APPLICATION OF AGILE SOFTWARE DEVELOPMENT PRACTICES IN SOFTWARE ENGINEERING EDUCATION

MERT AKKANAT

Master's Thesis

Graduate School

Izmir University of Economics

İzmir

2022

APPLICATION OF AGILE SOFTWARE DEVELOPMENT PRACTICES IN SOFTWARE ENGINEERING EDUCATION

MERT AKKANAT

A Thesis Submitted to

The Graduate School of Izmir University of Economics

M.SC. Program in Computer Engineering

İzmir

2022

ABSTRACT

APPLICATION OF AGILE SOFTWARE DEVELOPMENT PRACTICES IN SOFTWARE ENGINEERING EDUCATION

Akkanat, Mert

MSc. Program in Computer Engineering

Advisor: Asst. Prof. Dr. Kaya OĞUZ

June,2022

Agile development practices have been in widespread use in many software companies since their introduction. While the principles clearly state that face-to-face communication is the best way to convey information to other team members. However, the global pandemic of 2020 has forced the practices to be applied online instead of face-to-face. The scope of this study is to analyze the effect of the Agile methodologies on software education projects. To analyze the effect, several Agile practices were applied to a junior-level software engineering course which includes a team project assignment. The course had 59 students who formed 15 teams in the Fall semester for the 2021-2022 academic year. Two of these teams have volunteered to participate in the application of Agile practices that are based on the Scrum methodology. The purpose is to compare these two teams with other teams who have not applied any Agile practices but followed the fundamental prescriptive process that is made up of specification, design, implementation, and testing activities. With the differences between both these groups, this study expects to reveal Agile practices are suitable to applying to course projects. The following practices are incorporated into the two volunteer teams: Sprint planning meetings, Daily meetings, Weekly meetings, Retrospective meetings, Pair programming sessions, Code review sessions. At the end of the semester, two surveys that focus on the effects of the Agile practices and performance have been conducted of survey and the results show that the customized

Agile practices are suitable to apply in university education.

Keywords: Software Engineering Education, Agile Methodologies, Teamwork, Assessment, Software Project Management, Scrum Framework

ÖZET

YAZILIM MÜHENDİSLİĞİ EĞİTİMİNDE ÇEVİK YAZILIM GELİŞTİRME UYGULAMALARININ UYGULANMASI

Akkanat, Mert

Bilgisayar Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Kaya OĞUZ

Haziran, 2022

Çevik yazılım geliştirme uygulamaları, ortaya çıktıklarından beri birçok yazılım şirketinde yaygın olarak kullanılmaktadır. Çevik ilkeler, yüz yüze iletişimin bilgiyi diğer ekip üyelerine iletmenin en iyi yolu olduğunu vurgular. Ancak 2020 yılında ortaya çıkan küresel salgın, uygulamaların yüz yüze yerine online olarak uygulanmasını zorunlu kılmıştır. Bu çalışmanın kapsamı, Çevik yazılım metodolojilerinin yazılım eğitimine etkisini analiz etmektir. Etkiyi analiz etmek için, takım proje ödevi içeren üçüncü sınıf yazılım mühendisliği kursuna Çevik yazılım yöntemleri uygulanmıştır. Dersin 2021-2022 eğitim-öğretim yılı güz döneminde 15 takım oluşturan 59 öğrenci yer aldı. Bu ekiplerden ikisi, Scrum metodolojisine dayalı Çevik uygulamaların uygulanmasına katılmak için gönüllü oldu. Bu tezin amacı, iki ekibi herhangi bir Çevik uygulama uygulamamış ancak spesifikasyon, tasarım, uygulama ve test faaliyetlerinden oluşan temel kurallar içeren süreci takip eden diğer ekiplerle karşılaştırmaktır. Her iki grup arasındaki farklılıklar ile bu çalışma, Çevik uygulamaların üniversite eğitimine uygun olduğunu ortaya koymayı beklemektedir. Agile yöntemler iki gönüllü takım üzerinde uygulanmıştır: Sprint planlama toplantıları, Günlük toplantılar, Haftalık toplantılar, Geriye dönük toplantılar ,Eşli programlama oturumları ,Kod inceleme oturumları.Çevik yazılım geliştirme yöntemlerinin katkılarını izlemek için TPS ve GitHub günlükleri kullanılır. Ayrıca

haftalık toplantı notları, ikili programlama takip formları, kod incelemelerine ilişkin yorumlar ve sprint geriye dönük dokümanları Google Drive'da ortak bir dizinde tutulmaktadır. Dönem sonunda hem çevik uygulamalara hem de çevrimiçi performanslarına odaklanan iki anket yapılmıştır ve sonuçlar incelenip Çevik uygulamaların, üniversite eğitiminde uygulanmaya uygun olduğunu göstermektedir.

Anahtar Kelimeler: Yazılım Mühendisliği Eğitimi, Çevik Yöntemler, Takım Çalışması, Değerlendirme, Yazılım Proje Yönetimi, Scrum Penceresi

ACKNOWLEDGEMENTS

I would like to thank Asst. Prof. Dr. Kaya OĞUZ, my supervisor, for his research support, guidance, advice, critique, motivation, and understanding during the research process. I've always been able to reach him when I need him, he didn't hesitate to help me through the process. Thanks to him, I learned a lot of subjects and I was able to complete this thesis. I would be grateful if I have a chance to studying Ph.D. with him.

I would like to thank my dear wife Göksu, who has always supported and motivated me and shown me patience when undertaking my research and writing my thesis. Lastly, also the most important thanks to my family. Everything they have done in my life without you through the process was more challenging.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Agile development practices have been in widespread use in many software companies (Nagappan et al., 2003) since their introduction (Navas et al., 2014). Agile principles are applied in frameworks such as Scrum Framework, Extreme Programming (XP), Kanban, Feature-Driven Development (FDD), Lean, and Dynamic System Development Method (DSDM). They are proven project management methodologies in the software industry (Chen, 2017). One of the most popular framework is Scrum and it is mostly combined with XP. (Schwaber, 2010).) There are attempts to teach and apply Agile practices in software engineering education (Hanks et al., 2004). This study differentiates them with several customizations including other disciplines such as XP, pair programming, code reviewing, and several testing approaches. While the principles clearly state that face-to-face communication is the best way to convey information, the global pandemic has forced the practices to be applied remotely. The global pandemic provided an opportunity to analyze their effectiveness in a remote setting.

The scope of this study focused on the application of the customized Agile practices on undergraduate software engineering course that includes a team project assignment. The modified Agile practices were used to increase awareness, contribution, and knowledge about the project. Agile practices need to track the individual contributions of the team members at the task level. In this course, the projects of the students are tracked by the application named “Task Point System (TPS)” which evaluates each student individually to find out their contributions and grade them justly (Oguz and Sevcan, 2017). The course had 59 students who formed 15 teams in the Fall semester for the 2021-2022 academic year. Two of these teams have volunteered to participate in the application of Agile practices that are based on the Scrum methodology. The purpose is to compare these two teams with other teams who have not applied any Agile practices but followed the fundamental prescriptive process that is made up of specification, design, implementation, and testing activities.

The following practices were incorporated into the two volunteer teams.

- Students were asked to hold the online daily Scrum meetings.
- The student groups scheduled pair programming sessions and reported their session details on the cloud.

- With the code reviews, we aimed to increase their dominance over the whole project.
- Extreme Programming (XP) practices were applied and analyzed in this thesis such as incremental design, and continuous integration and continuous delivery.
- TPS and GitHub logs were used to monitor the contributions of each individual in the teams.
- Additionally, weekly meeting notes, Pair Programming tracking, comments on the reviews, and sprint retrospective documents were kept in a shared directory on Google Drive.

  All these Scrum ceremonies and methods were modified for the university setting.

At the end of the semester, two surveys were conducted to evaluate the experience gained. This first survey has been conducted on 59 students who enrolled in this course to make a comparison between Agile and non-Agile teams. On the other hand, the second survey was only filled by Agile Teams to understand the effects of Agile practices on the development team. The survey questions focus on both Agile practices and their online performance. With the result of the surveys, observations, Scrum documents, we collect data to analyze. In this thesis, we compared and analyzed the benefits and the results of the application of Agile practices within the scope of SE 302 course.

Research Questions:

1. Are Agile methodologies suitable to be applied in the university setting?
2. What are the negative and positive effects of the Agile application on course projects?

To answer these research questions, A general survey (GS) evaluated the course term project outcomes for both Agile and non-Agile groups in Appendix-2. In GS the students were asked about the organization, challenges, completion of the project, obstacles, and their feelings about the project process. An Agile survey (AS) is applied to the volunteer groups to evaluate the outcomes of Agile practices from the perspective of students. AS is focused on Agile practices and their effects on the grades and software development process. With the results of the surveys and graded assignments, the effect of Agile practices has been analyzed and discussed.

The outcomes of these analyses and measurements are discussed in Section 4 as answers to the research questions posed earlier.

CHAPTER 2: LITERATURE REVIEW

There are numerous studies on the use of Agile methods in software engineering education. In this section, we closely examine the ones that are similar to like our work, specifically those which implement the Scrum framework, since the practices applied during the course rest on the Scrum framework, too. Scrum is widely used in project management and has become more popular day by day (Chen, 2017; Dingsøyr and Lassenius, 2016). In the software industry, many start-ups and large companies use it, which is a framework of Agile methodology (Lindvall et al., 2005). We divide literature review into 5 parts, Agile, Scrum, Pair Programming, Code Review, and Task Management System.

Martin Blom discussed Scrum and XP for computer science education in 2010. Martin assumed that business and the larger projects need long-term support and need maintenance. However, student projects are short-life projects and he mentioned that Scrum adapt to education easily. Because Scrum is more suitable for short-life and small projects. He analyzed every section of Scrum and explained its suitability for Computer Science. According to his findings, all points are relevant except the Product owner because there is no exact customer in term projects. The outcomes were listed in Table 1.

Table 1. Scrum attributes and their availability to education (Source: Blom, 2010)

Scrum Attribute	**Convenience**
Sprints	Extra relevant
Team	Relevant
Product Owner	Less relevant
Backlogs	Relevant
Burndown Chart	Relevant
Sprint planning meeting	Relevant
Sprint Review	Relevant
Scrum Master	Relevant

Blom analyzed the suitability of Extreme Programming (XP) for computer science education. XP is an Agile Software development framework that aims to increase the quality of software without the old-school methods, processes, and documents. According to his research, students reported that working in pairs has advantages over working alone and they believe pair programming is beneficial even if it has several challenges. Test-Driven Development was analyzed, and it was found TDD was difficult to apply but also relevant to Computer Science Education (CSE). He suggested in Table 2 that all key attributes of XP are suitable for applying to education.

Table 2. Extreme Programming attributes and their availability to education according to Blom (Source: Blom, 2010)

XP Attribute	**Convenience**
Pair programming	Relevant
TDD	Relevant
Incremental Design	Extra relevant
Continuous Integration	Relevant
Collective Code Ownership	Relevant
Informative Workspace	Relevant
Coding Standard	Relevant
Sustainability Pace	Relevant

Scrum Framework has been applied at Ohio University during the Fall and Spring semesters of the 2020-2021 academic year by Lynn C. Stahr. Scrum introduction, instructions, and basics such as daily meetings, sprint planning, sprint review, and sprint retrospective were explained to groups for their adaption to the education perspective at the beginning of the semester. To evaluate the outcomes, Lynn preferred to use surveys, reflected essays, mid-term, final, and project grades and he summarized students were confident in their technical skills and soft skills. Lynn experienced the most improvement in management skills. The reflected essays were

collected from students and their overall results show students indicated a valuable experience.

One of the existing applications of Agile methodologies was done by Antonio Jurado Navas and Rosa Munoz Luna where they applied Scrum methodologies to higher education in a discipline other than software engineering (Jurado Navas et al., 2014). Their thesis shows us that Scrum has benefits in other disciplines, too. Their inspiration was the gap between industry and university education. They followed all Scrum basics in an English writing course. They observed that,

"With the constant feedback from the teacher, both in the classroom (review meetings, especially) and online, the level of self-efficacy in students and their need to perform better increases".

To summarize, observations and findings show Scrum is valuable and able to be used for effective teaching for university education.

The other approach is the practice of pair programming under Extreme Programming which has also become a popular approach for the software development teams. Pair programming has been applied in the course and senior projects of computer and software engineering departments. They mentioned pair programming process was retained without any issue and it helped students to increase motivation and collaboration. In addition, they noted that personality played a significant role in matching patterns. In the future, they will apply the Myer-Briggs personality test before pair programming. (Nagappan et al., 2003; Hanks et al.,2004; Williams et al.,2000)

Regarding Williams and Kessler's experiments, pair programming plays a key role to increase the quality of software, and decrease the time of the development process, teamwork, and knowledge transfer (Williams and Kessler, 2002). Pair programming helps software engineers to be social, increase communication skills, and make the development process more pleasant. According to Williams and Kessler, the usual way is to apply pair programming with two main roles which are called driver and navigator. The driver stands for writing code or design with a keyboard, the navigator monitors the process, and the driver's work in terms of code quality and gives suggestions to make the code cleaner. Their evidence indicates that sharing a keyboard and screen brings code reviewing, faster thinking, and higher defect

prevention for creating a better-quality product. They assume that pair programming increases ownership of the works of both peer's work and the entire project. In addition, pair programming forces programmers to work together and transition their knowledge to each other.

Cockburn and Williams applied the pair programming approach in a Software Engineering course at the University of Utah and found out that groups that applied pair programming are more successful than other groups (Cockburn and Willams, 2002). The main outcome of this experiment is to analyze how pair programming affected the learning outcomes and to measure quality increase without increasing time. According to Cockburn and Williams, pair programming plays an important role to enhance programming knowledge and other skills such as problem-solving, investigating skills, team building, and communication among the students.

Hanks et al. reviewed the application of pair programming during software engineering education (Hanks et al, 2004). They analyzed the benefits, challenges, and practices of pair programming. The metrics such as performance were calculated and quality of code, duration, and effort of programmers on their own and when they worked in pairs. They address that pair programming helps students to learn and absorb course outcomes. They compared final grades of beginner and later courses between paired students and non-paired students and suggested that paired programmers were more successful. Another study in this review shows that students who practice pair programming have shown better results on graded assignments and more satisfaction and less frustration in doing course projects. On the other hand, there were some challenges such as scheduling pair programming sessions. Most of the students reported spending extra time to pair programming was compelling because of the availability of the group members. The other concern was partner compatibility. Bevan et al. reported a disparity between the partners (Bevan et al, 2002). They mentioned that more experienced students were impatient and ignored the suggestions of others. Bevan et al. also added experienced programmers were confident that their coding style is the right, and this situation brought mislearning for novice programmers who are paired with.

Code review is a common approach that has been used in the software industry for many years (Dogan and Tuzun, 2022). For example, in Google, code review has been used for many years. According to Caitlin et al., Google uses two concepts:

ownership and readability. Ownership means the code which is written by developers should be accepted by directory owners. The second concept is readability, and the changed code should be reviewed by someone who has a readability certification in a particular programming language. In this thesis, code review was used for improving code quality, teamwork, and knowledge transfer.

The benefits of code reviews in education have also been reported in the current literature (Lindwal et al., 2005; Blom, 2010). For example, Sripada et al. applied a code reviewing approach to undergraduate software engineering education (Sripada et al., 2015). They applied a code review approach to their mid-sized project for 12 weeks. 46 groups contained 4-5 developers in each group. They wanted to write a sample report about each code review section that includes team name, reviewer name, review time, tools, defect list, number of functions, and lines reviewed. After the code review session, the development team fixed issues and indicated the troubleshooting steps in this report. For analyzing the results of the code review application, they conducted an online survey for each release. Regarding survey results, they assume that collaboration of students, team communication skills, and awareness of the whole project has increased. Several students reported that their code development and analyzing skills were dramatically improved.

Rong et al. examined the code review performance with a checklist on inexperienced students (Rong et al., 2012). They created a checklist for managing the code review section which includes goals, questions, and metrics. This checklist was used by 9 students and the other 7 students didn't use it. They compared the results of the collected data in the data analysis part. There was no evidence to prove the checklist approach helps students to find more defects and review rates. However, their study shows us the checklist helps students to conduct code reviews and increases its efficiency.

The other work was related to avoiding unfair grading for the term projects with the assessment of the individual contribution. Task Point System (TPS) is an online application that keeps track of the contributions of each member of a team. It has been developed by Kaya Oguz and has been in use since 2017. The details of the system are provided in his master's thesis of Muratoglu, titled "Assessment of Team and Individual Contribution in Computer and Software Engineering Education" (Muratoglu, 2021) and will be discussed in more detail in the method section, since it

is one of the tools that have been used in this thesis, as well. TPS provides shreds of evidence and outputs. TPS also has proven benefits such as the following.

- TPS provides a professional software development environment to students
- The lecturer and Scrum master are able to see the workload and problems of the development teams.
- TPS encourages the students to break down the process into individual tasks so that the lecturer would be able to see their contribution and provide a fair assessment.
- Students are able to see tasks of team members and they can comment on tasks, this supports the transparency pillar of the Agile methodology. This option increases awareness of the whole project.
- The Scrum team is able to create tasks and change with the acceptance of other members, this option supports the "Agile teams are self-organized" principle.

These benefits make the project management process easier to follow with a student-friendly user interface and allow teams to be better organized. According to Muratoglu, TPS has increased the fairness of the grades because of the monitoring team member's participation and contribution to the project.

As a result of the literature reviews and considering the popularity of Scrum, pair programming and code review were used. In addition, students were already taken courses which includes these methods and techniques. Therefore, they were familiar how to apply and knowledge.

CHAPTER 3: METHOD

The proposed Agile practices in a Scrum framework have been applied in the "SE 302 Principles of Software Engineering" course with 59 students at the Department of Computer Engineering of the Izmir University of Economics. The instructor randomly creates project groups. In two of these groups, a total of 9 students volunteered to try the proposed practices. A "Family Tree Application" has been assigned as the course project which represents the relations between family members. They were free to select any programming language, database, or platform. However, there were several restrictions as listed below.

- The application must have a graphical user interface (GUI).
- The application must have a one-click setup file.
- The application must have a help file.
- The application must store data in a file, database, JSON, etc.
- The application must run without any error.
- The application must not require the installation of additional software or servers.

Before the implementation phase, the teams had to extract the functional and system requirements by interviewing the lecturer, since the project topic and scope were delivered orally. The extracted requirements were listed in a formal requirements document that has been reviewed by the lecturer. After the feedback, the students were asked to design the software using the requirements they have extracted. Each team submitted their software design documents which includes the major components of the software and how these components interact. The text was supported by standard UML diagrams, such as class diagrams to illustrate the relationships of the classes and interfaces and sequence diagrams to visualize the interaction between the classes. These documents were also reviewed by the lecturer and feedback has been given before they start the implementation. The time spent on these documents is important since they help the students to understand the concept and boundaries of the application.

3.1. Application of the Scrum Framework

In this study, we have taken advantage of the Scrum Framework, pair programming, code review, and TPS (Task Point System) to evaluate the success of the application of Agile practices in education.

Scrum is an empirical framework that was first mentioned in 1986 January by Hirotaka Takeuchi. Scrum is the framework that provides teams and organizations to solve complex issues with transparency, inspection, and adaptation.

1. Transparency: Everyone is aware of every part of the project.
2. Inspection: Everyone is involved in the project development process including the stakeholders.
3. Adaptation: Ability to respond to changes in the development process. Adaptation is strongly related to transparency and inspection.

The main purpose of Scrum is to increase adaption speed to changes, continuous development, and continuous delivery under consideration of customer feedback.

All processes are driven by the Scrum team which has three roles. These roles and their representation in the team project are given below.

1. **The Product Owner** acts in an active role in the development process in terms of creating backlog items which are the requirements that are extracted from customer demands. They should understand customer expectations and submit this list to the Scrum team.
    - The lecturer acted as a Product owner and gave feedback to students to analyze requirements.
2. **Scrum Master** is responsible for following Scrum rules to make Scrum more effective. They coordinate the team to daily Scrum, lead sprint meetings, and communicate with stakeholders.
    - The MSc student was a Scrum master on the project to help the team to obey Scrum rules.

    2.1. Mert Akkanat, the Scrum Master MSc. researcher

    - Mert Akkanat is an MSc. student at the Izmir University of Economics and graduated from the Software Engineering department in 2017. He has 5+ years of experience in the software industry and has been

working as a Software developer and Software Team Lead. In his current position in the industry, he is managing 4-6 people in the multicultural Scrum team with the following Agile principles. He strictly follows Scrum ceremonies to increase agility in the team. He is actively using Jira Atlassian, Azure CI/CD pipelines, Git, code reviewing, and refactoring basics. He also has "Software Processes and Agile practices", "Client Needs and Software Requirements", "Agile Meets Design Thinking" and "Agile with Atlassian Jira" certificates from various authorities.

3. **The Development Team** manages the completion of tasks that are listed in the backlog in Sprints. The team contains all members for complete sprints such as engineers, designers, and business analysts. The development team is responsible for finding best practices, implementation, writing tests, and creating releases.
    - Students were the development team in the Scrum team which managed all sprints and their recommended process.

The regular Scrum process starts with the creation of a product backlog. The Product owner gathers requirements from customers and creates a product backlog with user stories. These backlog items are going to be used to create tasks and features to manage sprints.

An iterative cycle in the Scrum Framework is called a Sprint. Sprints are all planned based-on backlog items which are gathered from the customer before starting the development process with the development team. Sprints must be between 2 weeks and 4 weeks; longer sprints are not applicable so that responsiveness is preserved. The main reason for that is increasing responsivity and agility. In the industry, development teams want to create one or two-weeks sprints because planning sprints require good developer experience, and the team wants to increase velocity and agility. Teams aimed to deliver good quality releases while considering the feedback from customers.

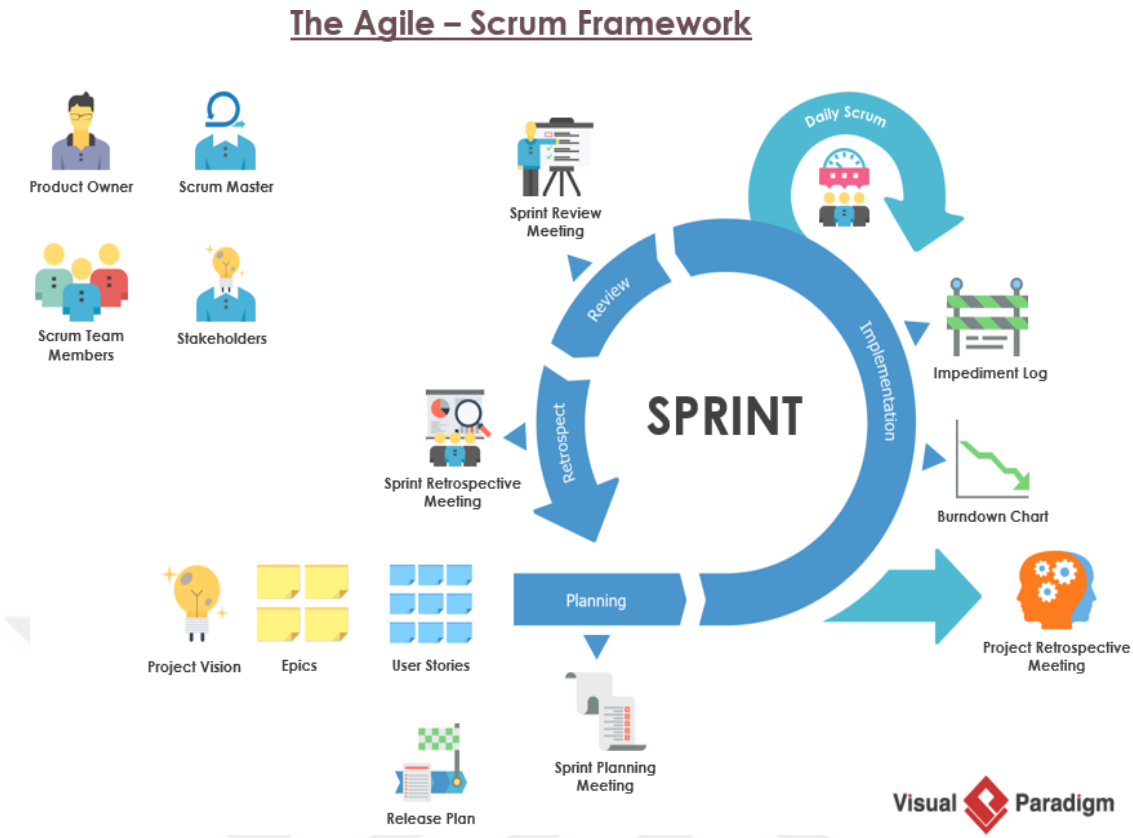

Figure 1. Agile & Scrum framework flow (Source: Visual paradigm, 2021)

Scrum teams are self-organizing, Scrum team try to find out the best way the solution by themselves instead of managing from anyone outside of the team. Teamwork and agility are essential for Scrum Framework and Scrum needs daily, weekly, and sprint meetings with documentation for the monitoring process. For evaluating the benefits of pair programming the team kept pair programming sessions in an online spreadsheet tracker. We stored daily meeting notes and weekly notes for monitoring what the team did last week. All documents were open access to the development team, Scrum Master, and lecturer. The team used this information for planning the upcoming sprints. All these activities make the process visible and help the sprints to follow the progress, increase team cohesion, and provide transparency, adaptation, and inspiration in the project development process. They are important because Scrum is not fully theoretical but also empirical. These three pillars also play an important role when the Scrum team works with customers in an Agile way because Scrum accepts CI/CD (continuous integration and continuous delivery), and this approach needs client feedback. However, CI/CD pipelines were not created in any

environment because of the lack of technical competence in DevOps. Students proceed with their functional tests in a local environment after every Sprint.

In this thesis, we have strictly followed the Agile manifesto's four principles Beck et al. (2001). These principles and their representation in the team project are given below.

1. Individuals and interactions ***over*** processes and tools
    - Human-centralized mentality instead of using procedures to proceed. This situation increases the responsivity time. During the semester, we focused on the lecturer's expectations and project requirements with strong team communication. They were able to customize the processes personally if necessary.
2. Working software ***over*** comprehensive documentation
    - Except for requirements and design documents, the lecturer did not want comprehensive documents. However, he wanted to see working software at the end of the semester. The development team conveyed information to other team members via daily and weekly meetings instead of creating any document about implementation. This approach was acceptable in short-term projects because no one would need historical stored data in the future.
3. Customer collaboration ***over*** contract negotiation
    - In term projects, there was no contract negotiation. The whole process is managed like customer collaboration because of the role-playing system. We followed a role-playing system because this is a course project, and we have no customers. The development team has gathered feedback from the lecturer and the MSc student.
4. Responding to change ***over*** following a plan
    - In the implementation phase, the team realized some missing requirements because of lack of experience. The development team adapted to the situation and created user stories, implementation, and tests for responding to these changes.

Regarding Beck et al. (2001) The Agile 12 principles help teams to be more responsive, Agile, flexible, and able to adapt to new requests. These principles, and how we applied them with an educational approach are as follows:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
    - Weekly meetings and in-class meetings with the lecturer were good for observing customer satisfaction.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
    - The changes that come in the middle of the project are very natural because of the requirements written by inexperienced students. New requirements always occur during the project development process.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
    - The Scrum master acted like a customer, and they wanted to see a working demo at the end of each sprint.
4. Businesspeople and developers must work together daily throughout the project.
    - The lecturer and the Scrum master role played as businesspeople and gave feedback to the team.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
    - The MSc student acted as Scrum master and motivated the team about their capacity and ability.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
    - The weekly, sprint online meetings are done via Teams and Google meet. For daily meetings, we have met on a mobile chat application because of the busy schedule of both the developer team and MSc. student.
7. Working software is the primary measure of progress.
    - Working software was shown to the Scrum master every sprint with new features. In addition, project groups reported their project to the lecturer at the end of the semester with an oral presentation.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- All participants were able to see due dates, estimations, and work logs on TPS.

9. Continuous attention to technical excellence and good design enhances agility.
    - The development team acquired clean code and useful design enhanced with code review sections, pair programming, and a shared GitHub repository.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
    - The development team has taken advantage of TPS which has priority, and difficulty properties to identify the precedence of work.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
    - The lecturer did not interfere with the team to keep the team self-organizing. The Scrum team decided on requirements, architectures, and design with their experience.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
    - The Scrum team met one day each week to describe the status, milestones, and plans.

These principles were also helpful for students because undergraduate students are remarkably busy with other responsibilities such as projects, presentations, and midterm exams. In the limited time allocated to their project, they do not want to use tools, write documentation, and create a full plan. This approach encourages the students to increase their concentration and contributions to the project. The other advantage of Scrum Framework is increasing awareness of the project management tools such as Microsoft Excel, MS Project, and Jira for being ready for the industry.

At the start of the semester, the lecturer created randomized groups, prepared a term project, and shared it with groups during the course lectures. After the project assignment phase, both Agile and non-Agile groups prepared requirements and design documents to be ready for development progress. Two groups volunteered to apply Scrum to their project, rest of the groups were free to select their project management process. However, all groups had to use TPS (Task point system).

We met with Scrum groups to give a brief explanation of Scrum, code management, and team collaboration. Ensured all Scrum members understand the definition of done (DoD). DoD is a changeable acceptance criterion for closing a task. In our Scrum approach, code standards should be regular, the implementation must be done, the review must be done, and tests are executed for closing a task successful. We expected to address all functionality, bugs, obstacles, and risks by removing dependencies. Regarding creating a backlog, Scrum teams used requirement documents. Thereafter, backlog, daily meetings, spring planning, sprint review, and sprint retrospective were explained to the Scrum groups. We created a sample guideline about how to manage the Agile process in a software engineering course. The education perspective guideline is available in Appendix 1.

Scrum ceremonies and practices were customized because the Scrum criteria does not meet in availability, knowledge, and capacity of the students. Therefore, some modifications and additions were existing in our Scrum Framework is shown below Figure 2.

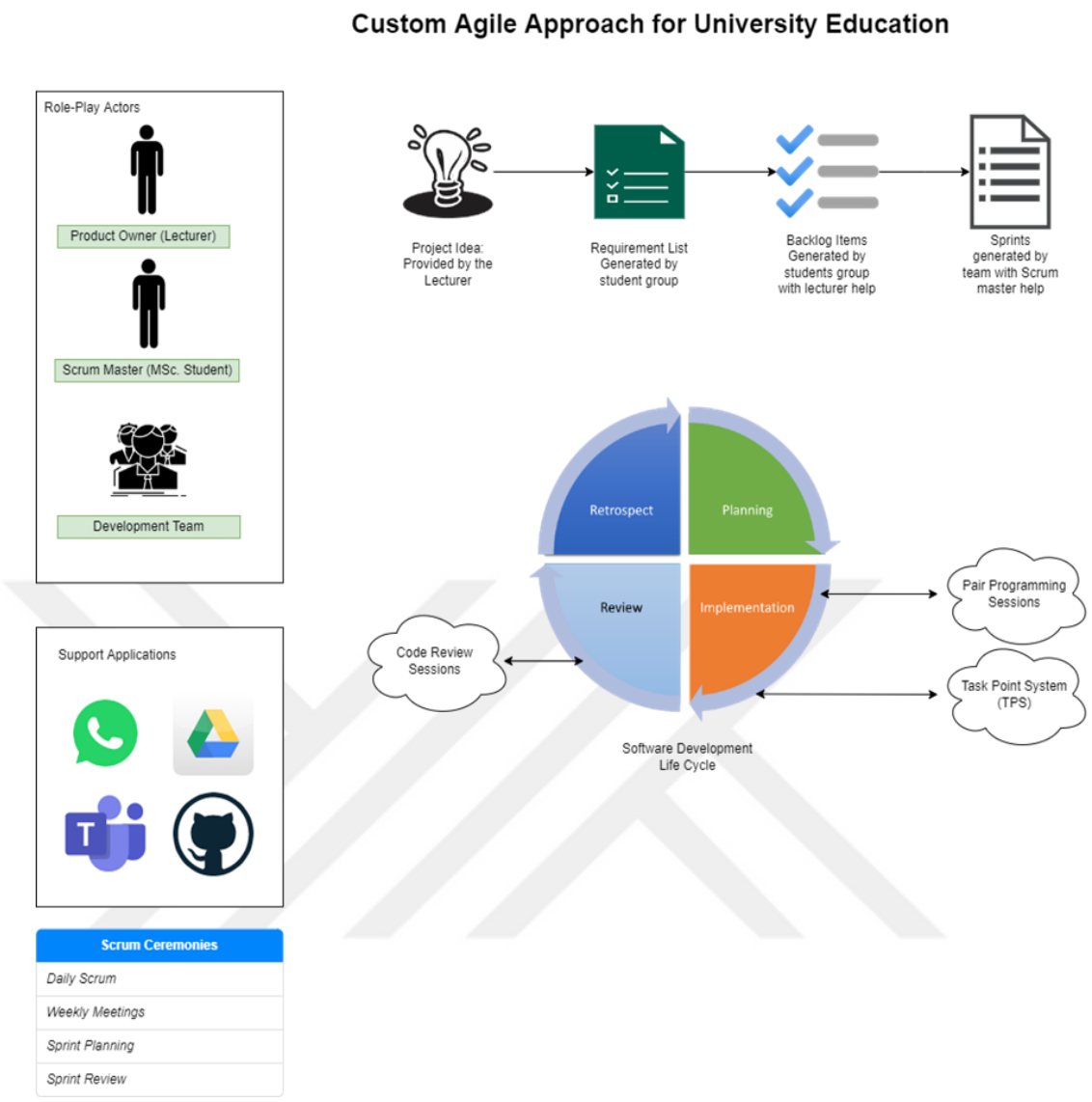

Figure 2. Custom Agile practices for university courses

3.2. Adaptation Scrum Ceremonies

Backlog is created by the Product Owner. However, in the term project, there was no product owner. The development team created backlog items using the requirements. The team faced some challenges to create a backlog because of a lack of experience. One of the skills they lack was how to prioritize the backlog items for the sprint. Another one was giving an estimation of each task in terms of importance, and difficulty

The Sprint planning part was done before starting the implementation part with the Scrum team by using the requirements document. This is the last part that the development team needs to discuss the road plan before the Sprint starts. A list of activities and user stories were written down to document with the developers' agreement by taking team capacity, and velocity into consideration. We scheduled weekly meetings for discussing “what we did”, and “what are we going to do next week” for each team and took notes on Google Drive. We compared notes in every weekly meeting to see progress.

Daily meetings are not feasible for students because of their other responsibilities to other courses. Instead of scheduling face-to-face daily meetings, we preferred to create a group on a mobile communication application to discuss our findings, obstacles, and future plans. All team members joined this group and give their contributions daily. Besides that, group members log their daily work to a spreadsheet as shown below.

Names/Days	***Monday***	***Tuesday***	***Wednesday***	***Thursday***
Std 1				
Std 2	Weekly meeting	Basic parent-child algorithm		Did GUI
Std 3	Weekly meeting	getting data from user to db	created the db called famtree and created the tab	
Std 4	Weekly meeting			db connections

Figure 3. Daily meeting tracker

However, continuity of daily meetings was not possible because students weren’t willing to do the extra job because of their other responsibilities and social life. For handling this situation, the team just chats daily on the mobile application to discuss the process.

A Scrum board is necessary for tracking the process. Instead of using third-party applications like Trello, Jira, and DevOps, all teams used TPS. They always

access the board via the internet to follow processes and log activities at any time. There was no physical Scrum or Kanban board in the university.

Sprint Planning meetings were managed by the Scrum team at the start of each sprint. The Scrum team didn't get face-to-face because of the pandemic. The teams connected on Google Meets or Microsoft Teams to decide which backlog items move to the next Sprint.

Spring Review meetings are arranged at the end of each sprint. Scrum teams discussed the obstacles, risks, and implementation of the previous Sprint with the team and took notes on Google Drive. The challenge was about the lifetime of Sprints. Some of the students didn't finish their tasks in time. Therefore, teams wanted to extend Sprints one more week before jumping to the next sprint. Following the conventions of the Scrum framework, the tasks have been moved to the next sprint instead of an extension.

Cross Functionality has been seen many times in our approach. Because the development team must know every part of the project. Regarding this, they swapped roles between database design, back-end, and front-end. Pair programming especially helped students to be cross-functional. It has been made mandatory to change the roles and we realized that students weren't willing to take on extra responsibilities in a different part of the project.

3.3. Application of Code Review

Code review was one of the most important approaches for increasing code quality in the development phase. Regarding the survey results, we analyzed that code review increases team collaboration and team synergy but also helps the teams to find the best practices. During the semester, the teams aimed to decrease the costly errors by discussing with each other. They scheduled weekly code review sections and the findings were kept on a spreadsheet. In these sections, the Scrum team discusses implementation, and they have done code refactoring. These weekly sections are done by student-to-student sections without a Scrum master or lecturer. Code review sections increase team awareness of every piece of code on the project and knowledge about coding practices. The team found it beneficial to be ready for the presentation at

the end of the semester because the lecturer wanted to listen to every functionality of the project from each member.

During the term project, we reviewed code in two separate ways.

1. Student - Student review
    - Which stands for decreasing the number of bugs, increasing consistency, and doing optimization.
    - For knowledge translation to other team members.
    - These sessions help the developer to acquire estimation for future tasks.
2. Student - MSc student (Scrum master) review
    - Which stands for increasing engineering perspective with the Scrum master's experience in terms of code complexity, design patterns, optimization, naming convention, and redundancy.
    - That increases the habit of working with an experienced person to decrease the gap between university and industry.

3.4. Application of Pair Programming

In the industry, pair programming is used for adapting a new team member to a project. Pair programming used for adapting developers to other tasks and parts of the project such as documentation, user interface, database, implementation. We aimed to increase code quality and knowledge transfers between developers. Another reason to use of pair programming is to increase the number of instant feedback and decrease potential defects in development progress. Before creating pair programming groups, we consider the strengths, weaknesses, the experience of the students, and whether they have previously enrolled in this course or not. Another evaluation criteria for experience are internships and working experience in the industry. Because of such information, the development team created a dynamic pair programming schedule. During this project term, we applied Pair Programming at least one day a week. The time was not strict, but the Scrum master suggested one session for at least one hour with switching roles frequently. In the COVID era, we scheduled a pair of programming sessions online instead of working the same keyboard, and mouse. The

pairs should focus on the same problem instead of individual work with the patient. We aimed to improve teamwork and technical skills with the driver-navigator approach. We identified the weaknesses and strengths of students besides coding, UI design, database design, and algorithm design, and created weekly pairs. To encourage students, students role-played as senior and junior developers. Online pair programming sessions were organized by the Scrum team based on their availability and we tracked sessions in an Excel file on Google Drive. There are three types of Pair programming types that are applied:

1. Junior- Junior (Freshman or non-experienced students)
    - This approach forced the students to learn about their shortcomings in the topic. For example, if both students had no experience with databases, they should learn about database creation, queries, relations, etc.
    - Generally, pair groups used the ping-pong pair programming style which is switching roles between them.
2. Junior-Senior (First-year student or non-experienced and second year enrolled or experienced)
    - The aim of this style is for non-experienced can learn about other students experiences. For example, an inexperienced student uses the keyboard as a driver, while an experienced student monitors the driver's work, and gives suggestions.
    - On the other hand, if an experienced student is a driver, the inexperienced students can absorb and understand the driver’s development style and knowledge.
    - Generally, pair groups used the backseat pair programming as a style in which the navigator watches the driver’s work instead of using the keyboard.
3. Senior-Senior (the second year enrolled or experienced students)
    - In this approach, the objective is to increase code quality by decreasing bugs and defects. The navigator realizes the mistakes made by the driver and addresses the issue in the development phase instead of testing.

- They focused on the best practices instead of just writing code, this brings a higher engineering point of view.
- Generally, pair groups used the ping-pong pair programming style.

Pair programming adaptation in university education required several changes when applying to university education. As usual, pair programming is a face-to-face activity on the same desk with the same computer. However, it was impossible to adapt to students because of their availability. They applied that with screen sharing sessions via virtual environment applications such as Discord, Microsoft Teams, and Google Meet. One of the Agile groups has three students instead of four because one of the students did not participate in any Scrum ceremonies and implementation. We modified the pair programming approach for this group. They had longer sessions than the other teams. One of them drives the session and two of them act as navigators. They changed their roles frequently and we keep this section details on a spreadsheet as shown below in Figure 4.

	Week	Date	Driver Name	Observer Name	Duration	Comments
2	Week 1 group 1	03.12.2021	Std 1	Std 2	30 mins	Discussed about how "Me" will be created.
3	Week 1 group 1	03.12.2021	Std 2	Std 1	30 mins	A class named "Person" is created. JSON file saving and opening is still in test.
4	Week 1 group 2	04.12.2021	Std 3	Std 4	30 mins	Use case and state diagrams are designed.
5	Week 1 group 2	04.12.2021	Std 4	Std 3	30 mins	Activity and sequence diagrams are designed.
6	Week 3 group 1					
7	Week 3 group 2					
8	Week 4 group 1	20.12.2021	Std 3	Std 1	30mins	GUI design
9	Week 4 group 1	20.12.2021	Std 1	Std 3	30mins	GUI design
10	Week 4 group 2	20.12.2021	Std 2	Std 4	30mins	Designing relationships
11	Week 4 group 2	20.12.2021	Std 4	Std 2	30mins	Designing relationships
12	Week 5 group 1	27.12.2021	Std 1	Std 4	30mins	Functional testing
13	Week 5 group 1	27.12.2021	Std 4	Std 1	30mins	Unit testing
14	Week 5 group 2	27.12.2021	Std 3	Std 2	30mins	Database table creation and relations
15	Week 5 group 2	27.12.2021	Std 2	Std 3	30mins	Database connectivity and functions

Figure 4. Pair programming tracker

3.5. Application of TPS (Task Point System)

In this course, both Agile and non-Agile groups should use the Task Point system for tracking individual contributions to the project. TPS is a project task management software that is currently in use in courses that are offered by Kaya Oguz. The details of this approach and the software itself is part of the thesis by Muratoglu. He has also used TPS to evaluate its contribution to the assessment of team project. (Muratoglu, 2021).

This system allows the track of students' contributions with the task management system. This industry-like approach encourages students to increase awareness of the project and contribution. TPS provides shreds of evidence, outputs,

and observable environment for the lecturer assess the individual grade for each student. In this study, we got help from TPS to manage the Scrum process. Both the MSc student and the lecturer can monitor the process as shown in Figure 5. On the left tab, the overalls are divided into four milestones: overall project, requirement document, design document, and project delivery. These overall points were individual grades of the students with respect to their contribution to the related tasks. On the right table team members and their own tasks were listed with related milestones.

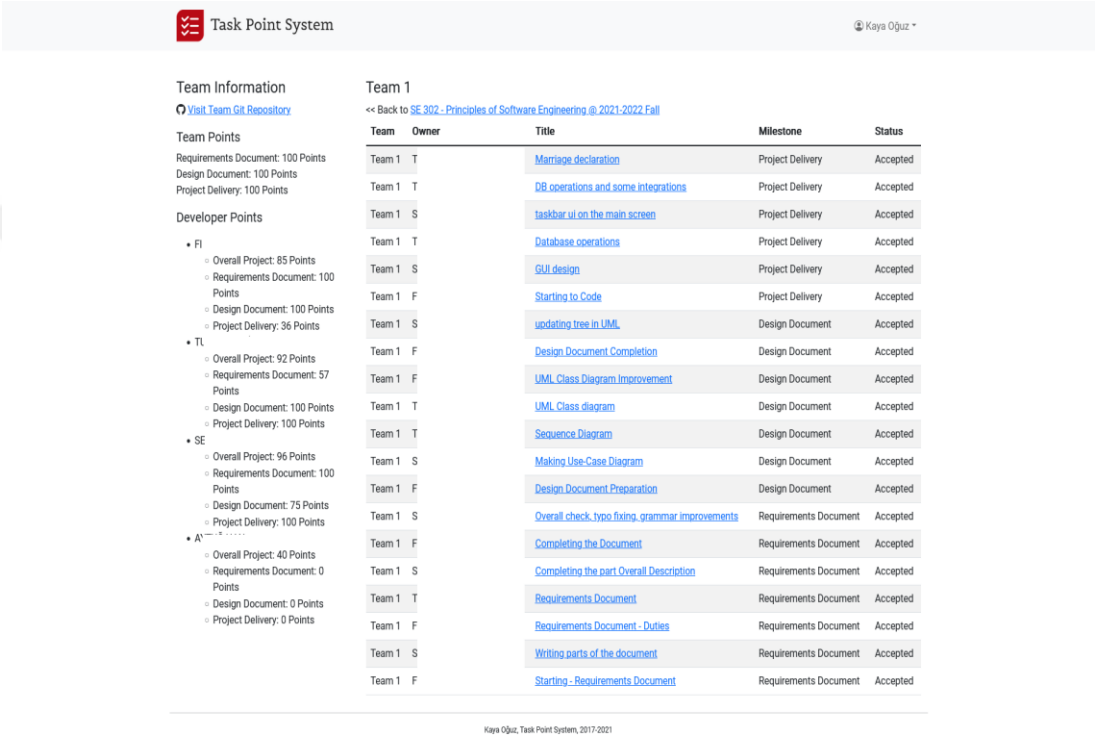

Figure 5. The lecturer's view of TPS

Milestones were only created by the lecturer with details, grade weight, and due date as shown in Figure 6. All created tasks should be related to the milestones. With milestones, the lecturer was able to track the progress of the tasks.

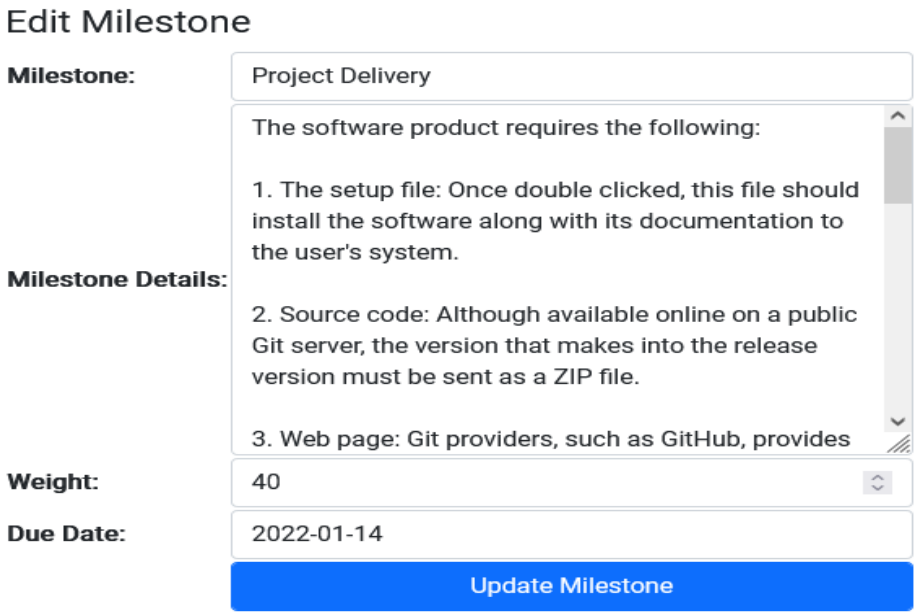

Figure 6. Milestone creation screen

The details of the TPS are listed below:

1. Team members created tasks with names, detailed descriptions, milestones, and due dates. They defined *difficulty* and *priority* value for the job. In Figure 7. task details were listed on the left-top menu.

2. Team members reviewed the tasks and gave feedback if needed. Tasks were updateable. However, for updating and approving at least half of the team should confirm the changes. After confirmation, the assignee of the task started the development process. Figure 7. showed that all approval processes for assigning a task to a developer.

3. Task point is calculated with *difficulty x priority.* The lecturer can modify these values if they are abused, such as setting a high difficulty for a relatively simple task. These constants were used to calculate the individual score for each student on the project. They were listed in the description of the task in Figure 7.

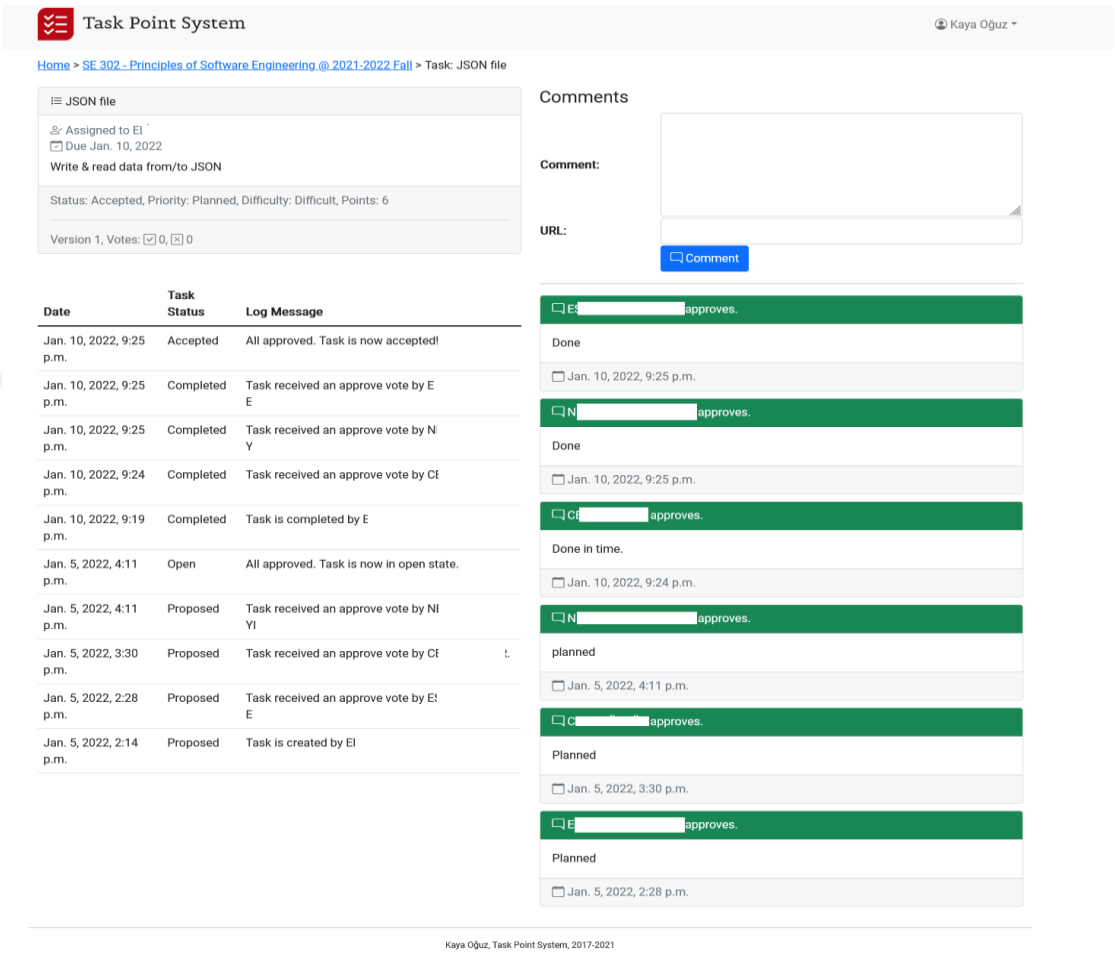

Figure 7. The task screen of one of the Agile teams was presented.

4. After completion of tasks, the assignee shared their findings and implementation with the other team members. The development team reviewed the implementation, and they were able to approve the task or ask for improvement.

5. Regarding calculating the individual grade, the total point of tasks was calculated, divided by the number of developers in a team. This number was the expected point for each developer. The lecturer compared the developer's points and expected points to fair grading.

The Scrum team monitored their Sprint progress and commented on obstacles, bugs, and ideas under the related tasks. Before the implementation phase, the team

created milestones and their due dates on TPS. Developers were able to create proper tasks which the Scrum team decided on in daily and weekly meetings. In the application phase, the Scrum master had given suggestions to the developer team for creating real-like tasks. According to tasks, the development team started their implementation in terms of UI, database, and algorithm. During the implementation phase, the team encountered a variety of problems, and reported them on TPS, then solved them in the development process. Commits are named with the issue number or task number in TPS. This creates a relationship between GitHub and TPS. The development team discussed the process in the comment section of the tasks. The lecturer and Scrum master can easily understand the individual contribution of the students with the help of TPS. This application can help the lecturer to grade fairly. The Scrum master and the lecturer analyzed individual commits on the repository.

3.6. Application of Document and Code Storage

In Scrum teams, we preferred to use online tools such as Google Drive to keep documents synchronized. Every team member was able to update these documents with their contribution. We have kept project documents such as requirements, UML diagrams, project design documents, and project schedules on the drive. Also, we stored Scrum documents of daily meeting notes, sprint retrospective notes, pair programming schedules, code review section, and functional testing reports. The lecturer was able to access these documents for monitoring the development process example team drive folder in Figure 8.

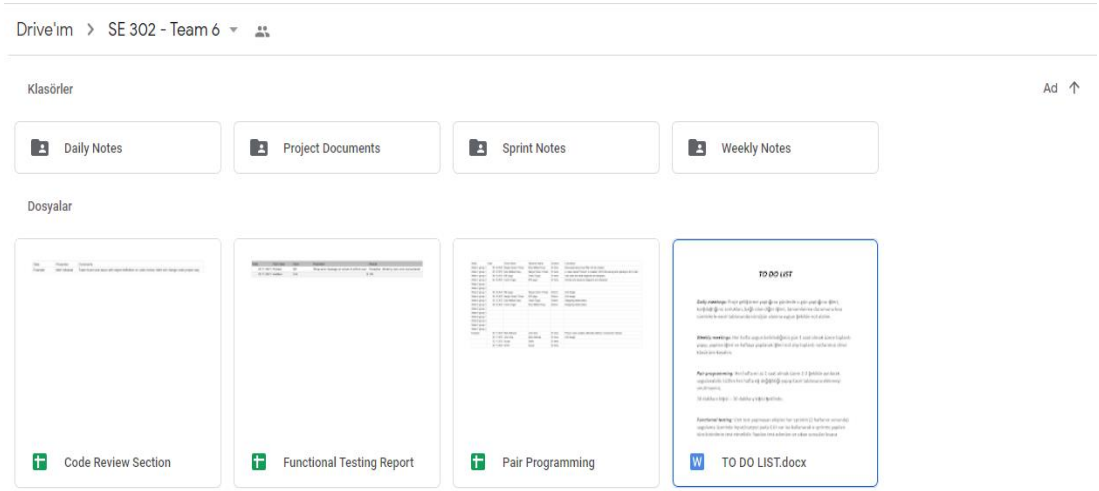

Figure 8. Example Drive Folder

Every team had to use the GitHub version control system to keep code safe, and Lecturers can observe their code contribution from commits. We preferred GitHub because it is the most common version control system in the world, and we want to decrease the gap between industry and university education. Code review sections were made on the GitHub platform with the git diff command. A student who wrote code explained code to team members for increasing awareness of every piece of the project and better code quality.

CHAPTER 4: EVALUATION

This section summarizes the experiences and observations of students who applied Agile practices during team projects and analyzes how university students apply some of the Agile practices during their projects. Our Scrum application has applied to the Department of Computer Engineering in Izmir University of Economics course called Principles of Software Engineering which has 59 students. Two of these groups (9 students) volunteered to work on the Agile approach Figure 9. The course final notes assessed 30% mid-term, 40% final exam, and 30% course projects. The project groups should select any development process and almost 50% of them stated that they used Agile on their project.

We conducted two surveys at the end of the semester to evaluate the impact of the Agile practices both positive and negative. To analyze the Agile adaptation of the project, the lecturer assigned random four people for each group because size of the project. The reason for that rests on the gap between industry and education. In industry, people cannot choose their team members and should be adapted to make teamwork because almost every software project is developed with a small or large team. Two of these groups volunteered to work with MSc. student with Agile practices and modified Agile practices, Scrum, pair programming, and code review techniques were applied to these teams.

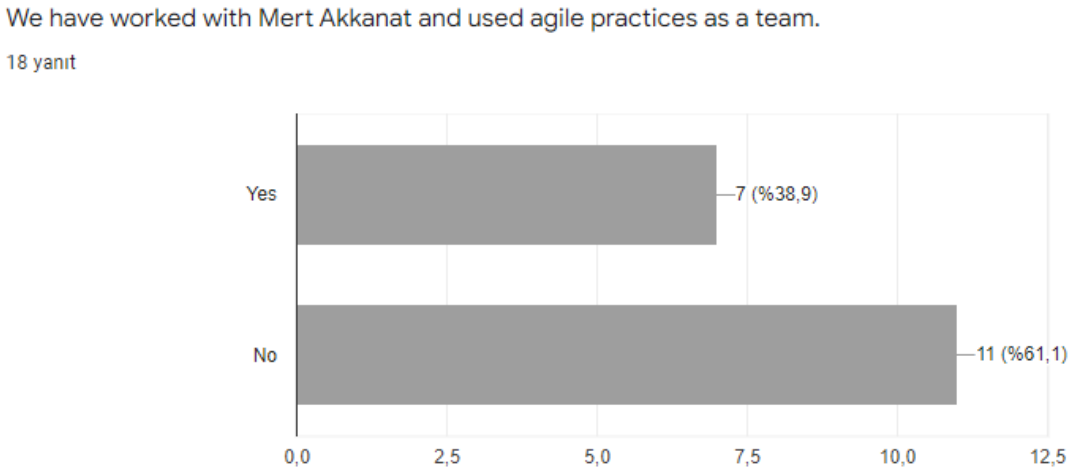

Figure 9. The number of students who applied Agile practices

4.1. Comparison Of Agile and Non-Agile Teams

In the project development process, the essential thing is team communication because almost all projects need to develop with coordination. Regarding survey results in Figure 10, students faced communication issues in both Agile and Non-Agile teams. The average of the non-Agile team is 4 but the Agile team has a better average which is 4.5. We can identify that an Agile team has better communication but not enough to improve to acceptable values. On the other hand, almost every group aligned that they faced communication issues in the project development process. Regarding verbal feedback from students, the main reason for that is randomized groups and the busy schedule.

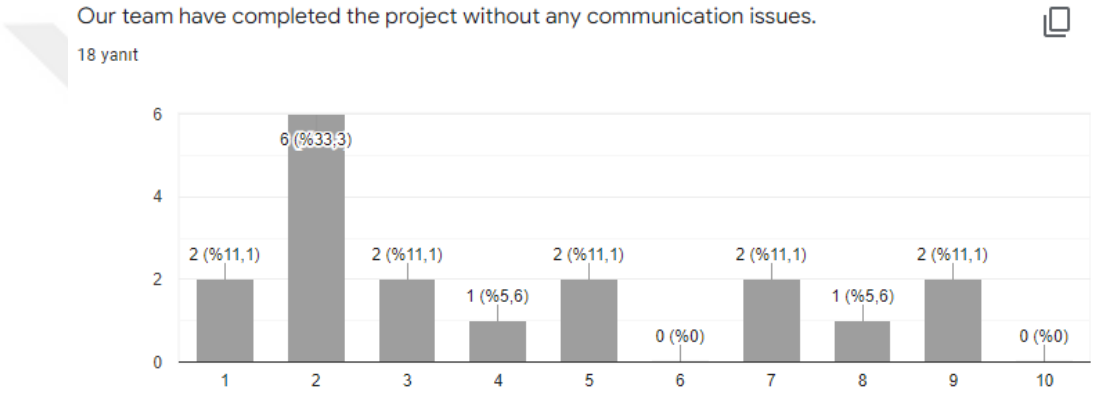

Figure 10. The number of students who faced communication issues

Participants of the Agile teams realized how important teamwork and communication play an important role in the project development process than non-Agile teams. In terms of the survey, the average of the non-Agile teams is 7.4 but the Agile team's 9.0. This result shows that the Agile application helped students to understand teamwork and communication are important. This is a good outcome to prepare university students for the industry.

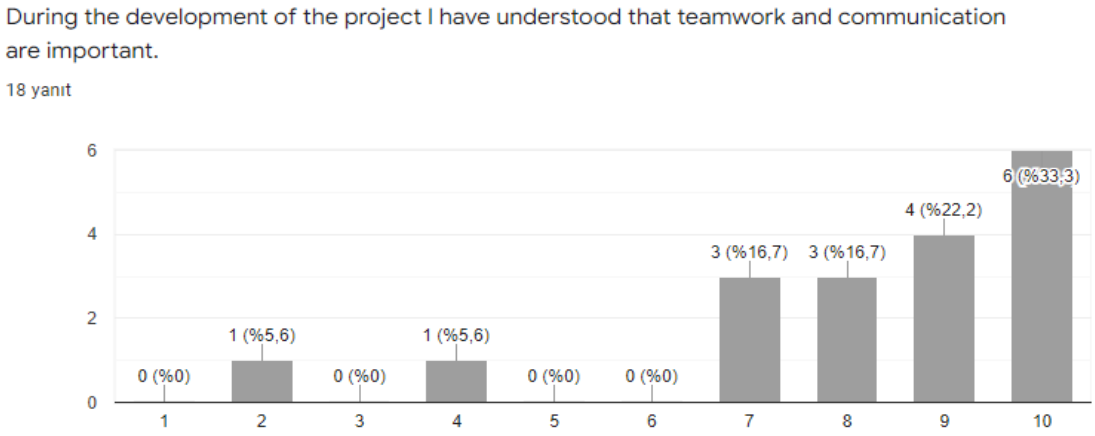

Figure 11. Results of the teamwork and communication

Another metric that we analyzed was team organization. The team should be self-organized in the Scrum framework. The results of this survey show the Agile team has a 6.42 average point and the non-Agile team has 3.6. This result indicated that Agile practices and ceremonies triggered the team to be organized.

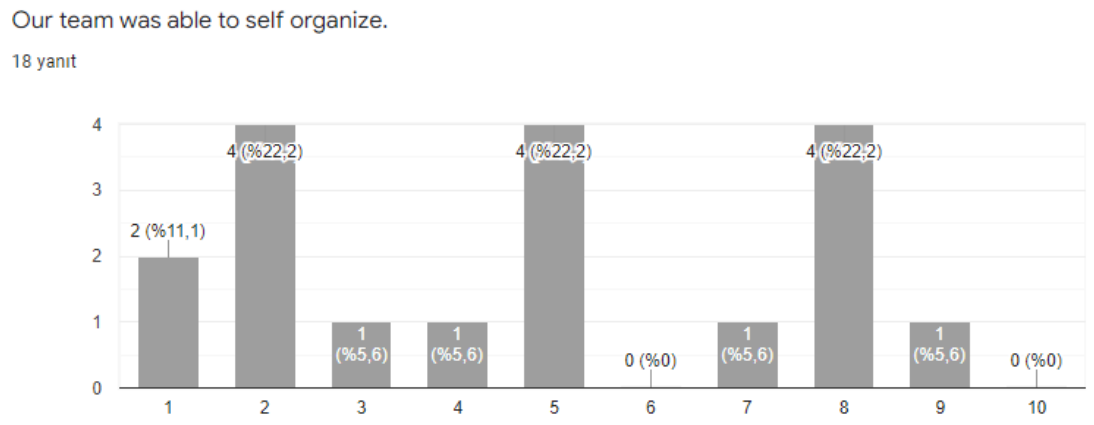

Figure 12. Results of the self-organization

According to Figure 10, Figure 11, and Figure 12 we can say that becoming a team was an important factor in the project development lifecycle. In Figure 13 teams reported their ability to become a team, Agile teams have 85% points however, non-Agile teams have 36%.

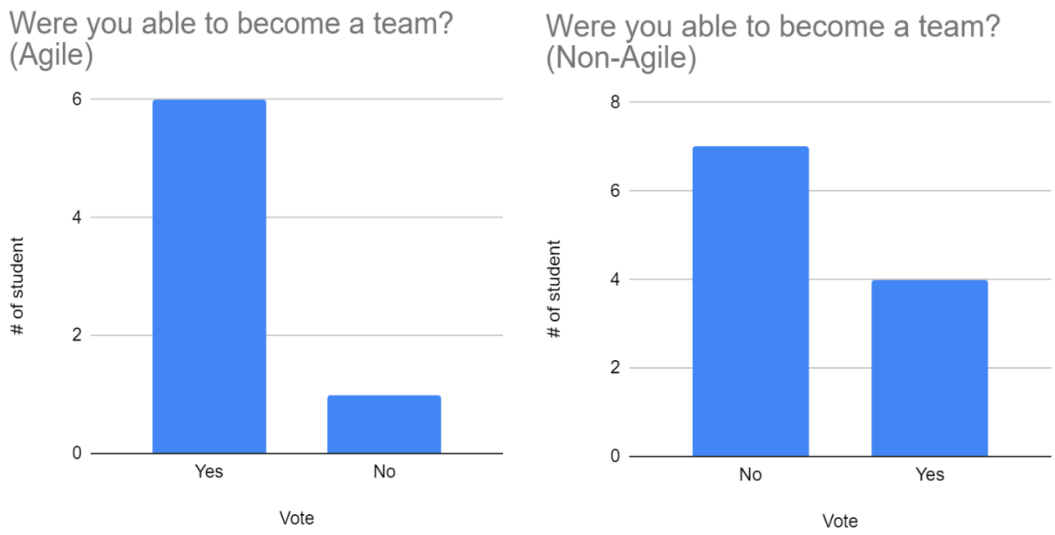

Figure 13. Results of becoming a team

Figure 14 shows that TPS did not help all students to organize their work. Non-Agile teams saw that TPS was not beneficial for organizing work with a 1.8-point average. However, the Agile team agreed that the TPS or another software management software helped the team to organize with 3.1 points out of 5.

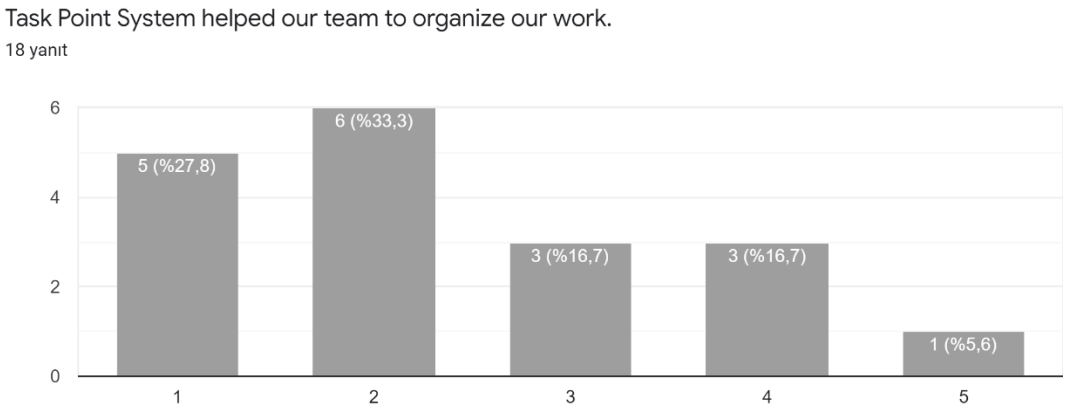

Figure 14. TPS contribution to team organizing

Groups reported the status of the completion percentage as Figure 15. The non-Agile team average was 75% and the Agile team completion was 75%. This result showed that there were no differences and positive effects of Agile on the completion of the project. Both teams completed projects successfully.

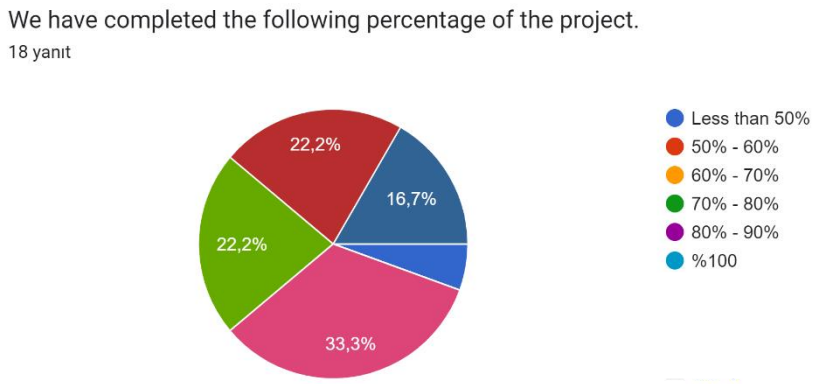

Figure 15. Results of the completion of the project

Another metric is the awareness of the project. We aimed to increase awareness with Agile practices, but the results show the opposite. The results demonstrated the non-Agile group revealed 7.9 points but the Agile group 5.2. Custom Agile implementation failed to increase awareness of functionalities and development of the project. We can say that non-Agile teams did not become a team because of the communication issues as shown Figure 12 and Figure 14. In addition, non-Agile teams complained the lack of contribution of the team members. These complaints showed that just one or two members of each group developed the project. We are assuming that just developer who developed the project were filled this survey because this is a volunteer based.

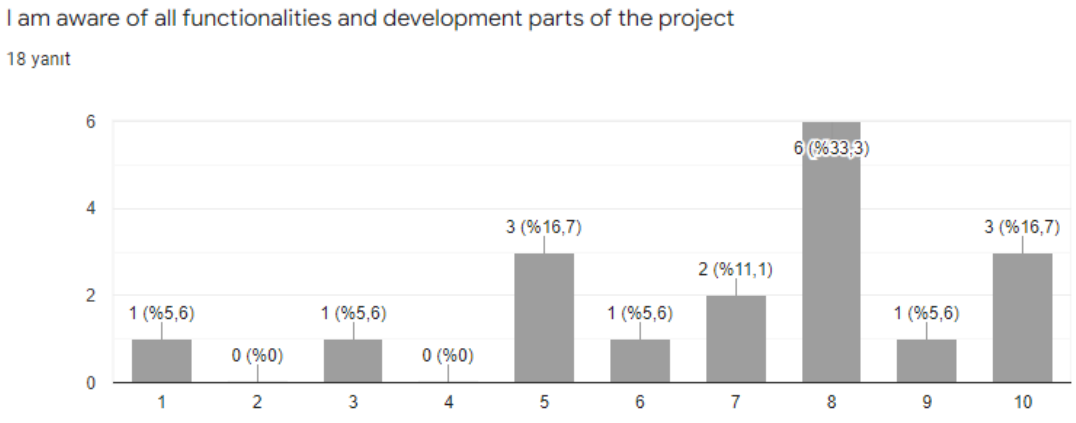

Figure 16. Results of the awareness

Another question was What do you think were the important negative factors in the completion of the project? In figure 17 Lack of contribution of other team members is mostly selected from students. The enormous number of students who selected lack of contribution were non-Agile students, we can say that our Agile approach helped the team to contribute more. The main reason was that one of the important Agile pillars was transparency. In TPS, students can track the contribution of the team members. Also, they were aware of the individual grades given by the lecturer with TPS help.

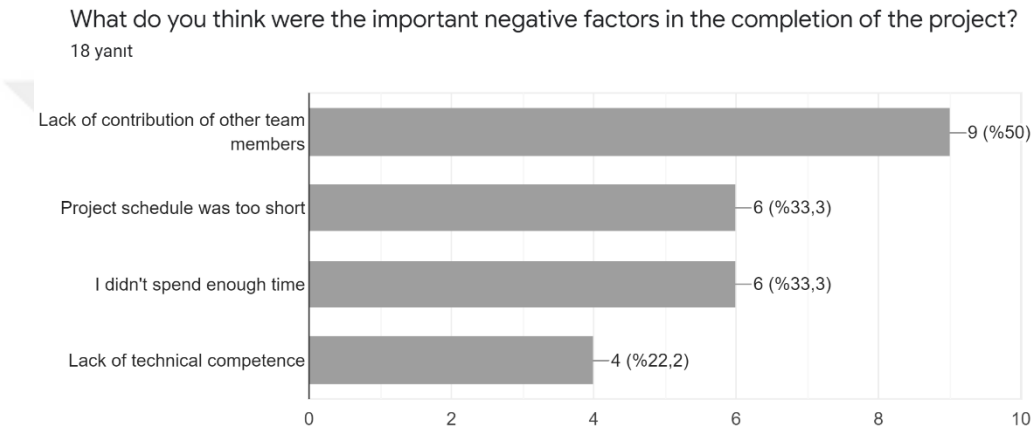

Figure 17. Results of the negative factors for completion

We analyzed negative factors in detail and Figure 18 shows that the non-Agile team mostly complained about a lack of contribution. TPS provided a good outcome from which students can comment to share with their group members in task sessions. Concerning these outcomes, both Agile and non-Agile teams identified the contribution of other team members. Another important complaint was the timeline of the project schedule, students reported completion of the project time was too short and they did not spend enough time on the project because of their busy schedules and other responsibilities.

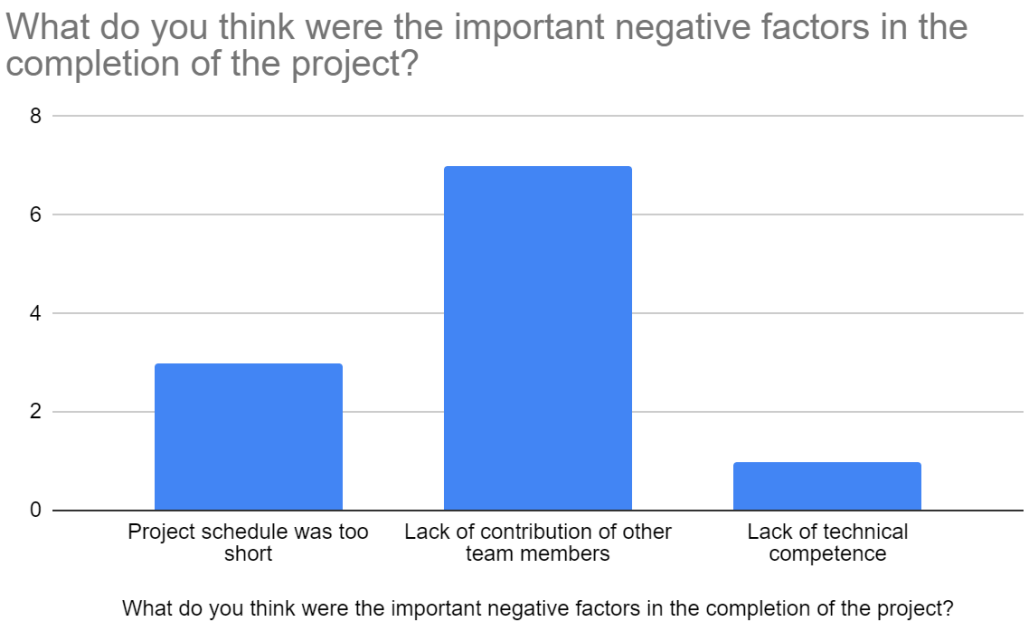

Figure 18. Negative factors for Non-Agile Teams

On the other hand, Agile team survey results show that in Figure 19 teams self-evaluated their performance in terms of spending time, technical competence, and project schedule instead of blaming the team. These results show the Agile team was confident about team contribution. According to their feedback, we can assume that Agile groups are better teams than non-Agile teams in Figure 19. Because Scrum Framework and TPS involve *Transparency* and *Inspection* that means every team member can see other team members' tasks and jobs.

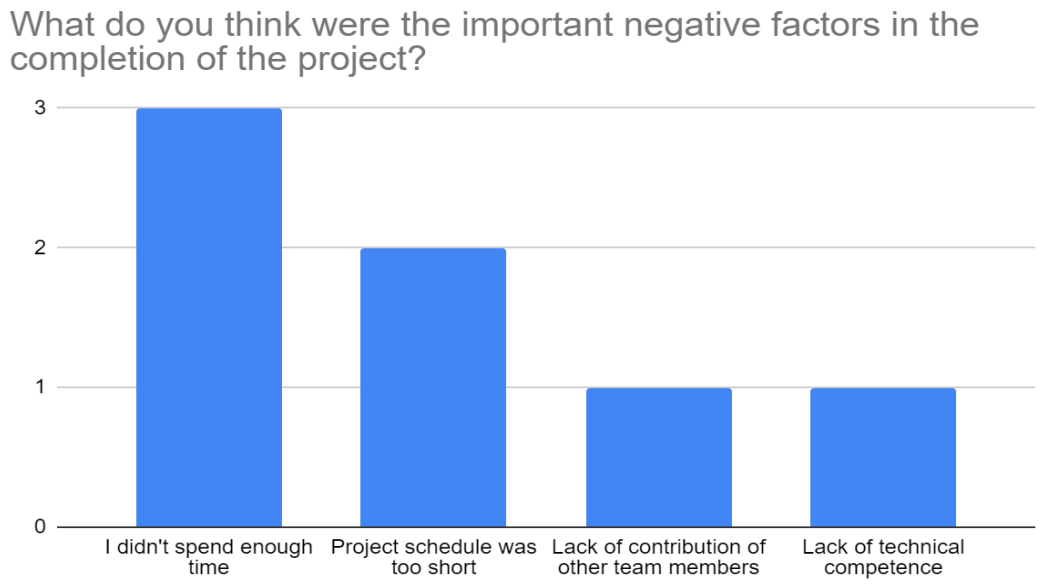

Figure 19. Negative factors for Agile Teams

4.2. The Effects of The Agile Methodology on Project Groups

In this section, the effects were evaluated for two volunteer groups which applied Agile principles in their project. For the measure, the Agile survey was analyzed. We collected responses on a 10-point scale, from ‘strongly disagree’ to ‘Strongly agree.’ Survey questions and the results were listed below

Q1: *“After using Agile practices for your project, would you recommend using them for the participants of next year’s course?”*

- All group members recommended Agile practices for next year's course; the average of the answer is 8.5 out of 10 points. These results show that Agile is liked by students and suitable to apply in university courses.

Q2: *“Did the Agile practices you have used helped to improve your teamwork skills?”*

- Agile has already demonstrated the increasing communication skills and teamwork abilities in Section 4.1. Figure 11 shows that team also agreed that teamwork skills are improved. Results also showed that the Agile team has a better average point for becoming a team.

Q3: *I am aware of all functionalities and development parts of the project*

- The average is 6.4 points in awareness of all parts of the project, we were expected to do more. It is possible that Agile teams realized they were not so good in every part of the software because they already know their technical capacity. Also, in Figure 20. they mentioned they did not spend enough time on to project. It is shown that they contributed to the project enough instead of non-Agile teams.

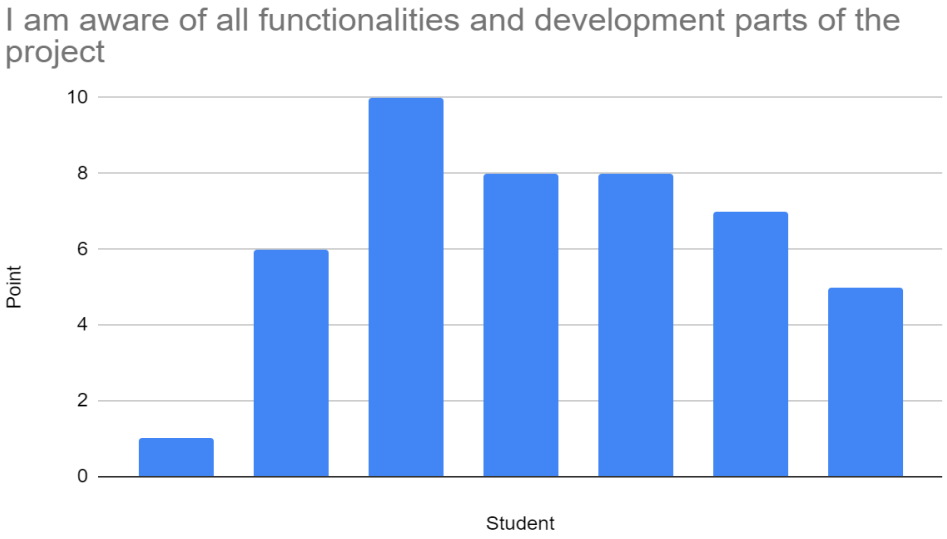

Figure 20. Agreement for awareness functionalities of the project

Q4: *"Do you agree that pair programming helped you to improve your technical skills?*

- At the end of the course, the students were much more confident in increasing technical skills with pair programming in Figure 21. Oral feedback was also positive about pair programming, developers mentioned swapping roles increased their knowledge about different tasks.

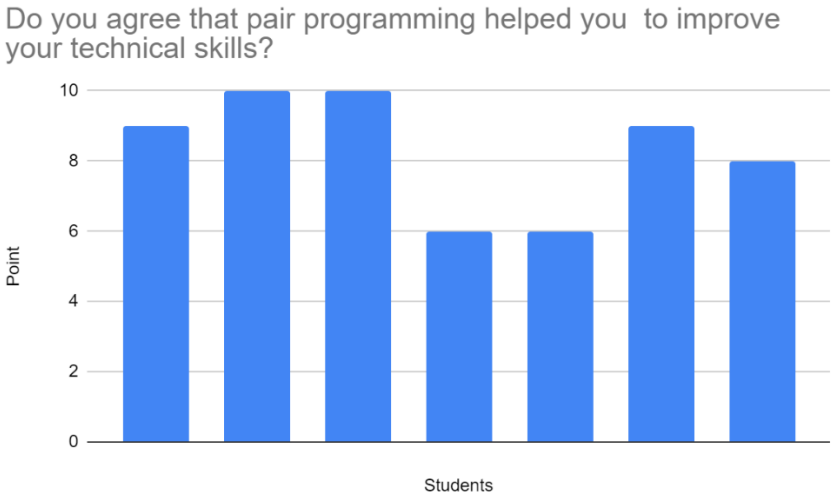

Figure 21. Agreement for XP helped to improve technical skills

Q5: *"Do you agree that using Agile practices encouraged you to participate more in the project?*

- According to Figure 22, Agile practices encouraged the team members to do more contributions. Students gave various verbal feedback during the development process, and they mentioned that Scrum ceremonies forced the team to work and follow the process. They also mentioned that enjoyed becoming an Agile team member and this situation brings their motivation up.

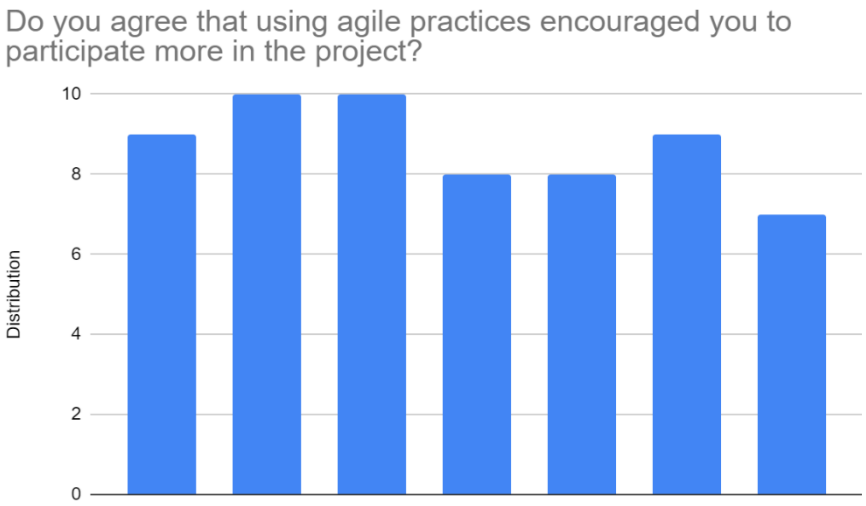

Figure 22. Agreement for Agile encourages to participation

CHAPTER 5: CONCLUSION

The negative and positive effects were evaluated in Section 4 and discussed in Section 5. Survey results showed that the Application of Agile Software development practices helped students in several ways. The first question was the Agile Survey "After using Agile practices for your project, would you recommend using them for the participants of next year's course?" Most of the students in Figure 23 agreed and recommended that the Agile methodologies were suitable for university course projects. As a result, both surveys and grades proved that Agile methodologies are suitable to be applied in the university setting.

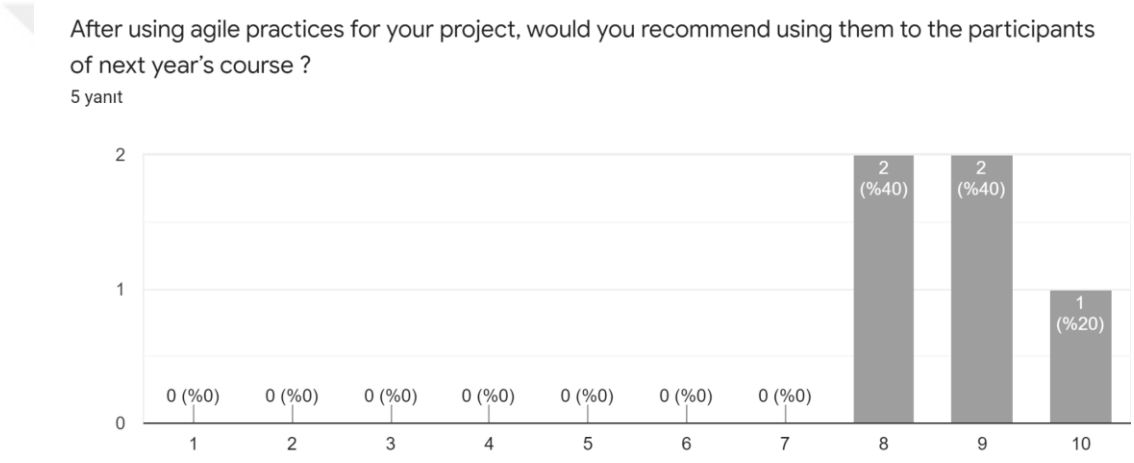

Figure 23. Results of the teamwork and communication

In the software industry, teamwork and communication are particularly important to complete a project successfully and important in the term projects. In this study, we analyzed how Agile affects individual contribution, team communication, and grading in real-life classes at university. Regarding the results of the surveys and the oral feedback, this study proved the benefits of the application Agile software development practices. All figures showed that Agile helped student in several ways. These results answered research question "What are the negative and positive effects of the Agile application on course projects?"

In capstone and senior projects, Agile is not use as effective because of the several reasons that are listed below:

1. Number of groups and students in course
2. Ability of student to apply Agile
3. Other responsibility of the students
4. Lack of time to lecturer
5. Daily meetings were challenging

For removing the obstacles, industry-like role play-based learning was applied to university classes. Agile, Scrum and XP practices were customized for students and lecturers. For assessment of the benefits of Agile, we collected data from surveys, a Task point system, and oral feedback.

For improving the results, teaching assistants or lecturers can manage the process like Scrum Master for all groups in courses instead of volunteer-based groups and they will be able to evaluate students. To evaluate the effect of the principles which are listed in methodology part can be evaluate with the exam questions and oral project presentation. To provide this, teaching assistants should attend to the development progress. To increase awareness of the project and increase technical skills, we will be able to increase the code review and pair programming sections to increase awareness of all parts of the project. Code review sessions will be able to monitor by the teaching assistant and the lecturer. This thesis proved that Agile applications are suitable to adapt and apply to university education. In the future, Agile methodologies will be applied to more groups with teaching assistants. Teaching assistants can lead the Scrum teams face-to-face after the pandemic will over. Other methodologies such as Waterfall, Spiral can be used and comparable in the future.

REFERENCES

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A. and Jeffries, R (2001) *Manifesto for Agile Software Development.* [Online] Available at: https://agilemanifesto.org/ (Accessed: 29 June 2022).

Bevan, J., Werner, L., and McDowell, C. (2002) *Guidelines for the use of pair programming in a freshman programming class*. In Proceedings of the 15th conference on software engineering education and training, pp.100–108. Washington, DC: IEEE Computer Society.

Blom, M. (2010) *Is Scrum and XP suitable for CSE Development?* Procedia Computer Science, vol. 1, no. 1, pp. 1511-1517,

Chen, Z. (2017) *Applying Scrum to Manage a Senior Capstone Project*, ASEE Annual Conference and Exposition Proceedings, (Columbus, Ohio), p. 27605

Cockburn, A. and Williams, L. (2002) *The costs and benefits of pair programming*. In eXtreme Programming and Flexible Processes in Software Engineering XP2000 pp. 223-247.

Dingsøyr, T. and Lassenius, C., (2016) *Emerging themes in Agile software development: Introduction to the special section on continuous value delivery*. Information and Software Technology, vol. 77, pp. 56–60.

Dogan, E. and Tüzün, E. (2022) *Towards a taxonomy of code review smells*. Information and Software Technology, vol. 142

Hanks, B., McDowell, C., Draper, D. and Krnjajic, M. (2004) *Program Quality with pair programming in CS1*. ACM SIGCSE Bulletin vol.3, pp. 176-180

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L. and Zander, C. (2011) *Pair programming in education: a literature review*. Computer Science Education, pp. 135–173.

Hundhausen, C., Agrawal, A., Fairbrother D., and Trevisan, M., (2009) *Integrating pedagogical code reviews into a cs 1 course: An empirical study,* SIGCSE Bull., vol. 41, no. 1, pp. 291–295.

Hundhausen, C.D., Agrawal, A., and Agarwal, P. (2013) *Talking about code: Integrating pedagogical code reviews into early computing courses*, Trans. Computer Education., vol. 13, no. 3, pp. 14-28

Jurado-Navas, A., Munoz-Luna, R. and Taillefer de Haya, L. (2014) *Scrum methodology in university classrooms: bridging the gap between academia and the business world*. in Experiencias en la Adaptación al EEES. pp. 321-330.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M. Kiefer, D., May, J. and Kahkonen, T. (2005) *Agile software development in large organizations*. Computer Practices. vol. 37. pp. 26- 34.

Muratoglu, M.K. (2021) *Assessment of Team Projects in Computer and Software Engineering*, Unpublished Master Thesis. Izmir University of Economics

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. and Balik, S. (2003) *Improving the CS1 experience with pair programming*. ACM Sigcse Bulletin. 35. pp. 359-362.

Oguz, D. and Oguz, K. (2019*) Perspectives on the Gap Between the Software Industry and the Software Engineering Education*, in IEEE Access, vol. 7, pp. 117527-117543

Oğuz K and Gül S. (2017), *Yazılım Mühendisliği Eğitiminde Takım Projelerinin Yönetimi ve Değerlendirilmesi*, In Proceedings of the 11th Turkish National Software Engineering Symposium, Alanya, Turkey pp. 184-195.

Rico, D. and Sayani, H. (2009) *Use of Agile Methods in Software Engineering Education*. Agile Conference. IEEE. pp. 174-179.

Rong, G., Li, J., Xie, M., Zheng T. (2012*) The Effect of Checklist in Code Review for Inexperienced Students: An Empirical Study*, IEEE 25th Conference on Software Engineering Education and Training pp. 120-124.

Sadowski, C., Söderberg, E., Church, L., Sipko, M. and Bacchelli, A. (2018) *Modern code review: a case study at google*. pp. 181-190

Sripada, S. Reddy, Y., and Sureka,A. (2015) *In Support of Peer Code Review and Inspection in an Undergraduate Software Engineering Course*, 2015 IEEE 28th Conference on Software Engineering Education and Training, 2015, pp. 3-6

Stahr, L. (2022) *Comprehensive implementation of agile principles in a computing capstone design course*. Master of Computer Science, Miami University, Computer Science and Software Engineering.

Takeuchi, H. and Nonaka, I. (1986), *The New New Product Development Game*, Harvard Business Review, vol.64, pp. 137-146.

Williams, L. and Kessler, R. (2002) *Pair Programming Illuminated*., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Williams, L., Kessler, R, Cunnigham,W and Jeffries,R. (2000) *Strengthening the case for pair-programming*. IEEE Software, 2000. vol. 17 pp. 19-25.
[VisualParadigm]. (2022, June 10) *Agile for small teams* [Web-based visual]. Available at: https://www.visual-paradigm.com/Scrum/from-small-teams-to-scaling-Agile/

APPENDICES

Appendix A – To Do List

Daily meetings: Let's store the difficulties you faced, dependent jobs, and progress status in the Excel file when you implement. I prepared name/day spreadsheets in an Excel file on the cloud. I created WhatsApp groups for our team to discuss daily issues and tasks. Please attend this group and give your comments daily.

Weekly meetings: Let's store our weekly meetings on Google Drive. We need to schedule once a day a week. We need to write:

- What we did
- What were the obstacles and challenges
- What we are going to do
- What are the risks

Pair programming: Let's schedule pair programming sessions once a week. The session should be at least 1 hour. More sessions are welcome.

- Please do not forget to change your pairs every week.
- Please do not forget to change the driver-navigator role.
- Please do not forget to log your work on a spreadsheet.

Functional testing: Unit testing is preferred but it's not mandatory. If your team will not implement unit testing, you need to perform a functionality test at the end of each sprint (2 weeks).

How to perform functionality tests?

Validate the functionality of the requirements in Graphical User Interface. Give input to the application and check whether the output is expected or not for every unit and function. Please do not forget the list down your steps.

Example: Consider the calculator application and steps for addition.

- Input: 3+6
- Expected value:9
- Actual value: 8
- Note: the result is not expected, review the code

Example 2: Consider an application that needs a Turkish Nationality ID number

- Input: 123456789
- Expected: TC number length should be 11 dialog boxes
- Actual: Accept 123456789 ID number

Note: The application must show a dialog box to the user TC ID must be 11 lengths

Example 3: Consider a tax application which has contains a division

- User tries to divide tax to 0
- Input: tax value / 0
- Actual value: Fatal error
- Note: Handle divide by zero exception.

Code review:

The team must schedule a code review session every week for at least 15 minutes for knowledge transfer to other team members.

The team must schedule a code review session at the end of each spring with the Scrum Master for architectural and high-level code review. The Scrum Master will give suggestions to the team to improve code quality and engineering perspective.

Appendix B- Survey Questions

Survey questions for analyzing suitability of Agile methodologies and negative and positive effects.

1. We have worked with Mert Akkanat and used Agile practices as a team.
2. During the development of the project, I have understood that teamwork and communication are important.
3. Our team have completed the project without any communication issues.
4. Our team was able to self-organize.
5. We have completed the following percentage of the project.
6. I am aware of all functionalities and development parts of the project
7. How ready do you feel to develop a software project in business life?
8. Were you satisfied with the scores you have received in your project?
9. I had a good time during the project development process.
10. What do you think were the important negative factors in the completion of the project?
11. How many course projects have you completed so far? SE302 does not count.
12. How many real (non-course) projects have you completed so far?
13. In how many of these projects, both course and real, have you worked as part of a team?
14. In the course and real projects besides the one in SE302, how many members did you work with at most?
15. How many hours did you spend on this project?
16. What did you spend this time on the most?
17. Which development method did you use?
18. Was the development method that you have used helpful?
19. Were you able to become a team?
20. Would you like to develop another project with your team in the SE302 course?
21. How much do you think you contributed to the project?
22. Select the hardest parts of teamwork.
23. Task Point System is easy to understand and use.
24. Task Point System helped me see my contribution to my team.
25. Task Point System helped our team to organize our work.

26. I would like to use the Task Point System again because it provides the project grading to be fair among the members of the team.
27. The lecturer was helpful and helped our team to create a better project.
28. We have gained experience from working with the lecturer.
29. If there is anything you would like to say, please use the text box below. This is only for feedback, it will not be used in the research.

Appendix C- Ethics Committee Report

SAYI : B.30.2.İEÜFMB.0.05.05-020-050 11.04.2022

KONU : Etik Kurul Kararı Hk.

Sayın Mert Akkanat,

"Application of Agile Software Development Practices in Software Engineering Education" başlıklı çalışmanızın etik uygunluğu konusundaki başvurunuz sonuçlanmıştır.

Etik Kurulumuz 14.03.2022 tarihinde sizin başvurunuzun da içinde bulunduğu bir gündemle toplanmış ve projenin incelenmesi için üç kişilik bir alt komisyon oluşturmuştur. Projenizin detayları alt komisyon üyelerine gönderilerek görüş istenmiştir. Üyelerden gelen raporlar doğrultusunda Etik Kurul 11.04.2022 tarihinde tekrar toplanmış ve raporları gözden geçirmiştir.

Sonuçta 11.04.2022 tarih ve 44 numaralı Etik Kurul toplantısında "Application of Agile Software Development Practices in Software Engineering Education" başlıklı çalışmanızın etik açıdan uygun olduğuna oy birliği ile karar verilmiştir.

Gereği için bilgilerinize sunarım.

Saygılarımla,

Prof. Dr. Ismihan Bayramoğlu
Fen ve Mühendislik Bilimleri
Etik Kurulu Başkanı