Hybrid Agile Approach in Software Engineering Education – A Case Study

M. Kaluza, S. Candrlic and M. Asenbrener Katic
Faculty of Informatics and Digital Technologies, Rijeka, Croatia
marina.kalu5a@gmail.com, sanjac@inf.uniri.hr, masenbrener@inf.uniri.hr

***Abstract* - By definition, software engineering is a systematic approach to software development. However, to succeed in today's dynamic business environment, development teams must adapt and respond to change quickly. To prepare students for their role in real-world software development teams of which they will soon be a part, teachers attempt to create as realistic a project environment as possible. The software development education model presented in this paper is based on the hybrid agile approach. It combines both the agile approach, which is suitable for dealing with changes in requirements, and the planning-oriented, systematic approach, which is traditionally used in software engineering. In addition, the students' attitude towards this hybrid agile model is investigated.**

Key words – hybrid agile, agile, software engineering, education

I. INTRODUCTION

Higher education courses are prone to many changes over time as the industries and jobs, for which universities prepare students evolve and change. Software engineering is one of the fundamental courses of study programs in the field of informatics, computer science and engineering. Software engineering is a systematic approach to software development in which the software development process is viewed as a series of phases and steps [1], [2].

Traditional development approaches are focused on predefined phases. Requirements are fully specified before coding begins. This approach gives large companies a structure for monitoring, and simplifies the understanding and scheduling of tasks. Managing large teams of developers is easier because the end goal and all requirements are specified before the development phase. Some of the drawbacks include lack of adaptability, delayed testing, and delayed product release. The lack of customer involvement in development can lead to dissatisfied customers who have waited too long for a product that ends up not providing everything the customer wants. One of the most commonly cited traditional models is the waterfall model [1], [2], [3].

The Waterfall model consists of seven development phases: requirements analysis, design, coding and implementation, testing, operation and deployment, and maintenance. Each of these phases must be fully completed before moving to the next phase. In the first three phases, all planning is done, and only when all coding and testing is completed, can the project be deployed. The last phase does not end, because perfecting the product and improving of its functionalities will continue as long as the product is used [3].

With the publication of the agile manifesto, a rapid development of many different agile methods began. Agile approaches were created to overcome the shortcomings of traditional approaches. The agile approach focuses on tasks, users and team members, rather than documentation, and it focuses more on responding to change rather than following a fixed plan. Development in agile approaches is carried out through iterations. Unlike traditional approaches, user involvement and feedback are crucial in the development process. Some advantages of agile approaches are easier adaptability to changes, quick user feedback, and earlier product implementation. On the other hand, agile approaches lack documentation, which can be a problem when new members join the project. It is more difficult to measure progress because the end result is not completely known. Short iterations can also lead to avoiding the development of some large features due to their complexity and not leaving enough time to design ideas, which can lead to later changes due to user dissatisfaction [4], [5], [6].

Scrum is the best-known agile methodology. In Scrum there are three different roles: Scrum team, Scrum master and product owner. The Scrum team is a team of different experts who work together to develop the product. The Scrum master is a member of the team who makes decisions when there are disagreements within the team. The product owner provides feedback to the user. The Scrum methodology is based on development in iterations, sprints. At the end of each sprint, a new product increment is achieved. A Scrum sprint lasts from at least a week to a month and consists of brief planning, development of selected tasks from the backlog, testing, and at the end a sprint review is conducted. Progress is presented to the project owner in the review meeting. Development is monitored through daily scrums when developers discuss what smaller tasks should be completed that day. Scrum promotes good communication between team members and ensures regular communication with the product owner, which is why it is possible to reduce costs, since in each development step only what the project owner needs is done, and all members are well acquainted with the requirements and needs of the project [7].

In addition to traditional methodologies and the agile approach, companies today also use a hybrid agile approach. This approach combines the advantages of traditional and agile methodologies to produce a methodology which can be used with minimal customization [8], [9]. Hybrid approach is usually based on waterfall model and some agile methods. The focus is on the need for an initial plan, which is not as extensive as in the waterfall model, but consists of initial documentation, time and cost estimates and main goals. In the next step, development is carried out according to an agile model. For example, if the Scrum methodology is used, the sprint iteration begins in this step [9], [10]. Since the main development is done using an agile approach, this approach to software development can also be called a hybrid agile approach, which is the focus of this paper.

The paper describes the implementation of the hybrid agile approach in a software engineering course and the evaluation by the students. After the introduction, the second chapter presents related work on the implementation of the hybrid agile approach in industry and software engineering courses. The methodology that covers course structure and the evaluation of the implemented approach is described in the third chapter. The fourth chapter presents the survey results and their discussion. Finally, the conclusions and plans for future work are presented.

II. Related work

Many different studies have been conducted on the implementation of traditional and agile methods in work and educational environments. The main focus of this paper is on the hybrid agile approach.

A. Development methodologies used in work environments

Previous research shows that companies mostly use different hybrid approaches for software development [11], [12]. The authors in [11] explored which methodologies are used in different organizations. Out of 6 respondents, 5 use a hybrid development model, consisting of the waterfall model and the Scrum methodology, while only one respondent uses a pure agile approach with the XP methodology. Based on this, it can be concluded that in the organizational environment, when developing complex projects, it is usually not possible to exclude documentation, which is emphasized in traditional methodologies but neglected in agile methodologies. The authors in [12] collected survey data on global software development (GSD), which is characterized by distributed and large-scale development and concluded that GSD needed to move away from the use of traditional approaches, resulting in companies primarily using some type of hybrid approach to development. They also noted that pure agile approaches are somewhat rare in GSD, which is consistent with [11].

The authors in [13] discuss adoption of a hybrid development model in companies outside the IT industries. The implementation of the hybrid methodology was conducted under the supervision and guidance of three agile methodology coaches who participated in this study. Companies that moved to a hybrid model experienced several benefits of implementing some agile practices in development, such as creative contributions from individuals, as well as better communication between different teams and more transparency within the team.

When companies hire new employees, they want applicants to have extensive knowledge that universities have difficulty providing [14]. The authors in [14] noted that because expectations are too high, companies are forced to hire individuals who lack soft skills, which proves to be a greater disadvantage than less knowledge. Software engineering education needs to focus more on computer science and the current tools used by the companies that will employ the students upon their completion of higher education. On the other hand, companies need to hire people who are ready for teamwork even if they have less knowledge than other applicants.

B. Implementation of agile approaches in education

The authors in [15] conducted a comprehensive study of software engineering education papers that address software engineering trends. The most common trend used in education was agile software development. The authors of the paper found that, while the trend of implementing agile approaches in education is the most popular, it comes with its own challenges. This paper provides guidelines for practitioners, researchers, and educators.

The following papers describe how hybrid agile practices have been implemented in the higher education environment. The most commonly cited agile methodology in software engineering education is Scrum [16-20], which is not surprising since Scrum is often used in industry. The following papers show the implementation of agile approaches in higher education courses.

The authors in [16] presented a course in which a project for a real user was carried out. A team of 6 students participated in this course. Students were assigned roles and each was responsible for a different aspect of the project. The main problem was that the students had to learn about the technologies and procedures and apply them at the same time applying them. In addition, the students had to be encouraged to communicate with the users, so that the final product met user expectations. The authors concluded that not all agile practices are appropriate for the educational environment, but that they should still be used. Another conclusion is that the project needs to be designed simply enough that it can be done in a single course and students have enough time for other courses during the semester.

The authors in [17] did not face the problems mentioned in [16] that the students did not know fundamentals and technologies because their course consisted of a 3-week fast-track theoretical course and a 10-week product development course. Each team had the same assignment and was given a user with whom they had to analyze and specify product requirements. Students could choose which agile methodology and practices they wanted to use. The course was deemed successful as all teams were able to successfully produce a final product. It was found that weekly meetings and user involvement played the biggest role in project success as students were warned if they were going in the wrong direction or taught how to adopt more agile practices.

The authors in [18] had also introduced agile development in the form of 1-week and in later years 2-week assignments for teams. The teams worked on all assignments separately, so the final results were different. Students confirmed that this type of project in a course gave them a better understanding of software development.

In contrast to the before mentioned sources, the following initially implemented traditional methodologies and then gradually implemented Scrum [19] or completely switched to Scrum [20]. Both papers found that the introduction of Scrum increased student interest. The authors in [19] initially taught students a traditional approach to software analysis. The agile approach was introduced gradually so that students could become more familiar with agile practices. Daily meetings encouraged students to actively participate. With this approach, student satisfaction in learning and developing software was observed as they developed a project that had a value to the end user. The authors in [20] faced the problem of students' working at the last minute when using traditional methodology which led them to switch to agile methodology. The introduction of Scrum in education was done in a controlled environment where students were given detailed instructions for the goals of each sprint. The authors also wanted to prevent non-participation in the team, so team work was eliminated and assessments were introduced at the end of each sprint. To promote the team aspect of agile methodologies, a platform was introduced where students could communicate, ask questions, and help each other. The conclusion is that independent projects like this, with the introduction of some aspects of agile approaches, can teach students about agile paradigms and development.

III. METHODOLOGY

The Software engineering course is one of the fundamental courses of the Graduate University Study Program Informatics at the Faculty of Informatics and Digital Technologies, University of Rijeka. The course was offered in the winter semester with 2 hours of lecture and 2 hours of practical work per week. The workload for the course is 6 ECTS credits. The course is taught by two teachers: a professor and a teaching assistant.

The aim of the course is to acquire knowledge in the field of software engineering and covers software development phases: requirements analysis, project development, team software development and software testing. It covers the application of both agile and traditional methods, techniques, and approaches that help with planning, team organization, and task management during software development within a specific timeframes and resources.

The course includes the following topics: Models of software development; Traditional, agile, and hybrid approaches to software development; Methods and techniques used in different phases of software development; Team management; Analysis and management of user requirements; Estimation of resources for software development; Risk management; Software design and architecture; Implementation; Construction of program code in collaboration; Refactoring; Testing; Version management; Software documentation; Professional responsibilities of software engineers; Software re-engineering.

Course outcomes are listed below. It is expected that upon completion of all course assignments, students will be able to:

- O1. Distinguish basic concepts, methods, techniques, and approaches in the field of software engineering, particularly as they relate to traditional and agile approaches.
- O2. Develop models of a system based on an analysis of user requirements and market needs in a given domain.
- O3. Estimate the resources required to build the software.
- O4. Plan software development considering the various roles of development team members and users in a software development team project.
- O5. Based on the analysis performed and the project created, create the software in the chosen development environment and prepare its documentation.
- O6. Perform tests based on the planned test cases and document the test results.

Course content is delivered through project-based learning, so that assignments simulate real-life situations.

The activities used to assess the acquisition of learning outcomes are a written exam (max. 30 points), project meetings (0-20 points for active participation, preparation, and proposed solutions) and software development using a hybrid agile approach (0-50 points).

At the beginning of the semester, students were divided into five teams of four. Each team member was assigned a primary role (project manager, designer, programmer, and tester), with responsibilities for the work performed. The teachers presented two main topics/themes that were used as global user requirements. Staring from these, each team developed its project idea. Each topic was selected by at least two project teams competing to develop better software for that topic.

The project began with a software requirements analysis to derive the key requirements for software development. Each team chose its own development framework. For each project meeting, the teams were given several tasks and deliverables to complete. Each meeting focused on a different deliverable, e.g., data model, mockup, test cases and scenarios, program logic and flow, etc. During the meetings, students presented what they had done so far, commented on their solutions to the tasks, and set new tasks and plans for the next project meeting. There was a total of 4 project meetings and a final presentation of the software created by the end of the semester. A combination of traditional face-to-face classes and online classes supported by a learning management system (Moodle LMS) was used. Figure 1 shows the main activities in the course.

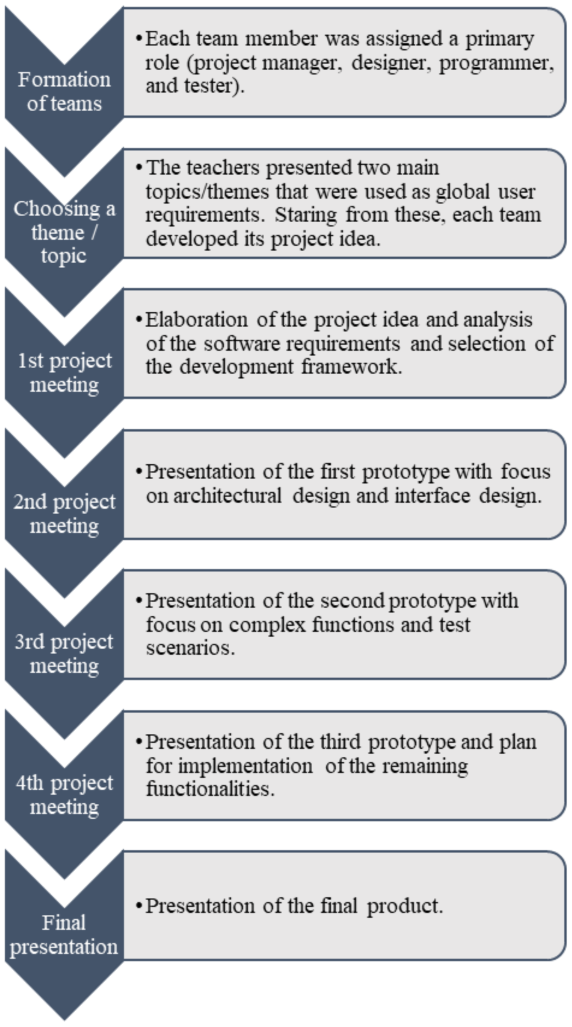
Formation of teams
• Each team member was assigned a primary role (project manager, designer, programmer, and tester).
Choosing a theme / topic
• The teachers presented two main topics/themes that were used as global user requirements. Staring from these, each team developed its project idea.
1st project meeting
• Elaboration of the project idea and analysis of the software requirements and selection of the development framework.
2nd project meeting
• Presentation of the first prototype with focus on architectural design and interface design.
3rd project meeting
• Presentation of the second prototype with focus on complex functions and test scenarios.
4th project meeting
• Presentation of the third prototype and plan for implementation of the remaining functionalities.
Final presentation
• Presentation of the final product.

Figure 1. Activities in the course

After completing all teaching activities, students were asked to participate in a survey. The questionnaire was created using Google Forms. It consisted of questions with predetermined answers (yes/no), questions with a Likert scale of 1 to 5, where 1 was “strongly disagree” and 5 was “strongly agree”, and open-ended questions. The survey was sent to all 20 students taking the “Software Engineering” course in 2020/2021, and all completed the survey. The Likert scale questions are shown in Table I, and the responses to the remaining questions are discussed in the text, both of which are listed in the next chapter.

IV. SURVEY RESULTS AND DISCUSSION

Students on the teams were assigned roles (project manager, architecture designer, programmer, and tester), and most (95%) were satisfied with the role they chose. When asked if they would prefer to perform all activities equally, without being responsible for only one group of activities (e.g., as a tester for all testing activities), they had mixed opinions. 60% of students thought independent, separate roles were a better option, while 40% would prefer to participate equally in all activities, without being a separate person responsible for each activity. The latter group explained that with independent roles, team members are not equally motivated to participate and do not gain the same knowledge during the project. When asked how students should choose a role on the team, most agreed that everyone should try to expand their knowledge. However, when asked how this could be achieved, they had different ideas. Some felt that everyone should choose a role where they know more (60%) and then help the rest of the team learn, a few students felt it was better to choose a role where they are less experienced (15%) so that they can learn through the project, and others (25%) could not decide between the two because they felt that all students should participate in every aspect of the project.

Most students (75% strongly agreed and 10% agreed) felt that it was easier to work in a team with friends, i.e., they favoured independent division into teams rather than random assignment to a team. Opinions were divided on the question of whether the choice of the main project topic should be predetermined by the teacher: 35% of the students disagreed with this statement, 35% agreed, and 30% did not know.

Almost all agree (45% strongly agree, 45% agree) that the project meetings are a good way to evaluate progress. They think that it was good that the tasks for the project meetings were set 3 weeks in advance. One of the questions on which there was no unanimous opinion was: is it better to set the content and required documents of the meeting in advance, or should the meeting be flexible without full planning in advance. The general opinion is that some level of meeting organization is necessary to promote progress, but that too much documentation should be avoided to be consistent with the spirit of agile development. Some students suggested writing tasks on a Kanban board each week.

When asked if everyone participated equally, they were divided. Consistent with the previous question, students were asked how they would describe their contribution (Figure 2). Almost half of the students (45%) felt that their contribution was equal to that of the rest of the team, 35% and 20% felt that they had contributed less and more, respectively, to the final solution than the rest of the team.

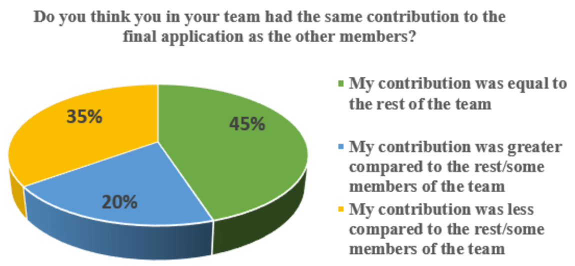

Figure 2. Contribution in team

TABLE I. QUESTIONS WITH LIKERT SCALE ANSWERS

Question	**1 (%)**	**2 (%)**	**3 (%)**	**4 (%)**	**5 (%)**
Having roles in the team organization is a good way for all members to acquire equal knowledge	5	20	50	25	0
I am satisfied that we could chose our team	5	0	10	10	75
It is easier to work in a team with friends	0	10	15	10	65
Teacher should assing students into teams	25	50	10	5	10
In our team, the responsibilities were shared - everyone was responsible for the task they undertook (or were assigned)	0	20	20	35	25
The teacher's instructions for a main project topic (general user request) were sufficient to develop the project idea	0	0	15	20	65
The fact that we were competing with another team on the same topic motivated me to work harder and better	5	25	15	30	25
I would prefer if each team had a different topic as a main project topic, instead of having team competition	25	45	15	10	5
I would prefer each team to define on their own main project topic and idea	10	25	30	10	25
Project meetings are a good way to evaluate the progress of the project	0	0	10	45	45
Project meetings are a good way to evaluate the contribution of each individual team member	0	10	25	40	25
For a successful participation in a team (such as during this project), all team members should have the same level of knowledge	15	40	25	15	5
For a successful participation in a team (such as during this project), team members should have complementary levels of knowledge	0	10	5	50	35
It is good that tasks for each project meeting are set in advance (3 weeks before the meeting)	0	0	5	35	60
I would prefer that all tasks for all the meetings are known at the beginning of the semester	15	35	10	15	25
I would prefer that instead of having pre-defined content and required documents for each meeting, problems faced by the team and the functionalities to be included in the next version are discussed freely during the meeting, without planning it in advance	10	35	15	30	10
The assessment done during meetings encouraged me to actively participate in all project meetings and prepare the required deliverables of the project activities (documentation).	0	0	15	55	30
All members of my team participated equally actively in the project	5	25	20	40	10
Having a prototype required for one of the project meetings motivated me to work on the project.	0	5	0	75	20
I would prefer teachers to evaluate only the final product, without the checkpoints that were introduced during the project meetings	35	45	10	5	5
I believe that this project gave mean understanding of the problems and insight to real team software development.	5	5	10	55	25
In general, I think the experience of working on this project was useful.	0	10	0	30	60

It is interesting to note that only the students who held the roles of architecture designers and testers felt that their contribution to the final solution was less than the contribution of the other team members. The role of programmer was considered the most challenging by more than half of the students (55%), 25% considered the role of project manager the most challenging, and 20% considered all roles equally challenging, with no focus on the roles of architecture designer and tester (Figure 3).

When asked which approach, they thought should be the first choice for the software engineering course project, half of the students answered that it was a hybrid agile approach, the other half chose the pure agile approach, while no one chose the traditional approach. Even though they do not have much practical experience to judge this issue, the students’ opinion was based on the theoretical ground they acquired in this this course.

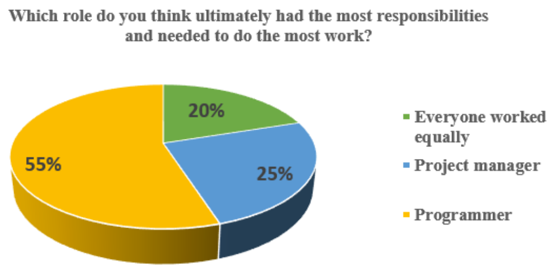

Figure 3. Hardest role

The students' opinions on the questions about the course organization and the project are shown in Table I. A Likert scale of 1 to 5 was used, with 1 representing “strongly disagree” and 5 representing “strongly agree”.

Students generally found this project to be a useful experience, as all topics were adequately supported by theoretical classes and they were satisfied that it was possible to choose the development framework in which to develop the application. Suggestions for improvement included the introduction of "user" feedback to bring the agile approach closer. It was also suggested to introduce the Scrum methodology and define the objectives until the next meeting at the end of the previous meeting. In addition, it was suggested to introduce individual evaluation in the teams, as some students felt that not everyone contributed equally to the final product.

V. CONCLUSION

The survey results confirm some of the conclusions drawn in previous research on agile and hybrid agile methods in teaching. It was found that continuous monitoring of project progress is necessary, while evaluation of only the final product is rejected. Students believe it would be useful to introduce as many features of agile development as possible, less structured meetings and the introduction of some form of feedback immediately after the meeting.

However, as shown in the survey results, many students think that it is beneficial to have roles in the team organization, which was also the case in [16]. Another opinion, as well as the conclusions from [17-19], is that roles should not be used to avoid the possibility of only 1 or 2 students can complete the whole project alone. However, it would be useful to improve the organization of roles or find another way to evaluate individual work, since half of the students felt that not everyone contributed equally to the project, and that some roles had more tasks and responsibilities at the end. The authors suggest dividing work by tasks rather than roles, which is more in line with the spirit of the agile approach. This would prepare the ground for all students to contribute equally to the development process through different types of tasks.

The competitive nature of the course organization was very motivating for the students. As in [16-20], a predefined project topic and implementation method (using specific tools) was considered a good way to go. However, unlike [16], [18-20], in the implementation of this project it was possible to choose a development tool that the students were satisfied with.

In our future work, we plan to improve the presented hybrid agile model. To make the agile approach more prevalent, this model will introduce a work management tool. This will allow teachers to better monitor the work done by students, as well as their individual contributions. Our future research will also focus on methods to assess individual contribution in software development team projects.

Acknowledgements

This work has been fully supported by the University of Rijeka under projects uniri-drustv-18-73 and uniri-drustv-18-140.

References

[1] KPI Partners, "Traditional vs. Agile Software Development Methodologies", 2023. Available: https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies [Accessed:15-Jan-2023]

[2] Geeksforgeeks, "Difference between Traditional and Agile Software Development", 2023. Available: https://www.geeksforgeeks.org/difference-between-traditional-and-agile-software-development/ [Accessed:15-Jan-2023]

[3] B. Lutkevich, "Waterfall Model", Available: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model [Accessed:18-March-2023]

[4] S. Al-Saqqa, S. Sawalha, & H. AbdelNabi, Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11), 2020.

[5] A. M. Gheorghe, I. D. Gheorghe, I. L. & Iatan, Agile Software Development. Informatica Economica, 24(2), 2020.

[6] Active Collab, "Advatages and Disadvatages of Agile Project Management", 2017, Available: https://activecollab.com/blog/project-management/agile-project-management-advantages-disadvantages [Accessed:18-March-2023]

[7] A. Srivastava, S. Bhardwaj and S. Saraswat, "SCRUM model for agile methodology," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017, pp. 864-869, doi: 10.1109/CCAA.2017.8229928.

[8] J. Noll, & S. Beecham, How agile is hybrid agile? an analysis of the helena data. In Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20 (pp. 341-349).

[9] M. A. Jabar, S. Abdullah, Y. Y. Jusoh, S. Mohanarajah and N. M. Ali, "Adaptive and Dynamic Characteristics in Hybrid Agile Management Model for Software Development Project Success," 2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS), Johor Bahru, Malaysia, 2019, pp. 1-5, doi: 10.1109/ICRIIS48246.2019.9073337.

[10] N. Smits, "A Hybrid Software Development Method", 2022 Available: https://www.devfacto.com/blog/a-hybrid-software-development-method [Accessed:18-March-2023]

[11] N. Yahya, & S. S. Maidin, The Waterfall Model with Agile Scrum as the Hybrid Agile Model for the Software Engineering Team. In *10th International Conference on Cyber and IT Service Management (CITSM)*, 2022, pp. 1-5.

[12] M. Marinho, J. Noll, I. Richardson & S. Beecham, Plan-driven approaches are alive and kicking in agile global software development. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2019, pp. 1-11.

[13] F. P. Zasa, A. Patrucco & E. Pellizzoni, Managing the hybrid organization: How can agile and traditional project management coexist?. *Research-Technology Management*, *64*(1), 2020, pp. 54-63.

[14] V. Garousi, G. Giray, E. Tuzun, C. Catal and M. Felderer, "Closing the Gap Between Software Engineering Education and Industrial Needs," in *IEEE Software*, vol. 37, no. 2, pp. 68-77, March-April 2020, doi: 10.1109/MS.2018.2880823.

[15] O. Cico, L. Jaccheri, A. Nguyen-Duc & H. Zhang, Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. *Journal of Systems and Software*, *172*, 110736, 2021.

[16] M. Olszewska, S. Ostroumov and M. Olszewski, "To Agile or not to Agile Students (With a Twist): Experience Report from a Student Project Course, *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2017, pp. 83-87, doi: 10.1109/SEAA.2017.54.

[17] D. F. Rico and H. H. Sayani, "Use of Agile Methods in Software Engineering Education," *2009 Agile Conference*, 2009, pp. 174-179, doi: 10.1109/AGILE.2009.13.

[18] K. Fertalj, B. Milasinovic, B., & I. Nizetic, Problems and experiences with student projects based on real-world problems: a case study. *Technics Technologies Education Management*, *8*(1), 2013., pp.176-186.

[19] B. Bruegge, M. Reiss and J. Schiller, "Agile Principles in Academic Education: A Case Study," *2009 Sixth International Conference on Information Technology: New Generations*, 2009, pp. 1684-1686, doi: 10.1109/ITNG.2009.76.

[20] M. Madeja and M. Biňas, "Implementation of SCRUM Methodology in Programming Courses," *2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2018, pp. 333-340, doi: 10.1109/ICETA.2018.8572161.