 3
Week 11	• Team Presentation & Submission

Just like in a typical agile project, the clients were actively involved in the module: by providing a written project brief describing the core features of the software, its target audience and technology limitations; presenting the projects to the students; fielding questions throughout the project either in person or by email; meeting students at the end of the first sprint to provide feedback on what had been achieved so far; attending the final presentations; assessing the software developed and providing feedback to the teams. The clients were also available to clarify requirements via emails or for a face-to-face meeting with the teams depending on the teams preferences.

General communication skills, motivation and engagement were monitored and facilitated via meetings with the mentors, and the module was assessed on the basis of five key deliverables/criteria:

- **Team Documentation** - This showed whether the teams could adequately describe/present their solution so that it could be used by the client in their absence. Skills assessed included: critical assessment of information, the organisation and expression of ideas, and evidence of appropriate use of agile processes.
- **Software Developed** – We (with the client) assessed how well the delivered software met the requirements specified from a functional and usability perspective. Evidence of adequate testing and overall code quality were also taking into consideration.
- **Teamwork** - We examined how the team self-managed. Specifically, we examined the contribution of team members, how work was split amongst the members and reviewed notes made by the mentors during the weekly stand-ups.
- **Presentation** - Each team was allocated 15 minutes and every member of the team was expected to contribute to this activity. The teams presented their solution and approach to a panel comprising their client, their mentor and one of the module tutors.
- **Individual Reflection** - At the end of the module, each student was required to submit a short reflection on what their contribution to the project had been, and what had been learnt.

In addition to forming the basis for the students' grades, this substantial body of information also provided us with a basis for assessing how well we achieved our own goals. In particular, the individual reflections provided information about soft skills and learning outcomes, as seen by the students themselves, as well as general information about their confidence in using these newly acquired skills, and the extent to which team members contributed equally. Our weekly meetings with the mentors provided information as to how team dynamics developed and the extent to which members supported each other and the team as a whole. The inclusion of clients in all stages of the project (including assessment) allowed us to assess how well a module of this kind can address complex requirements over periods as short as a single semester.

An anonymous Module Evaluation Survey was also sent to the students towards the end of the module. This was part of a general survey of modules undertaken each year and reviewed by both the department's Teaching Operations Committee and it Staff-Student Liaison Committee (SSLCOM). The survey affords students an opportunity to give their views on module content and delivery, both by giving basic satisfaction ratings and by providing free-text responses. The results are collated and presented to module leaders, who are then required to provide a timely response to SSLCOM, commenting on the feedback and outlining any changes that they intend making for the coming year.

IV. Results and Lessons Learned

In order to minimise problems associated with students being unwilling to criticise their colleagues in public, we also provided means for students to comment anonymously at any time and on any aspect of the module (including module content, mentors, clients, module leaders and fellow team members). Surprisingly, only one such comment was received, detailing the concern a student had for the apparent non-contribution of a fellow team member – they felt unable to discuss this openly during team meetings. In addition, one student approached us privately to discuss what appeared to be a culturally-motivated dispute between two of their team members. We dealt with this (to the student's satisfaction) by informally reminding all teams, as part of our general ongoing feedback, of the potential benefits of combining members with very different backgrounds and skills, and the advantages of maintaining team harmony.

Feedback from both the mentors and the clients confirmed that students were extremely nervous at first as to what was required of them, but by the end of the first sprint this nervousness had largely subsided, and anecdotal evidence (including details in their Individual Reflections) suggests that students ended the module with an enhanced sense of their own ability to handle ill-defined requirements. Some international students, in particular, whose English language skills had initially been weak, expressed surprise at how fast their conversational skills improved once they started having regular stand-up meetings with their colleagues.

Overall, the students engaged very well, with very good attendance record and team spirit, and clients rated the quality of the delivered systems as very good. For many of the students, they started the module with no knowledge of Agile methodology and at the end of the module, they expressed confidence with the various concepts and ceremonies in a scrum based agile project. For many, this was their first experience working in a multi-cultural team, developing real-world software to the specification provided by a client.

Particularly, this module provided several students the opportunity to leverage existing skills (programming, automated testing, documentation) and acquire new ones like GIT version control and working with complex databases. In many teams, there was also evidence of self-directed peer learning; one student indicated, for example, that they had learnt Python and RESTFUL API development in support of a requirement self-imposed by the team. By working together in teams developing software to meet the deadlines, the feedback suggests that students quickly realised the need for effective communication and, in particular, that version control tools do not replace the need to clarify technical direction with other members of the team.

Another student commented on how the use of prototyping could help clarify complex requirements and explained how their team used that approach. In general, the students were also able to experience project management first-hand, with some students particularly taking leadership roles within their scrum teams such as development team lead and project manager.

With regard to the Module Evaluation Survey, 73 responses were received (69.5% of students taking the module). Of these, 80% indicated that the module was interesting, challenging and helped them to learn, while 91% indicated that they were able to use the feedback provided effectively and 90% indicated that they were satisfied with the module. Indeed, levels of engagement were so high that many students seem to have contributed significantly more time to their work than the credit value of the module warranted. While this led to a remarkably high average module grade of 74% for students taking the module, it is possible this was achieved at the expense of lower grades in other modules, and steps will be needed to ensure that students are aware of the potential consequences of allocating time unevenly across the curriculum.

Communications were a major factor in multicultural teams. For example, a typical team might comprise three Chinese and two English students, with at least one of the Chinese students having poor English and at least one of the English students having poor (or non-existent) Mandarin. Of the two English students, one might be a CSSLP student with little system design experience. There are thus two cultural divides at work, and these were left to the students to resolve; they did so in a variety of ways. In some cases, students who were excellent at both languages were appointed to translate during team meetings (and likewise, translators might be appointed to explain technical details to students with non-programming backgrounds). In others, the two groups held separate ad-hoc meetings, and then came back together at (translator facilitated) weekly stand-ups to ensure everything remained on track. Many teams used international communication platforms to communicate (especially WeChat). The use of diagrams and sketches was also a useful cross-language tool.

Although the focus of the module was on teamwork, the quality of systems produced by the teams was extremely high. Teams typically identified a large number of technologies and languages that would be needed to complete the project, and then put in a considerable amount of effort to master them (different team members were allocated different technologies and languages to focus on, none of which were formal components of either the ACS or CSSLP programmes). It appears that providing only limited information at the start of the module prompted the students to identify required skills for themselves, and then to share the workload involved in acquiring them.

A common problem with teamwork-based modules is the feeling of some students that their teammates are not pulling their weight. The team documentation, individual reflections and (lack of) complaints all suggest that this problem largely evaporated, because students with weak programming skills explicitly acknowledged that they would take on roles that contributed to overall team output, even if no programming was involved. It likewise helped the excellent programmers who were poor at expressing themselves in writing. Each student was able to use their own skills to the full, knowing their fellow team members could be trusted to fill the gaps.

Given the number of students taking the module, one of the expected challenges was the complexity involved in managing the module, ensuring that students were able to learn the course material, and providing timely formative feedback to the students. But by adopting the module structure and leveraging the scrum framework, several forms of formative feedback were provided on a continual basis through the team mentors/scrum masters, the clients and also on the feedback on the report of progress provided by the module tutors after every sprint.

Overall, the clients were pleased with the amount of work completed by most teams. While some did not deliver all the features asked for, others put in extra time and ensured that they delivered all the features in the project. The level of professionalism displayed by the teams throughout the project and particularly during the presentations was commented on by both the clients and the module tutors.

V. Areas for Improvement

While this initial run of the module has gone well, we feel that a number of changes/additions could be made. First, to further increase the relevance of this module to the students' employability, the students could be assigned projects hosted on public Git repositories such as GitHub as this will add to their portfolios for review by potential employers. This year, the projects were not particularly confidential, but perhaps still not appropriate to be hosted publicly.

Secondly, most of the systems were implemented as web-based software applications that could be tackled by any team of computer science students but feedback from CSSLP students shows they lacked opportunities to fully leverage their specialist knowledge. In future we will consider sourcing projects that are entirely (or at least, more specifically) relevant to CSSLP students. This might, however, mean segregating the ACS and CSSLP students, and needs further consideration.

Also, the feedback we received suggests that some teams were confused by the various agile terminologies presented during the introductory period (although the team mentors were able to help clarify some of this and the module tutors provided further slides/resources to help resolve the problem). In the next run of the module, we believe it would be beneficial to the students to have more lectures available for those teams that need them, perhaps delivered by way of pe-recorded videos providing some form of blended learning [12].

VI. Conclusions

Based on module feedback and the students' individual reflections, the students gained a wide range of soft skills including teamwork, project management, communication, problem-solving, accountability and time management as well as technical skills such as source control (using Git), testing, debugging, algorithm development and knowledge of agile; all of which are relevant for a software engineering or computer science career in industry.

In this paper, we have presented a unique module design that focuses on helping students acquire non-technical competencies that are relevant for a software engineering career which makes use of a combination of project based learning and an agile-based scrum framework.

With respect to research question 1 (RQ1), we conducted a literature review of various approaches that have been utilised to teach the technical skills required by employers, including project-based learning, practical laboratory sessions and traditional teaching approaches. To teach soft skills, some modules have embedded guest lectures from industry into their design, having industry practitioners act as lecturers or mentors, explored project-based learning with students working as consultants to companies, and approaches which involve students starting a start-up or a digital agency. However, in many approaches the teaching of soft skills was implemented as an 'add-on' to modules mostly focused on technical content. Overall, we found that modules, where a decent balance of technical and soft skills were acquired by the students, made use of active learning and contained project elements.

To address RQ2, we reviewed the students' feedback provided at the end of the module, including individual reflection submitted by all 105 students, and reflected on notes made by the module tutors and mentors during the semester, as well as client feedback. The evidence, though anecdotal, suggests that students have been able to gain several key technical skills (new languages, familiarity with Git, requirement elicitation skills) as well as knowledge of agile. Moreover, the module appears to have achieved its learning outcomes of getting students to engage with many key aspects of team working: presentation, collaboration, mutual support, project management, communication, problem-solving, accountability and time management.

With respect to RQ3, the module was designed to provide students an opportunity to work as constructive and effective members of a team, and to improve their professional and interpersonal skills while solving real world software problems. Through our approach of using project-based learning facilitated by the use of an agile based scrum framework, we succeeded in providing the students an opportunity to learn new technical skills, acquiring knowledge on agile and experiencing what working as a software engineer feels like in practice.

Nonetheless, key questions remain. In particular, given cultural differences in attitudes towards competition, how much does competition within teams depend on the cultural backgrounds of its members, and does this have positive or negative effect on team dynamics? Our experience shows that language problems can largely be overcome, but how well do the ad hoc solutions adopted by our teams translate to the much larger projects prevalent in some IT sectors? Likewise, we specifically appointed mentors we knew would be capable of handling the various teams – but in a more general setting, to what extent would the mentor's own experience be a key factor in overcoming problems arising through multicultural alignments and differences?

In summary, while our approach has many features to recommend it, and in particular appears to have succeeded in overcoming problems associated with multicultural team working, further work is, as always, required.

References

[1] S. Heggen and C. Myers, "A Study in Developing Self-Confidence and Marketable Skills," pp. 32–39, 2018.

[2] L. M. Grabowski, C. F. Reilly, and W. A. Lawrence-fowler, "Emulating a Corporate Software Development Environment Through Collaboration Between Student Projects in Six Courses," 2014.

[3] N. H. El-khalili, "Teaching Agile Software Engineering Using Problem- Based Learning," Int. J. Inf. Commun. Technol. Educ., vol. 9, no. September, 2013.

[4] M. E. Morales-Trujillo and G. A. Garcia-Mireles, "Participating in an industry based social service program: A report of student perception of what they learn and what they need," Proc. 2019 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2019, vol. 18, pp. 861–870, 2019.

[5] P. Grimaldi, L. Perrotta, V. Corvello, and S. Verteramo, "An agile, measurable and scalable approach to deliver software applications in a large enterprise," Int. J. Agil. Syst. Manag., vol. 9, no. 4, pp. 326–339, 2016.

[6] J. Marino, "Goldman Sachs is a tech company," Business Insider, 2015. [Online]. Available: https://www.businessinsider.com/goldman-sachs-has-more-engineers-than-facebook-2015-4?r=US&IR=T. [Accessed: 28-Dec-2019].

[7] D. Mishra, A. Mishra, and S. Ostrovska, "Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation," Inf. Softw. Technol., vol. 54, no. 10, pp. 1067–1078, 2012.

[8] M. Liu and Y.-P. Hsiao, "Middle school students as multimedia designers: A project-based learning approach," J. Interact. Learn. Res., vol. 13, no. 4, pp. 311–337, 2002.

[9] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with Scratch," Comput. Educ., vol. 120, no. July 2017, pp. 64–74, 2018.

[10] M. Spichkova, "Industry-oriented project-based learning of software engineering," Proc. IEEE Int. Conf. Eng. Complex Comput. Syst. ICECCS, vol. 2019-Novem, pp. 51–60, 2019.

[11] J. A. Maćias, "Enhancing project-based learning in software engineering lab teaching through an e-portfolio approach," IEEE Trans. Educ., vol. 55, no. 4, pp. 502–507, 2012.

[12] A. Meikleham, R. Hugo, and R. Brennan, "BLENDED AND PROJECT-BASED LEARNING : THE GOOD , THE BAD , AND THE UGLY," in 14th International CDIO Conference, 2018.

[13] D. Petkovic, R. Todtenhoefer, and G. Thompson, "Teaching Practical Software Engineering and Global Software Engineering : Case Study and Recommendations," pp. 19–24, 2006.

[14] A. Heberle, R. Neumann, I. Stengel, and S. Regier, "Teaching Agile Principles and Software Engineering Concepts through Real-Life Projects," pp. 1723–1728, 2018.

[15] J. Siegeris and H. Barke, "Agile for agile - new ideas for the transformation of student projects Juliane," in Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, 2019, no. October, pp. 151–163.

[16] V. Devedžić and S. R. Milenković, "Teaching agile software development: A case study," IEEE Trans. Educ., vol. 54, no. 2, pp. 273–278, 2011.

[17] J. H. Sharp and G. Lang, "Agile in teaching and learning: Conceptual framework and research agenda," J. Inf. Syst. Educ., vol. 29, no. 2, pp. 45–52, 2018.

[18] C. Sibona, S. Pourreza, and S. Hill, "Origami: An active learning exercise for scrum project management," J. Inf. Syst. Educ., vol. 29, no. 2, pp. 105–116, 2018.

[19] M. Holcombe, M. Gheorghe, and F. Macias, "Teaching XP for real: Some initial observations and plans," pp. 14–17, 2001.

[20] A. Stratton, M. Holcombe, and P. Croll, "Improving the quality of Software Engineering courses through University Based Industrial Projects .," 1998.

[21] S. Ul Huda, T. S. Ali, K. Nanji, and S. Cassum, "Perceptions of Undergraduate Nursing Students Regarding Active Learning Strategies, and Benefits of Active Learning," Int. J. Nurs. Educ., vol. 8, no. 4, p. 193, 2016.

[22] S. Freeman et al., "Active learning increases student performance in science, engineering, and mathematics," Proc. Natl. Acad. Sci., vol. 111, no. 23, pp. 8410–8415, Jun. 2014.

[23] L. O. Seman, R. Hausmann, and E. A. Bezerra, "On the students' perceptions of the knowledge formation when submitted to a Project-Based Learning environment using web applications," Comput. Educ., vol. 117, no. October 2017, pp. 16–30, 2018.

[24] M. Bernstein, K. M. Fitzgerald, J. P. Macdonell, and A. I. Concepcion, “AlgorithmA Project : The Ten-Week Mock Software Company,” pp. 142–146, 2005.

[25] R. E. Bruhn and J. Camp, “Capstone Course Creates Useful Business Products and Corporate-Ready Students,” vol. 36, no. 2, pp. 87–92, 2004.

[26] K. Gary, B. Gannod, G. Gannod, H. Koehnemann, T. Lindquist, and R. Whitehouse, “Work in Progress – The Software Enterprise,” pp. 19–20, 2005.

[27] A. Rusu and M. Swenson, “An Industry-Academia Team-Teaching Case Study for Software Engineering Capstone Courses,” pp. 18–23, 2008.

[28] F. Meawad, “The Virtual Agile Enterprise : Making the Most of a Software Engineering Course,” pp. 324–332, 2011.

[29] K. Buffardi, C. Robb, and D. Rahn, “Learning Agile with Tech Startup Software Engineering Projects,” pp. 28–33, 2017.

[30] L. Johns-boast and S. Flint, “Simulating Industry : An Innovative Software Engineering Capstone Design Course,” in 2013 IEEE Frontiers in Education Conference (FIE), 2013, pp. 1782–1788.

[31] J. Vanhanen, T. O. A. Lehtinen, and C. Lassenius, “Teaching Real-World Software Engineering through a Capstone Project Course with Industrial Customers,” in Proceedings of the First International Workshop on Software Engineering Education Based on Real-World Experiences, 2012, pp. 29–32.

[32] R. K. Yin, Case Study Research and Applications: Design and Methods. SAGE Publications, 2017.

[33] D. E. Gray, Doing Research in the Business World. SAGE Publications, 2016.

[34] D. Silverman, Interpreting Qualitative Data. SAGE Publications, 2015.

[35] M. Kropp, A. Meier, and R. Biddle, “Teaching agile collaboration skills in the classroom,” Proc. - 2016 IEEE 29th Conf. Softw. Eng. Educ. Training, CSEEandT 2016, pp. 118–127, 2016.