International Journal of Technology and Engineering Studies
volume 4 issue 5 pp. 197-202 doi: https://dx.doi.org/10.20469/ijtes.4.10004-5

Issues and Challenges of Adopting Agile Methodologies in Software Engineering Courses

Boris Milašinović*
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

Krešimir Fertalj
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

Abstract: This paper enumerates some of the possible pitfalls to the successful introduction of agile methodologies in software engineering education. Scrum is the most dominant, or at least the most trending agile methodology. The literature review has been done to find possible solutions to avoid the issues concerning adioption of agile methodologies. This paper elicits common problems, mentioned in literature, expanded with some issues, opinions, and suggestions from the authors' perspective. The findings highlight that the human factor can accelerate learning. However, it can also impede the acquisition of agile values due to lack of knowledge, cultural issues, resistance to change, wrong mindset, and collaboration. Furthermore, some aspects usually do not occur in the real world. Students are usually distracted by some other activities, do not have a typical workplace, causing additional effort when "meeting" in a distributed context. By comparing personal experiences with the literature review, it can be concluded that existing solutions and methods for adopting agile methodologies in software engineering education cannot be cloned as the success rate significantly depends on staff availability and personal and cultural factors. Based on these findings, some valuable suggestions for addressing these issues are enlisted. Educators and educational policy makers could use the findings to enhance the learning and development of engineering students.

Keywords: *Agile, Scrum, software engineering, education*

Received: 13 June 2018; ***Accepted:*** 2 August 2018; ***Published:*** 20 October 2018

I. INTRODUCTION

Happy "XP/Scrum/Agile in education" papers are all alike; every unhappy paper is unhappy in its own way [1]. A paraphrase of Tolstoy's first sentence from Anna Karenina could accurately describe the (not only scientific) papers deal with use of agile methodologies in education. Although all papers have unique features to qualify them as novel, basically all of them have the key aspects successfully implemented, and they conclude with the questionnaire or analysis. This usually shows that students are satisfied with the new approach, and that they acquired the new knowledge required by the software companies.

Anna Karenina principle states that if there is a deficiency in any of key aspects, the family (in general the whole process) will be unhappy. Paraphrasing Tolstoy's sentence is not novel at all, and Anna Karenina principle was used to explain numerous things in various areas, and agile methodology is not an exception [2]. However, it briefly describes problems adopting agile methodologies practically in software engineering education.

This paper tries to enumerate some of the possible pitfalls to successful introduction of agile methodologies in software engineering education, in which Scrum is the most dominant, or at least the most trending agile methodology. Motivation for adopting agile methods in education of software engineering is given in the second section, followed by the review of literature to find the most appropriate place in the curriculum to teach and practice agile methodology. The forth and the fifth

*Correspondence concerning this article should be addressed to Boris Milašinović, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia. E-mail: boris.milasinovic@fer.hr

section deals with students' preparation for the new process and their motivation and attitude. The sixth section deals with problem of role assignments when adopting an agile methodology in teaching process. The seventh section gives an author's context and some personal experiences. The paper concludes with somehow contrary opinions regarding attempts and methods to introduce agile methodology in educational process.

II. NEED FOR CHANGE OF REQUIRED COMPETENCES

Traditional teaching based on theoretical fundamentals supported by hypothetical case studies is not suitable any more in order to produce students for a competitive and changing IT market. Practical experiences in various real-life projects would give students various benefits, e.g., a portfolio to show to potential employers, and also a distinct advantage over those who lack such experience [3]. Involving students into real-world projects and real teamwork environment is of the great importance in software engineering education that is unfortunately sometimes ignored in academic environments. One of the main reasons is complexity of those problems that have to be reduced and partitioned in order to be feasible in academic settings taking care of a size of student team(s), teaching schedule and teaching workload. Educational institutions are often not allowed to change their curriculum due to various accreditation procedures, or in extreme cases due to absence of interest or the knowledge to change [4]. A workaround to introducing new knowledge and new practices is to adapt current courses by introducing real-life problems [5]. However, topics and connections with real-life problems are not the only issues that must be dealt with. As Král and Žemlička summarized in their paper [6], the problems frequently occur in the planning and managing phase rather than in the developing phase, or as a failure of development responsibilities. Consequently, it is obvious that some other skills beyond programming and technical excellence is needed [7] and those soft skills are not always easy to learn or acquire [8].

III. PLACE TO INTRODUCE AGILE METHODOLOGY

The rise of agile methodologies, with Extreme programming (XP) and Scrum as the two most dominant examples, stressed previously mentioned problems and triggered educational changes especially in Software engineering courses. Except an example of introducing some of XP concepts into iterative Unified process and thus creating a hybrid approach [8], most of the literature describes adoption of agile methodology (mostly Scrum in later papers). In their work 9 made a survey of papers describing introduction of agile methods in education from 2003 to 2008, in which they enumerated various issues e.g. lack of training, resistance to changes, problematic teamwork, administrative effort, etc. A similar study, but focused only on Scrum was performed by [10] in 2015. Both studies suggest a capstone project [11] as a place for teaching agile methodology to students and also give a possible course schedule for it [9, 12]. Additionally, some other authors suggest a capstone project as a proper place for teaching agile methodologies. However, there is no unique view on how long it would be and how would it be organized. For example, Scharff et al. organize an annual development project with participants from all the world and provides recommendations how to integrate Scrum in those projects [13]. Murphy et al. [14] propose two course sequences, which can cause organizational problems as described by Burris in his article [15]. As the agile competence pyramid consists of engineering practices, management practices and agile values, Kropp and Meier [16] suggest dividing a course into two parts. The first one should cover engineering practices by XP, and the second one working with Scrum. However, appropriate ecosystem must be created to avoid teaching agile values without being experienced in practice [17].

IV. STUDENT PREPARATION

Giving an answer to the question which methodology to use is not a simple task. Martin et al. [7] organize the so called Agile debates to enable their students to come to the conclusion that there is no silver bullet solution. Thus, the choice of Scrum is not a solution to the problem by itself. It is just good tool to help with the development for which it must be well prepared. Although Scrum as a concept is relatively easy to understand, its adoption and correct usage can be very hard. Studies have shown that even practicing software engineering professionals with university degree or industry experience could not easily adapt to agile methods in a very short intensive time and thus must prepare four weeks in advance [7]. On the other hand, Burris suggest that soft skills can be taught in anticipation of potential problems [15]. Potineni et al [18] suggest that students should first observe existing teams for a week and only then start to gather requirements. Mahnič [12] uses initial zero Sprint as an introduction to Scrum and Freitas Santana et al. [19] spend at least two sprints for students to adapt to Scrum. May et al. [20] suggest a game with a ball in which students must estimate how much time will elapse until all team members exchange a ball under certain rules; showing that in each

iteration estimation gets better, which would result in better estimation of development duration. Mahnič in [10] had also reviewed some other approaches using games (e.g. planning poker, LEGO bricks) as an alternative to practical work in case there is not enough time or skills for development.

V. STUDENTS MOTIVATION, ATTITUDE AND PROBLEMS

Perception about change process can be diverse [21], e.g. students could be very enthusiastic about extreme programming practices [22], and sometimes feel that things like project management are not important, or are applied just for lecturer's sake [17], or tend to follow waterfall-like plan rather that respond to change [23]. Mahnic states that better students are more aware of the benefits [10].

Human factor can accelerate, but it can also impede the acquisition of agile values in companies due to lack of knowledge, cultural issues, resistance to change, wrong mind set and lack collaboration [21]. In the same way this human factor can affect agile adoption in teaching process. Strong temperament and lack of interpersonal skills can undermine team effort [24]. Sometimes even excellent coders can cause problems by underestimating the importance of soft skills, or by having a bad attitude to technically less proficient users. Consequently they do not adapt well to the team [6].

Furthermore, there are some aspects that usually do not occur in real-world. Students are usually distracted with some other activities [4], do not have a common working place causing additional effort when "meeting" in distributed context [23]. This may even lead to misunderstanding, or even mistrust between project members and staff [25]. Thus Olszewska et al. [26] organized "daily" meetings every third day, and Freitas Santana et al. [19] organized meetings every fifteen days (but their project was planned with two years duration in mind). However, a student is more likely to quit the project (fail the course), perhaps more often than an employee resigns. Due to afore mentioned factors and the possibility of sickness, Olszewska et al. [26] assigns double responsibility to tasks. Another significant aspect is lack of resources combined with large class, especially tutors who can prevent successful transfer of knowledge [25].

Additionally, there is a problem of motivation. Students' only "salary" is their course grade, thus many students are only interested in grades and deadlines than the software quality. In order to change this attitude Murphy [14] suggest that in the second course of two courses sequence a student must continue working on other work, thus raising awareness of importance of a good code.

As in any team work, another significant problem is grading. Individual work must be recognized and valued appropriately [5]. However, working in an Agile team can mask individual contribution and it needs another type of grading framework [27]. Another approach is to user peer review to decide what percentage of overall score is owed to each student (Burris in [15]).

VI. ROLE DISTRIBUTION PROBLEM

One important problem is the distribution of roles, the most problematic being who should be Scrum master. As Mahnič has shown in his review [10] there are two different opinions about who should that be the teacher or a student. Advocates of students as Scrum masters state that students are able to work independently once they adopt Scrum, and that otherwise would feel that they are micro managed and not able to self-organize [28]. Some force rotating Scrum master role with a professor that teaches/trains them Scrum [17, 23]. Somewhere in the middle are those who use research assistants [29] or students that previously passed the course [14] as project managers or coordinators. Additional roles are not unusual, as many include Agile coach role taken by lecturer [17] that is usually not a member of the team [23]. A similar dilemma is present for the Product owner and is usually solved in the same manner as Scrum master role problem. Those who oppose students as Scrum master or Product owner usually state that students do not have a good overview of domain problem, and have a tendency to discard difficult problems [14] or that they could affect quality of user stories [10]. However, the lecturer should have experience in order to guide students, letting them learn from their mistakes, rather than showing them the solution to the problem [23]. Moreover, it would be beneficial if the lecturer owns a Scrum certificate [20].

VII. AUTHORS' CONTEXT

The authors have successfully emulated real-word projects in the past ten years at the 3rd year of bachelor study in the course Development of Software. In the course students have to extract requirements from an interview in which a real user is emulated and develop an application implementing user's requirements [5]. By the time students enrol on the courses they should pass (among other courses) introductory programming course, Object oriented programming, Algorithms and Data Structures, Databases, indicating that they should be at least formally capable of doing advanced things. Hence, this makes the course a good candidate for addition of agile elements, although teaching agile methodology is not mandatory or described in outcomes of the courses.

However, experiences with attempts to introduce elements of agile methodologies were somehow contrary to results shown in reviewed scientific papers. Maybe one of the reason was a too optimistic answer to the first question from [30]: "Is the student ready for this level of independence?". In none of the previous courses had students learned how to program mobile, web or standalone application. This lead to the paradox similar to [15] that they have to learn how to manage development process, but they do not have the development skills. Neither is something novel. For instance [28] requires that students use programming language that has not been officially used in previous courses, what perhaps looks like a too harsh measure and definitely does not help in estimating the duration of work.

The first attempt of introducing agile elements in the course was to create an assignment to enter all user stories. There were several pitfalls with this approach. The first one was that all groups did the same project and only their implementation was different, thus making user stories ideal for replication among groups. The second one was that the progress of development was heavily constrained by teaching development process (teach them how to develop) and that choosing what to do in which iteration/sprint was not the students' choice but the side effect of the learning progress. Perhaps because of this, students were subdividing user stories in tasks after the development had been done, just to earn points for the final grade and not to help them in whole process. Contrary to approach in [29], where the Scrum master and Product owner should not do programming tasks, the intention was that each student does every stage of a software lifecycle so everyone felt that entering stories and tasks were unnecessary additions to the development process, as was seen in the complains in course feedback questionnaire. The other remarks in questionnaire was similar to other problems mentioned in literature: project quitting, delays due to sicknesses of team members, distraction with other obligations, role problems, meetings problem etc. The third problem was lack of staff to fulfill of tasks: being teacher, product owner and agile coach at the same time.

VIII. CONCLUSION AND RECOMMENDATIONS

By comparing personal experiences with the literature review it can be concluded that others' solutions and methods for adopting agile methodologies in software engineering education cannot be cloned as the success rate for significantly depends on staff availability and because students attitude varies by country or part of the world. Even more, there is no unique opinion on many aspects, e.g. how to distribute roles.

There are some paradoxes and inevitable problems with the fact that students are not comparable to regular employees and have different attitude and approach to work. Also, teaching large number of students all aspects of a software lifecycle by emulating teamwork cannot be equal to real life situation, as in real life teams roles are usually strictly divided, and the process cannot be emulated completely.

Lastly, the general dilemma refers to what is more useful for the bachelor student to teach them about methodologies although they are not experienced in development, or to teach them software development (if possible on real life projects) and wait for more suitable courses in master study where they should learn more about methodology. The problem of introducing some things too early is that students are not aware of the benefits as they did not experienced problems that are solved with the approach. However, the introduction of agile elements is inevitable, but instead of using a particular methodology authors suggest to use some kind of hybrid approach taking suitable elements (e.g., weekly meetings, iterative releases, user stories etc.) that best fits own environment, avoiding previously mentioned commonly known mistakes.

Declaration of Conflicting Interests

It is an original work of authors who declare that there are no conflicts of interest.

REFERENCES

[1] B. Milašinović, "An overview of key aspects in adopting Scrum in teaching process," in *Workshop of Cooperation at Academic Informatics Education across Balkan Countries and Beyond*, Primošten, Croatia, 2018.

[2] M. Breyter, "Practical guide to scaling agile," 2015. [Online]. Available: https://bit.ly/2VhpAS6

[3] A. Orr, "Learn by doing: Agile project based learning in the software development classroom," 2015. [Online]. Available: https://bit.ly/2Su13az

[4] U. K. Kudikyala and U. N. Dulhare, "Using Scrum and Wikis to manage student major projects," in *IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, Amritsar, India, 2015. doi: https://doi.org/10.1109/mite.2015.7375279

[5] K. Fertalj, B. Milašinović, and I. N. Kosović, "Problems and experiences with student projects based on real-world problems: A case study," *Technics*

Technologies Education Management, vol. 8, no. 1, pp. 176–186, 2013.

[6] J. Král and M. Žemlička, "Experience with real-life students' projects," in *Federated Conference on Computer Science and Information Systems*, Warsaw, Poland, 2014. doi: https://doi.org/10.15439/2014f257 pp. 827–833.

[7] A. Martin, C. Anslow, and D. Johnson, "Teaching agile methods to software engineering professionals: 10 years, 1000 release plans," in *International Conference on Agile Software Development*, Cologne, Germany, 2017. doi: https://doi.org/10.1007/978-3-319-57633-6_10

[8] M. I. Alfonso and A. Botia, "An iterative and agile process model for teaching software engineering," in *18th Conference on Software Engineering Education & Training (CSEET)*, Ottawa, Canada, 2005. doi: https://doi.org/10.1109/cseet.2005.5

[9] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in *Agile Conference*, Chicago, IL, 2009. doi: https://doi.org/10.1109/agile.2009.13

[10] V. Mahnič, "Scrum in software engineering courses: an outline of the literature," *Global Journal of Engineering Education*, vol. 17, no. 2, pp. 77–83, 2015.

[11] Great Schools Partnership, "Capstone project definition - The glossary of education reform," 2016. [Online]. Available: https://bit.ly/2T5tL6U

[12] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99–106, 2012. doi: https://doi.org/10.1109/te.2011.2142311

[13] C. Scharff, S. Heng, and V. Kulkarni, "On the difficulties for students to adhere to scrum on global software development projects: Preliminary results," in *Proceedings of the Second International Workshop on Collaborative Teaching of Globally Distributed Software Development*, Piscataway, NJ, 2012. doi: https://doi.org/10.1109/ctgdsd.2012.6226946

[14] C. Murphy, S. Sheth, and S. Morton, "A two-course sequence of real projects for real customers." in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle, WA: ACM, 2017. doi: https://doi.org/10.1145/3017680.3017742 pp. 417–422.

[15] K. A. Alshare, D. Sanders, E. Burris, and S. Sigman, "How do we manage student projects?: Panel discussion," *Journal of Computing Sciences in Colleges*, vol. 22, no. 4, pp. 29–31, 2007.

[16] M. Kropp and A. Meier, "Teaching agile software development at university level: Values, management, and craftsmanship," in *26th International Conference on Software Engineering Education and Training (CSEE&T)*, San Francisco, CA, 2013. doi: https://doi.org/10.1109/cseet.2013.6595249

[17] A. Meier, M. Kropp, and G. Perellano, "Experience report of teaching agile collaboration and values: agile software development in large student teams," in *IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, Dallas, Texas, 2016. doi: https://doi.org/10.1109/cseet.2016.30

[18] S. Potineni, S. K. Bansal, and A. Amresh, "ScrumTutor: A web-based interactive tutorial for Scrum Software development," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Mysore, India, 2013. doi: https://doi.org/10.1109/icacci.2013.6637469

[19] L. F. Santana, L. F. C. d. Santos, T. S. C. Silva, V. B. Villar, F. G. Rocha, and V. Gonçalves, "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*, vol. 2, no. 2, pp. 1–7, 2017. doi: https://doi.org/10.20897/jisem.201707

[20] J. May, J. York, and D. Lending, "Teaching tip: Play ball: Bringing scrum into the classroom," *Journal of Information Systems Education*, vol. 27, no. 2, pp. 87–92, 2016.

[21] T. J. Gandomani, H. Zulzalil, A. A. Ghani, A. B. M. Sultan, and K. Y. Sharif, "How human aspects impress agile software development transition and adoption," *International Journal of Software Engineering and its Applications*, vol. 8, no. 1, pp. 129–148, 2014. doi: https://doi.org/10.14257/ijseia.2014.8.1.12

[22] G. Melnik and F. Maurer, "Perceptions of agile practices: A student survey," in *Conference on Extreme Programming and Agile Methods*. Springer, 2002. doi: https://doi.org/10.1007/3-540-45672-4_27 pp. 241–250.

[23] G. Rodríguez, Á. Soria, and M. Campo, "Measuring the impact of agile coaching on students performance," *IEEE Transactions on Education*, vol. 59, no. 3, pp. 202–209, 2016.

[24] M. Villavicencio, E. Narváez, E. Izquierdo, and J. Pincay, "Learning scrum by doing real-life projects," in *IEEE Global Engineering Education Conference (EDUCON)*, Athens, Greece, 2017.

[25] G. Rodriguez, Á. Soria, and M. Campo, "Virtual scrum: A teaching aid to introduce undergraduate

software engineering students to scrum," *Computer Applications in Engineering Education*, vol. 23, no. 1, pp. 147–156, 2015. doi: https://doi.org/10.1002/cae.21588

[26] M. Olszewska, S. Ostroumov, and M. Olszewski, "To agile or not to agile students (with a twist): Experience report from a student project course." in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Vienna, Austria: IEEE, 2017.

[27] R. F. Gamble and M. L. Hale, "Assessing individual performance in agile undergraduate software engineering teams," in *IEEE Frontiers in Education Conference (FIE)*, Oklahoma, OK, 2013. doi: https://doi.org/10.1109/fie.2013.6685123

[28] R. T. Hans, "Work in progress: The impact of the student scrum master on quality and delivery time on students' projects," in *2017 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, Hong Kong, 2017. doi: https://doi.org/10.1109/latice.2017.22

[29] A. Scharf and A. Koch, "Scrum in a software engineering course: An in-depth praxis report," in *26th International Conference on Software Engineering Education and Training (CSEE&T)*, San Francisco, CA, 2013.

[30] Carnegie Mellon University, "Supervising independent student projects," 2011. [Online]. Available: https://bit.ly/2BQuEWb