A Case Study on Finding Convenient Approach to Practice Agile Methodologies in Software Engineering Courses

Boris Milašinović, Toni Bakarčić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{boris.milasinovic, toni.bakarcic}@fer.hr

Abstract*. Dominance of agile methodologies triggered the need to include elements of agile methodologies in software engineering education. However, it turns out that finding convenient approach to practice agile methodologies in software engineering courses is not an easy task and could be full of pitfalls and obstacles. Based on the previous experiences and literature overview it can be concluded that it is not possible, or it could be extremely ineffective to use a methodology "as-is" in educational context, due to various problems in terms of course organization, work schedule and shortage of staff. Instead, method tailoring must be done, and the authors propose a method for a gentle introduction to agile methodologies by combining use of agile techniques like user stories and Kanban with continuous integration/deployment as one of the common engineering practices. The approach is a result of analysis of the mistakes from the past attempts and two semester long experiment conducted in authors' environment. The introduction to agile techniques could be done in an optional manner not obligating students to strictly follow it. They would not be graded for following it, as its purpose is that students should conclude that it could be useful to them and that it is not just another (from their perspective) useless obligation. Initial questionnaires show that students understand the benefits of the used techniques and they would continue to explore other agile techniques.*

Keywords. software engineering, education, agile, Kanban, Scrum.

1 Introduction

In a competitive and changing IT market, practical experiences in various real-life projects could give students a distinct advantage over those who lack such experience (Orr, 2015). Therefore, traditional teaching based on theoretical fundamentals has been replaced or at least enriched with new practices trying to introduce real-life problems (Fertalj, Milašinović, & Nižetić, 2013) into software engineering courses. However, replacing hypothetical problem topics in courses with practical ones do not solve the gap between market needs and education system practices, and it is not the only issues that must be dealt with.

Project planning and estimation in terms of software cost, development time and effort is one of the most complex tasks (Čeke & Milašinović, 2015). As (Král & Žemlička, 2014) summarized in their paper, the problems frequently occur in the planning and managing phase rather than in the developing phase, or as a failure of development responsibilities. The afore mentioned papers, and remarks from (Martin, Anslow, & Johnson, 2017) leads to conclusion that technical excellence is not the sole important factor and that some other skills beyond programming are needed, and those skills are not always easy to learn or acquire (Alfonso & Botia, 2005).

The rise of agile methodologies, especially of Scrum or Scrum in combinations with Extreme programming (XP) and Kanban as the most dominant agile methodologies (*CollabNet VersionOne 13th Annual State of Agile Report*, 2019), inevitably causes need to include these methodologies to software engineering education. However, it turns out that it is not an easy task and could be full of obstacles (Milašinović, 2018; Milašinović & Fertalj, 2018). More on these problems is discussed in the next section that gives an overview of related works that elaborate general problems of course organization and students' preparation, followed by the discussion of problems somehow unique to education context like problem of grading and work schedule. The second section ends with an overview of works on method tailoring that supports the authors' approach conducted and described in this paper. Authors' context is described in the third section. As a response to spotted problems, two questionnaires and some experiments had been done in the current academic year which is elaborated in the fourth section, followed by the results discussion and description of threats to validity. The paper ends with conclusion and guidelines for further work and improvements.

2 Related Works

Various issues in adopting agile methods in software engineering education e.g. lack of training, resistance to changes, problematic teamwork, administrative effort, etc. had been already noticed as potential problems by (Rico & Sayani, 2009) in their papers survey, and further extended in systematic literature reviews of (Mahnič, 2015). In addition to those papers, systematic literature review on agile method tailoring (Campanelli & Parreiras, 2015) and the latest *CollabNet VersionOne* agile report (*CollabNet VersionOne 13th Annual State of Agile Report*, 2019) were used to track trends on usage of agile tools.

2.1 Course Organization and Students' Preparation

Usually, a common place to introduce agile methodology is a capstone project ("Capstone Project Definition - The Glossary of Education Reform," 2016). Although some of authors suggest how a capstone project should be organized e.g. (Mahnič, 2012), there is no unique opinion on its duration and organization as there are many different approaches as enumerated in (Milašinović & Fertalj, 2018).

There are also many different approaches to students' preparation to such course as using agile games, having previous training, doing initial few weeks of observations, etc. Various approaches had been already summarized in (Milašinović & Fertalj, 2018) and can be additionally extended by two new papers which further disperse set of possible approaches: (Chauhan, Probst, & Babar, 2019) suggest that first sprint should start 2 weeks of the first lecture, and (Hurbungs & Nagowah, 2019) had divided a course into classroom activities in which games typical for a particular agile methodology have been played, and labs activities for using agile tools to create user stories, maintain backlog, do pair programming etc.

2.1 Education context problems - Motivation, grading, work schedule, and role distribution

In addition to curriculum and capstone project organization problems, there are also some other elements that significantly differs in real-life situation and in educational context.

Students usually get into a habit of solving the assigned tasks focusing on the grades and deadline instead of on product quality (Murphy, Sheth, & Morton, 2017) and find themselves more comfortable in some kind of waterfall approach (Rodriguez, Soria, & Campo, 2016). Although according to (Hurbungs & Nagowah, 2019) pair programming increase students retention and confidence, this practice that is desired in real projects could jeopardise grading process by masking individual contribution. Thus, an individual work must be recognized and valued appropriately (Fertalj et al., 2013) or even with custom metrics for deliverables (Gamble & Hale, 2013).

Companies usually adopt practices from a field of project management to improve efficiency where such practices are not beneficial to students in such extent; one of the reasons could be the differences in work and schedule organization, and lack of common working place. Distributed environment may cause additional effort for organizing meetings (Freitas Santana et al., 2017; Rodriguez et al., 2016), and lack of face to face meetings can lead to misunderstandings and mistrust (Rodriguez, Soria, & Campo, 2015). Although (Masood, Hoda, & Blincoe, 2018) reported usage of virtual collaborative environment and chat bots instead of classical stand-up meetings, they still suggest that the meetings should rather be done in person when possible even with reduced frequency, and this practice has been done by some other authors, e.g. every third day (Olszewska, Ostroumov, & Olszewski, 2017) or every two weeks with virtual meetings in between (Freitas Santana et al., 2017).

The problem of role distribution also remain unsolved or at least without unique opinion on that. (Mahnič, 2015) notes two main approaches for Scrum master role assignments: a teacher or a student, but there are also some hybrid approaches like rotating role (Meier, Kropp, & Perellano, 2016; Rodriguez et al., 2016), use of research assistants (Scharf & Koch, 2013), engaging of students that previously passed the course both as teaching assistants and as project managers (Murphy et al., 2017) or introduction of coaching roles taken by lecturer (Meier et al., 2016) desirably not a member of the team (Rodriguez et al., 2016). (May, York, & Lane, 2016) recommend that those who are enrolled as Scrum master should have Scrum certificate.

2.3 Method tailoring

Systematic literature surveys of agile method tailoring done by (Campanelli & Parreiras, 2015), and (F. Tripp & Armstrong, 2018) supported by (*CollabNet VersionOne 13th Annual State of Agile Report*, 2019), shows the percentage of adoption rates of agile and engineering practices and the benefits of adoption of a particular practice. From the cited sources, it can be concluded that the most adopted practices are categorized in project management category where only unit testing and coding standards from software development approach category are in top 10 adopted agile techniques and engineering practices. There are also some examples of successful applications of agile methodologies not necessary related to software development, e.g. in learning and teaching units described by (Judd & Blair, 2019), stressing even more that the biggest benefit of agile practices is in management area. Naturally, this is expected and aligned with the claim that something more beyond technical excellence is needed (Martin et al., 2017).

3 Authors' context

There have been two possible courses during six semesters long undergraduate study of Computing in which authors could introduce or practice elements of agile methodologies: the course *Project* in the 5th semester and the course *Development of Software Applications* in the 6th semester.

Student at the third year of the study should be familiar with object-oriented programming and able to create a database and manage it using SQL. Some elements of software engineering as models of software engineering processes, requirement engineering, UML modelling, and software testing are introduced in the 5th semester in course *Software Design*.

3.1 The course *Project*

The course *Project* consists of mentorship work, and has loose week schedule with only few formal checkpoints, thus making it ideal to serve as a capstone project. However as its loose structure is a benefit, also it could be a drawback. Common perception of a course with small number of formal obligations combined with the lack of students' seriousness and proper attitude to regular work is common cause of problems as students (in most cases) tend to give priority to other courses in semester that have more strict rules.

Another drawback of the course lies in the current study program in terms of previous students' knowledge, as they lack the knowledge of general software engineering concepts and advanced programming techniques. The lack of knowledge and proper organizational skills caused that a lecturer must be not only a teacher that acts as a product owner and agile coach, but also some kind of lead developer at the same time, that is somehow schizophrenic and hard to successfully emulate.

Furthermore, number of students that would get insight to agile methodology in this course is rather small and depends on lecturers' affinity and free time.

3.2 The course *Development of Software Applications*

The average number of students enrolled to *Development of Software Application* each year is usually between 90 and 110. Organization of the course, teams organization and week schedule had been described in (Fertalj et al., 2013) and it has been used with slight modifications through the years. The last significant modification was to increase homework share in the final grade to 55% and make use of user stories as an optional task. The rest of points is divided on two exams with theoretical question (2 x 20%) and 5% for active participation during lectures.

Table 1. Homework elements in *Development of Software Applications* course

	Weeks	Percent of grade
Interview minutes	2	2%
Conceptual model	2-3	3%
Physical model	2-3	4%
Project plan	4-6	2%
Requirements specifications	4-7	2%
Design specifications	4-7	2%
Web-application (part 1. CRUD operations)	6-11	15%
Web-application (part 2. Complex interaction, reporting, layering)	11-15	20%
Technical documentation	15	2%
User manual	15	3%

The homework elements are shown in Table 1. The table shows homework topics with weeks in which students are expected to work on particular homework for a semester consisting of 15 weeks. Grading process is mostly individualized by dividing students inside a group by theme in order to enable students not to be penalized for failures of other members. Common tasks are related to modelling, planning, integration process, and development of common libraries.

The structure and topics of the course follow the elements of software engineering techniques and thus it looks like a good place for addition of agile elements, although agile methodology is not mandatory nor one of course outcome. As it is noted that students could have significant problems in creating correct database model based on user requirements, and errors in model would amplify errors in development phase, their models and specifications are checked and corrected in weeks 6 and 7.

Due to that, the first 6 to 7 weeks resemble more to waterfall model, but in the next 9 weeks students have to elaborate requirements, do detailed design and develop a web application and it is the good place to introduce an agile methodology or some of its elements.

Some previous attempts to introduce agile elements were quite unsuccessful. Except common problems found in literature like project quitting, delays due to sicknesses of team members, distraction with other obligations, lack of staff to fulfil the tasks has shown as one of the key elements. Most of the papers does not reveal the staff to students ratio, but from some notable examples, e.g. 3 course instructors and 35 students in (Chauhan et al., 2019) or having local industry representatives as product owners in (Masood et al., 2018), it is obvious that most courses organizations cannot be replicated due to lack of staff. Therefore,

making recommendations for educators stated by (Masood et al., 2018), looks like a set of nice but not achievable wishes.

To introduce some basic agile elements, an initial idea was to create new homework consisting of entering user stories that would be refined later and that should be used to track developing progress. As the most points are awarded for development and the fact that students are individually graded, students were more focused on own development (practices) and could not (significantly) benefit from a better teamwork and organization in latter phases of the course. Also, they were not able to understand the benefits of refining user stories and treated them just as another obligation they have to do. Moreover, in most cases they finished development tasks and then write user stories to mirror the tasks they had already done. In the meantime, course staff was unable to work more with students and could not practice and control meetings, product backlog refining, etc., and students have just found those elements as an additional work that has to be done only because they have been told to do that.

4 A case study

In order to avoid previous pitfalls described in the previous sections a different approach has been used for courses *Project*, and *Development of Software Applications* in academic year 2018/19. The first change was to exclude lecturers from any kind of agile coach roles or Scrum master role in order to be able to focus on emulating a customer and on grading process, and to find out how it affects the whole process. Those roles have been assigned to a master degree student as part of his project preceding his master thesis and as a test how much students would be relaxed when working in less formal context with the student that leads them.

4.1 Preparation

4.1.1 Agile tools analysis

Initial preparation consisted of an analysis of agile tools conducted to find an appropriate tool for use in the classroom. The tools were divided into the following categories: communication tools, repositories, versioning tools and issue tracking tools. Although the tools test was not a systematic review, in authors' opinion several most important have been tested and based on the conducted test and some previous experiences the decision was to use Slack for communication with the students, and Google Docs for writing the documentation. In the course *Project*, both *GitHub* and *BitBucket* were used as source code repositories together with *BitBucket Pipelines* used for continuous integration. They were replaced with Microsoft Team Foundation Server (academic licence) in the course *Development of Software Applications* in order to enable continuous deploy to on premise servers. Initially JIRA was used for project management, but it was replaced with Trello as usage of JIRA for large teams would generate significant costs due to its licencing model.

After defining the tools, the appropriate method had to be chosen. Given the drawbacks described previously, it was decided that none of the agile methods could be used in its full capacity, and that method tailoring should be used.

4.1.2 Course *Project* as a "dress rehearsal"

In the second phase of preparation the course *Project* was chosen as a playground and "dress rehearsal". Having decided on the approach to the class, and seen the general foreknowledge of the students, the course started with more insight on what the students need to learn and how.

The 9 students were divided into 2 groups of 5, with one student being assigned for the continuous integration in both teams. Given that one of the teams was more experienced and homogeneous, Scrum was selected for them. For the less experienced team a simpler variant with Kanban board and the elements of Extreme programming was chosen. The Scrum team obtained slightly better results but it can be explained by the fact that the team already had better cohesion and better development skills so comparison between two used methods cannot be established.

The important observation is that it turned out that use of Kanban have raised the awareness of all team members and motivated them to be more responsible. Seeing that team progress would be jeopardized by waiting someone's tasks is important motivational fact.

4.1.3 Students survey

To have a better overview of the students' experience and foreknowledge, a survey was conducted among the students who have just finished their fifth semester and enrolled to *Development of Software Application* course.

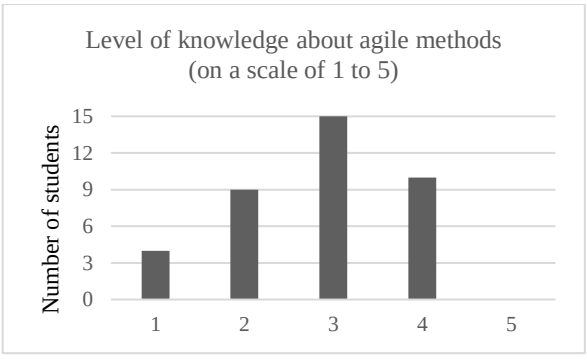

Figure 1. Students' knowledge about agile methods

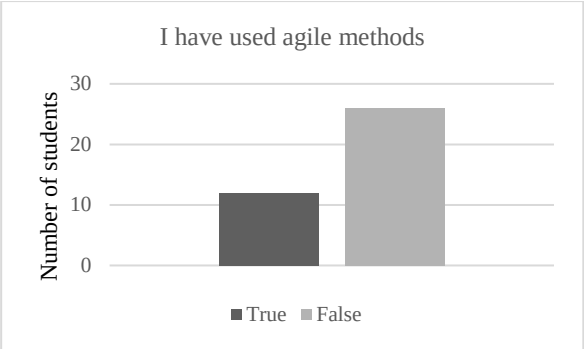

Figure 2. Students' familiarity with agile methods

It showed that students had some theoretical knowledge about agile methods, but less than a third had used them. That is because they had only worked on a couple of team projects beforehand, and none of those required the use of agile methods.

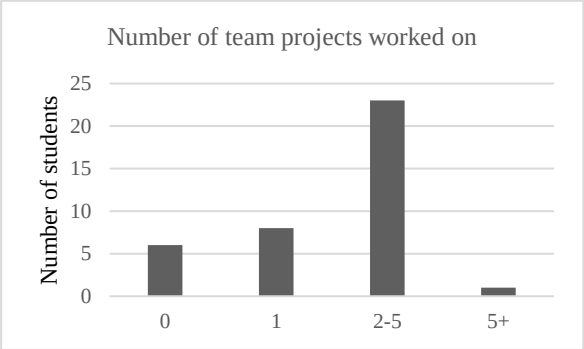

Figure 3. Students' previous experience

4.2 Kanban tutorial

After the results of the survey came in, a project organization exercise was held for the students of the course. Students were given their project assignment, and in the following week, a short tutorial was held on agile methods and Trello. After the tutorial, they were given 20 minutes to create a Kanban board in Trello and fill the backlog with at least 5 user stories, broken down into at least 3 subtasks.

A total of 54 students took part in the exercise, divided into 9 equal groups of 6 members. 3 teams fully completed their assignment, 3 teams created enough user stories but didn't break them down into subtasks, while the remaining 3 teams struggled and couldn't identify the main requirements of their project.

Through talking with the teams, it was noticeable that the teams that did well had at least one member with previous working experience, where they used agile methods. The teams that did not complete the task had no students with working experience; however, they said that they had knowledge about agile methods, and that they had worked on student projects.

4.3 Results and threats to validity

The Kanban exercise and the survey showed that there is a lack of practical knowledge among the students that only attend previous courses at the university but have not be involved in any other projects. The project showed that the students were excited about using agile methods, and it had an impact on their teamwork skills.

A couple of weeks after the Kanban exercise took place, the same students were asked to rate the likelihood of using agile methods on their upcoming projects on a scale from 1 to 5. The results showed that the majority of students are very likely to use them which shows that they are interested in bringing a practical use of agile methodologies in our courses.

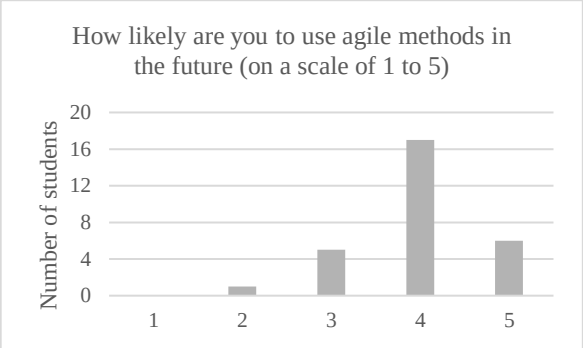

Figure 4. Students' likelihood of using agile methods

The main threat of the results is the small number of students involved in the project. Roughly 10% of the generation took part in the Kanban exercise, 5% filled out the survey, but only 3% were selected for the project.

5 Conclusions

Introduction of agile elements is inevitable, but in cases when there is a shortage of staff, industry partners and/or enough project problem themes (topics), and where the significant part of the course is devoted to learning development techniques then using a particular methodology as-is could be a wrong step depending on the course organization and main ideas. Instead, some kind of method tailoring must be done. However, tailoring by just following the trends and introducing mostly adopted practices from companies (i.e. daily stand-up, retrospectives, reviews) also could lead to discrepancies between expectation and results. Putting accent on project management part requires different grading model and week schedule that makes students free to organize being not limited by lecturing progress and homework topics, which is not always achievable due to curriculum and course aim. Thus, it could be only suitable for capstone projects with no (or small number of) lectures on which development depends. If the course accent is set on development techniques and individual grading is inevitable like in authors' context, then strictly formalizing the most

adopted agile techniques like meetings, planning, and retrospectives could just create an additional effort to students and staff, and lead to frustrations. If the homework elements are known in advance and students are divided by project topics in order to be graded individually, then meetings are just done *per se* as the only planning that could be done is how to divide topics inside the group. Reduced meetings frequency would happen anyway and students would benefit more from using a collaborative tool (e.g. Slack) then doing a formal meetings, although it must be done periodically to avoid misunderstandings and mistrust.

Therefore, for such situations authors suggest using some kind of hybrid approach, avoiding previously mentioned commonly known mistakes and applying suitable practices and techniques which align with the development process, e.g. iterative releases, continuous deployment, etc. Use of techniques more related to project management should be introduced to students in such way that they can optionally use it starting by an appropriate tool to capture user stories to produce requirements specification followed by the use of Kanban. The purpose of the approach should be that students should conclude by themselves that agile elements improves their efficiency, progress and coordination, and that is not another (from their perspective) useless obligation. Once when they find out the benefits of those agile techniques they would continue to explore other agile techniques which can be supported by a course at master level study when they significantly improve their development skills and fill the gap on the project management side.

Future work would be focused on repeating experiments in the next years in order to get larger sample and possibly eliminate some threats to validity and to measure in which extent this optional approach paired with lab tutorials under staff supervision can provide benefits.

References

Alfonso, M. I., & Botia, A. (2005). An Iterative and Agile Process Model for Teaching Software Engineering. *18th Conference on Software Engineering Education & Training (CSEET'05)*. https://doi.org/10.1109/CSEET.2005.5

Campanelli, A. S., & Parreiras, F. S. (2015). Agile methods tailoring – A systematic literature review. *Journal of Systems and Software*, *110*, 85–100. https://doi.org/10.1016/J.JSS.2015.08.035

Capstone Project Definition - The Glossary of Education Reform. (2016). Retrieved October 5, 2018, from http://edglossary.org/capstone-project/

Čeke, D., & Milašinović, B. (2015). Early effort estimation in web application development. *Journal of Systems and Software*, *103*. https://doi.org/10.1016/j.jss.2015.02.006

Chauhan, M. A., Probst, C. W., & Babar, M. A. (2019). Agile Approaches for Teaching and Learning Software Architecture Design Processes and Methods. In D. Parson & K. MacCallum (Eds.), *Agile and Lean Concepts for Teaching and Learning* (pp. 325–351). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-13-2751-3_16

CollabNet VersionOne 13th Annual State of Agile Report. (2019). Retrieved from https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

F. Tripp, J., & Armstrong, D. J. (2018). Agile Methodologies: Organizational Adoption Motives, Tailoring, and Performance. *Journal of Computer Information Systems*, *58*(2), 170–179. https://doi.org/10.1080/08874417.2016.1220240

Fertalj, K., Milašinović, B., & Nižetić, I. (2013). Problems and Experiences with Student Projects Based on Real-World Problems: A Case Study. *Technics Technologies Education Management*, *8*(1), 176–186.

Freitas Santana, L., Cirqueira, F., Santos, D., Suely, T., Silva, C., Villar, V. B., … Gonçalves, V. (2017). Scrum as a Platform to Manage Students in Projects of Technological Development and Scientific Initiation: A Study Case Realized at UNIT/SE. *Journal of Information Systems Engineering & Management*, *2*(7), 1–7. https://doi.org/10.20897/jisem.201707

Gamble, R. F., & Hale, M. L. (2013). Assessing individual performance in Agile undergraduate software engineering teams. *Proceedings - Frontiers in Education Conference, FIE*, 1678–1684. https://doi.org/10.1109/FIE.2013.6685123

Hurbungs, V., & Nagowah, S. D. (2019). A Practical Approach to Teaching Agile Methodologies and Principles at Tertiary Level Using Student-Centred Activities. In D. Parson & K. MacCallum (Eds.), *Agile and Lean Concepts for Teaching and Learning* (pp. 355–389). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-13-2751-3_17

Judd, M.-M., & Blair, H. C. (2019). Leveraging Agile Methodology to Transform a University Learning and Teaching Unit. In D. Parson & K.

MacCallum (Eds.), *Agile and Lean Concepts for Teaching and Learning* (pp. 171–185). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-13-2751-3_9

Král, J., & Žemlička, M. (2014). Experience with Real-Life Students' Projects, *2*, 827–833. https://doi.org/10.15439/2014F257

Mahnič, V. (2012). A Capstone Course on Agile Software Development Using Scrum. *Ieee Transactions on Education*, *55*(1), 99–106. https://doi.org/10.1109/te.2011.2142311

Mahnič, V. (2015). Scrum in software engineering courses: An outline of the literature. *Global Journal of Engineering Education*, *17*(2), 77–83.

Martin, A., Anslow, C., & Johnson, D. (2017). Teaching Agile Methods to Software Engineering Professionals: 10 Years, 1000 Release Plans. In H. Baumeister, H. Lichter, & M. Riebisch (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 151–166). Cham: Springer International Publishing.

Masood, Z., Hoda, R., & Blincoe, K. (2018). Adapting agile practices in university contexts. *Journal of Systems and Software*, *144*, 501–510. https://doi.org/10.1016/J.JSS.2018.07.011

May, J., York, J., & Lane, M. (2016). Play Ball : Bringing Scrum into the Classroom. *Jourrnal of Information Systems Education*, *27*(2), 87–93.

Meier, A., Kropp, M., & Perellano, G. (2016). Experience report of teaching agile collaboration and values: Agile software development in large student teams. *Proceedings - 2016 IEEE 29th Conference on Software Engineering Education and Training, CSEEandT 2016*, 76–80. https://doi.org/10.1109/CSEET.2016.30

Milašinović, B. (2018). An overview of key aspects in adopting Scrum in teaching process. In *"Cooperation at Academic Informatics Education across Balkan Countries and Beyond" workshop*. Primošten, Croatia.

Milašinović, B., & Fertalj, K. (2018). Issues and Challenges of Adopting Agile Methodologies in Software Engineering Courses. *International Journal of Technology and Engineering Studies*, *4*(5), 197–202. https://doi.org/10.20469/ijtes.4.10004-5

Murphy, C., Sheth, S., & Morton, S. (2017). A Two-Course Sequence of Real Projects for Real Customers. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*, 417–422. https://doi.org/10.1145/3017680.3017742

Olszewska, M., Ostroumov, S., & Olszewski, M. (2017). To agile or not to agile students (with a twist): Experience report from a student project course. *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 83–87. https://doi.org/10.1109/SEAA.2017.54

Orr, A. (2015). Learn By Doing: Agile Project Based Learning in the Software Development Classroom. Retrieved from https://www.linkedin.com/pulse/learn-doing-agile-project-based-learning-software-development-orr/

Rico, D. F., & Sayani, H. H. (2009). Use of Agile Methods in Software Engineering Education. *2009 Agile Conference*, 1–12. https://doi.org/10.1109/AGILE.2009.13

Rodriguez, G., Soria, A., & Campo, M. (2016). Measuring the Impact of Agile Coaching on Students' Performance. *IEEE Transactions on Education*, *59*(3), 202–209. https://doi.org/10.1109/TE.2015.2506624

Rodriguez, G., Soria, Á., & Campo, M. (2015). Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. *Computer Applications in Engineering Education*, *23*(1), 147–156. https://doi.org/10.1002/cae.21588

Scharf, A., & Koch, A. (2013). Scrum in a software engineering course: An in-depth praxis report. *Software Engineering Education Conference, Proceedings*, 159–168. https://doi.org/10.1109/CSEET.2013.6595247