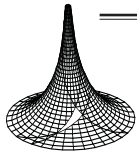

20/1 (2022), 1–32
DOI: 10.5485/TMCS.2022.0534
tmcs@science.unideb.hu
http://tmcs.math.unideb.hu

Teaching
Mathematics and
Computer Science

Teaching agile operation and leadership through linked university courses

Enikő Ilyés

Abstract. Agile software development methods, especially Scrum, are commonly used in software development companies. For this reason, our goal was that our undergraduate students gain experience as Scrum development team members and our master's students as agile leaders. To this end, we had redesigned and linked an undergraduate and a master's course, and launched the new course in the spring of 2021. The success of our approach was confirmed by a questionnaire survey of 86 undergraduate and 27 master's students. A/B testing was also performed. Our approach is a novelty compared to solutions where the Scrum Master is a course member, an instructor, or a university employee. In addition to being resource-efficient, it also offers master's students an unparalleled opportunity to develop agile leadership skills.

Key words and phrases: Scrum, Scrum Master, agile leadership, teaching methodology, university level.

MSC Subject Classification: 97U50.

1. Introduction

Effective software development methods are part of software engineering science. Therefore, faculties with software development profile should teach software development methodologies.

The author thanks Eötvös Loránd University's Department of Software Technology and Methodology for its contribution to this research.

Currently, agile methodologies are the most common, with the Scrum methodology being the most widely used. According to the 15th annual State of Agile Report survey (*15th annual State of Agile Report*, 2021), 81% of firms applying agile methodology use the Scrum methodology or a hybrid. Thus, teaching Scrum at universities is justified.

Scrum Guide (Schwaber & Sutherland, 2011) defines roles, events and artifacts of Scrum in only 14 pages. Still, teaching Scrum involves many challenges. Adoption of Scrum needs soft skills and true understanding of agile values (Kropp, Meier, Mateescu, & Zahn, 2014), which are hard to develop.

Teaching Scrum is mostly realized by adaptation, in the form of a simulation project. Often, a software development project, or a collaboration project like LEGO City construction (Paasivaara, Heikkilä, Lassenius, & Toivola, 2014), paper city construction (Hof, Kropp, & Landolt, 2017), etc., serves as a simulation project. Another typical approach is to offer Scrum courses focusing on individual Scrum roles. Famous Scrum teaching institutions, such as Agile Alliance (*Agile Training*, 2021) and Scrum.org (*Professional Scrum Training*, 2021), have Scrum basics training for development team members, Scrum Master training and Product Owner training, at both beginner and advanced levels. Role-based education is not yet common at universities.

We believe it is important that university students study Scrum at both undergraduate and graduate levels, with the emphasis on studying different perspectives at each level. Our goal is that our undergraduate students gain experience as Scrum development team members and our master's students as agile leaders. Undergraduate programs should provide experience of being part of a development team, allowing for more seamless engagement in Scrum teams after graduation. At that time, their professional knowledge is best suited for the developer role. During a master's program, they acquire a higher level of technical knowledge than an average developer. They may even be interested in other roles, for example, in team leader roles. In order to best support such interests, a master's program should convey leadership knowledge and skills.

Based on our believes, our research questions are:

(1) How can we provide undergraduate students with experience in Scrum methodology?

(2) How can we provide master's students with experience in agile leadership?

(3) How can we realize both goals in a resource-efficient way?

In our approach, we have redesigned two classical university courses by combining them. Undergraduates gain Scrum development experience, while master's

students agile leadership experience, all in a resource-efficient manner. We present the redesign in details in our Section 'Case study'.

The rest of the paper is organized as follows. In the next Section, we briefly summarize the basic concepts of agile methodology and Scrum we build on in this article. In Section 'Related work', we analyze the status quo of how agility and Scrum education are implemented in the world, with a particular focus on university education. Section 'Case study' presents our approach, starting from the framework of university courses, which gives context to our case study, ending in the exact schedule. Section 'Experiences' discusses our experiences with the case study we conducted. Section 'Measurements' summarizes the results of a questionnaire survey conducted among the participants of the case study. Finally, Section 'Lessons learned' summarizes our conclusions.

2. Basic concepts

A common feature of all agile methodologies is that they implement all four concepts of Manifesto for Agile Software Development (*Agile Training*, 2021). They focus on the people engaging in the development process and their interactions, involve the customer into the process, appreciate working code instead of plans, and encourage participants to be open to change.

Scrum supports agile principles through the events, roles and artifacts defined in the Scrum guide (Schwaber & Sutherland, 2011).

Scrum divides the entire length of the project into 2-4 week working periods called Sprint. Each Sprint contains a single Planning, Review, and Retrospective meeting, as well as a meeting every day, called Daily Scrum. The Sprint begins with Planning, where tasks to be performed by the development team during the Sprint Planning are decided. At the Daily Scrum meetings, team members assess in 15 minutes how well they are performing with the Sprint's tasks, whether there is anything hindering them in achieving their Sprint Goal. At the end of the Sprint, the developers present the results of the Sprint to the customer on a Review meeting, get feedback on it, and discuss the next steps. The Sprint concludes with a Retrospective, where the team evaluates the progress of the work and identifies opportunities for optimization for more effective work organization. Figure 1 summarizes Scrum events, which support transparency, inspection, adaption, that is, the three Scrum pillars. Transparency ensures that we base our decisions on correct and comprehensive data, while inspection and adaptation guide us in reaching our goals.

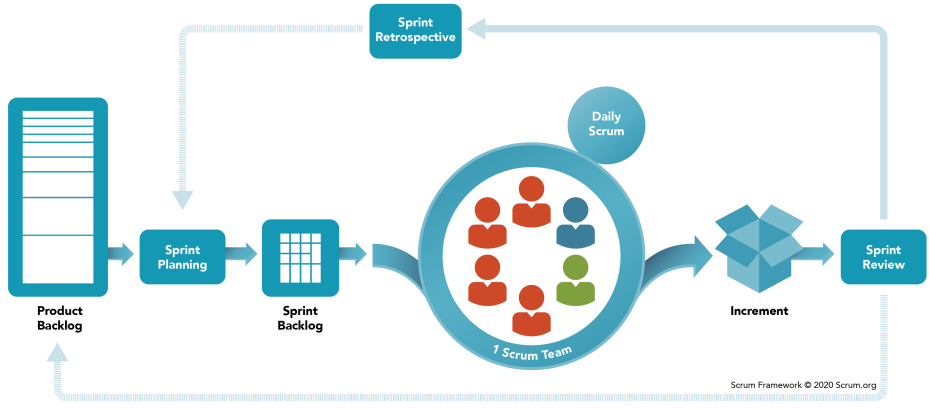

Figure 1. Scrum framework

The Scrum Master is responsible for the effective implementation of Scrum events. He/She is the serving leader of the team, coaching his/her team to function in the most self-organized and efficient way possible. In order to focus on creating business value, the Scrum Team also has a Product Owner. He/She organizes and prioritizes the costumer's needs, helps the team in understanding them, and then evaluates the team's performance at the end of the Sprint. It is very important to demonstrate a working code at the end of the Sprint contrary to a slideshow. The working code is called Increment, one of Scrum artifacts.

Scrum artifacts also include the Product Backlog and Sprint Backlog. The former is a prioritized list of the costumer's expectations. During Planning, the team selects high-priority items from the Product Backlog to be part of the Sprint Backlog. A Sprint Backlog contains the tasks that contribute to reaching the Sprint's goal.

Apart from events, roles and artifacts, Scrum also highlights five values such as openness, courage, respect, focus, commitment. Presence of these values can make working according to Scrum efficient.

3. Related work

Software developers are prepared for using Scrum most commonly with hands-on training, as we can see in the case of agility-promoting institutions holding courses on agile and Scrum. Typically, companies send employees to these

courses. For example, the Agile Alliance (*Agile Training*, 2021) organizes the following courses: Scrum Developer Certified Training, Certified Scrum Master Course, Certified Scrum Product Owner Course, Coaching Agile, Scaled Professional Scrum. At Scrum.org (*Professional Scrum Training*, 2021) we can see similar courses: Applying Professional Scrum for Software Development, Professional Scrum Master I and II, Professional Scrum Product Owner I and II, Professional Agile Leadership Essentials, Scaled Professional Scrum, etc.

These courses are mostly role based. The training for Scrum developers covers topics such as: agile mindset, the fundamentals of Scrum and how to apply it correctly, in which kind of teams to apply it, how to be a great Scrum Team member, identifying common pitfalls and avoiding them. The knowledge conveyed by such training is something members need to be aware of in order to be easily integrated into a Scrum Team. There are separate courses for the other two Scrum roles, the Scrum Master and Product Owner roles, often available at both beginner and advanced levels. Scrum Master training typically include the following topics: theory and principles behind Scrum and empiricism, how each part of the Scrum framework ties back to the principles and theory, what "Done" means and why it is crucial to transparency, the importance of self-managing teams, interpersonal skills needed, the Scrum Master role, behavior shifts required to be a Scrum Master. Courses usually last 2 days. Participates can gain a certification. Costs of training typically range from around €1,200 to €1,500.

Another solution in the industry is to provide agile training within a company with the help of external or internal trainers. Typically, the company as customer employs the solution as part of an agile transformation. In this case, we do not have precise information on training topics. It is likely that such training covers topics similar to the general agile training topics, tailored to the customer.

Unlike the industry, the university does not feature education along different Scrum roles. Agile methodologies are mostly applied on a practical course, for example, a project work of undergraduate students, where students work together as a Scrum Team and develop a software. According to Mahnič (2015), this is the most common form of educating agile methodologies at universities.

Cases differ the most in role assignment and Sprint length. It is common for the Product Owner role to be assigned to a teacher or other expert, as in the case of Damian, Lassenius, Paasivaara, Borici, and Schröter (2012), Kropp and Meier (2013), Mahnič (2011), Paasivaara, Lassenius, Damian, Räty, and Schröter (2013), Zorzo, de Ponte, and Lucrédio (2013). But there are examples of Product Owner students as well: Scharf and Koch (2013), Reichlmayr (2011). In the case

of the Scrum Master, it is more typical that the role is played by a student, as in the case of Damian et al. (2012), Kropp and Meier (2013), Paasivaara et al. (2013), Rodríguez, Soria, and Campo (2011), Scharf and Koch (2013). But there are also examples where teachers take the role, such as in the case of Mahnič (2011), Reichlmayr (2011).

Experience has shown that it is generally advantageous for an instructor to fill Scrum roles, because they can play their roles with more authenticity thank to their experiences. The same is not guaranteed if students fill Scrum leader roles, but in return the student has the opportunity to develop skills. However, we have to pay attention to the fact that some Scrum Master students wait too long for their teams to solve problems among themselves, while others intervene too soon and thus do not support self-management (Scharf & Koch, 2013). That is why, according to Mahnič (2015), student Scrum Masters should be supported by a teacher: "one of the instructors should still play the overall Scrum Master supervising the whole process." Scharf and Koch (2013) present a method that introduced several different supporting roles. They admit that their method needs a large staff, up to 11 supervisors for approximately 60 students. "The professor, two research assistants supervising the course in general, two research assistants playing the role of clients, and six student research assistants supervising each team as their tutor."

Apart from software development projects, another typical way to teach agile methodologies is to play educational games, such as in the case of Paasivaara et al. (2014), Hof et al. (2017), Ramingwong and Ramingwong (2015), Von Wangenheim, Savi, and Borgatto (2013). Games include Lego City construction (Paasivaara et al., 2014), paper city construction (Hof et al., 2017). According to Mahnič (2015) and Kropp et al. (2014), Scrum games are suitable for demonstrating the practical operation of Scrum in a short time.

Scrum, as an agile methodology, gives lot of space for cooperation and communication. Applying Scrum creates a wide variety of situations, and we cannot possibly prepare students for all of them. Thus, students are better served by teaching values which they can use as compasses in different situations. Teaching values is difficult because it is a 'deeper layer' of a person than skills and behavior (Dilts, 1996). Values drive behavior when we have the right skills.

Martin Kropp and colleagues emphasize the importance of developing communication skills and agile values (see (Kropp et al., 2014), (Kropp & Meier, 2013), (Hof et al., 2017)). In their Agile Competence Pyramid model, they place the acquisition of Agile Values at the highest level of cultivating agility (Kropp &

Meier, 2013). Three principles are offered for such a high level education of agile methods (Kropp et al., 2014):

(a) Make personal experience by working in an agile production environment.

(b) Cooperate in an agile group with a substantial amount of social exchange and discourse taking place.

(c) Develop and discuss agile values and attitudes.

Regarding Scrum in university education, another typical challenge is evaluation of students. While the basic unit of Scrum is a team, and we evaluate performance of our teams, it is the individuals who need to be evaluated in university education. The tension between the two ways of evaluation has been focused by several researchers, and is not addressed in our article.

4. Case study

In order to teach Scrum at the Faculty of Informatics of Eötvös Loránd University on a high level, we redesigned two traditional courses. The transformation took place in the spring of 2021. Our new courses are presented below.

4.1. Course details

4.1.1. Software Technology – practice

- Dedicated to: undergraduate students, fourth semester
- Duration: 90 minutes/week
- Type: mandatory
- Goal of the course: to provide students with an experience of the entire project lifecycle and collaboration on developing a larger software project.
- Previous knowledge of students: Before this course, students already learned programming on other courses. They created smaller game applications alone.
- Related courses: This practical course is the pair of the Software Technology lecture. The lecture includes the following topics in the given order: software development process, requirement engineering, object oriented design, object-oriented design patterns, project management tools, version control, build systems, verification and validation, unit tests, continuous integration and delivery, test driven development, clean code, multithreading, Agile methodologies.

4.1.2. Software Development in Practice – practice

- Dedicated to: master's students, second semester
- Duration: 90 minutes/week
- Type: optional
- Goal of the course: To enable students to consciously and proactively shape software development processes.
- Previous knowledge of students: Students already learned the basics of managing a software development process (theory). They most likely already developed a software as a team member during their BSc studies or at their workplace.
- Related courses: This practice course is a follow-up course of the Project Management in IT course. This lecture includes the following topics in the given order: definition of the project, project diamond, corporate culture, organizational models, project life cycle, Scrum methodology, personality models, customer reception at the highest level, effective team, leadership models, change management, burnout prevention. We considered designating the Software Technology course as a prerequisite for the Software Development in Practice course, but we rejected the idea. Undoubtedly, this relation would be useful for students. However, we did not want to exclude students who had not completed their undergraduate studies at our university. As most of our master's students have already gained teamwork experience at an undergraduate course, internship or workplace, we did not experience problems stemming from the lack of a prerequisite course.

4.2. Course implementation in 2021 – Our new approach

Software Technology students (undergraduates) are divided into teams of 3. (In describing the case study, the word 'team' always refer to a three-person team of developers. Our definition of 'team' is unlike the Scrum guide, where the team includes the Scrum Master and the Product Owner as well.) Each team has to develop a strategic game designated by the instructor in a programming language of their choice. All teams of a practice develop the same video game, but tech stack is not fixed, and can be chosen freely. They have to adopt Scrum during their teamwork. GitLab is used for version control, continuous integration, continuous delivery. GitLab's Issue Board is highly recommended for task management, but other tools, such as Trello, are allowed as well.

Every team has a Scrum Master, who is one of the master's students. This student does not undertake coding tasks, just gives support to the team in operational issues and is a servant leader. The Product Owner role is played by the teacher of the Software Technology course.

The semester is divided into four Sprint periods. This means that on approximately every fourth class the teams make a presentation of their Sprint achievements in front of other teams and the teacher. Teams also hold a Weekly Scrum on intermediate practices with their Scrum Master. After Sprint Reviews, teams are advised to hold an extra meeting where the Sprint Retrospective and the Planning of the next Sprint can take place.

The schedule of the semester is presented in Table 1.

	SWT – Software Technology practice	**Weekly activity of the Scrum Master student between the classes**	**SDP – Software Development in Practice**
1	Description of the project	Getting to know the Product Owner and the Team	Preparation training for Scrum Master role
2	Weekly Scrum Team building	Weekly Scrum	Preparation training for Scrum Master role
3	Weekly Scrum	Weekly Scrum, Preparation of the Sprint Review	Scrum Master exam Consultation (Subgroup A)
4	Sprint Review (use cases, use-case diagrams, class-diagrams)	Sprint Review, Retrospective, Planning	Consultation (Subgroup B)
5	Weekly Scrum	Weekly Scrum	Consultation (Subgroup A)
6	Weekly Scrum	Weekly Scrum, Preparation of the Sprint Review	Consultation (Subgroup B)
7	Sprint Review (Prototype 1 – 30% of use cases, use of Git tool)	Sprint Review, Retrospective, Planning	Consultation (Subgroup A)
8	Weekly Scrum	Weekly Scrum	Consultation (Subgroup B)
9	Weekly Scrum	Weekly Scrum, Preparation of the Sprint Review	Consultation (Subgroup A)

10	Sprint Review (Prototype 2 – 90% of use cases, use of CI and CD, unit tests)	Sprint Review, Retrospective, Planning	Consultation (Subgroup B)
11	Weekly Scrum	Weekly Scrum	Consultation (Subgroup A)
12	Weekly Scrum	Weekly Scrum, Preparation of the Sprint Planning Review	Consultation (Subgroup B)
13	Sprint Review (Final version Clean Code, all unit tests)	Sprint Review, Retrospective	Closing

Table 1. Schedule of the semester

4.2.1. Preparational phase

On the first practice, Software Technology students have the possibility to form teams of three on their own or to be assigned to a team by the teacher. The assignment is based on the students' self-declared knowledge on different aspects of software development such as: known programming languages (Java, C ++, C#, .NET), experience with version control systems, experience with continuous integration systems, experience in developing software as part of a team. In compiling the teams, the following principles are considered in the given order: each team should include a member who has experience with version control systems; each team member should have at least good knowledge level in the same programming language; each team should include a member who has experience with continuous integration systems; each team should include a member who has experience with software development in a team. On the first practice, all the teams are formed and then introduced to their Scrum Master.

Software Development in Practice students are prepared to fulfill the Scrum Master role not only in general but in this specific adoption of Scrum. For this reason, the first two practices are dedicated to a preparation training. The training presents models, elements of the following topics: Agile leadership, Scrum theory, Scrum overview, Scrum roles, powerful questions, situational leadership, Scrum values, Scrum Guide, Scrum events, Retrospective games and techniques, Timebox, Scrum products, giving feedback, icebreaker games. During the training, the focus is on understanding the main idea of agile leadership and on giving an insight into how this can be realized in this case. As part of the training, situations are outlined which are likely to occur. Students have to argue about the

correct attitude of the Scrum Master in the given situation. In the case of every situation, an example behavior of the Scrum Master that seems to be helpful is given. But in reality, these do not improve the self-organization of the team, so in fact, they are contraindicated behaviors. During the exercises, it is clarified that there are many good solutions in the given situations. However, all good solutions represent agile principles and values.

The discussed situations are:

Case 1. The team tells you they don't know exactly how the Product Owner envisioned the app.
Counterexample: You ask the Product Owner what he/she exactly wants and then you tell the team.

Case 2. The team signals that they don't understand version controlling, they don't even know how to get started.
Counterexample: You configure the version control system as best as you can, and then you present the team a simple guide on how to use it.

Case 3. The team signals that one of the members does not contribute to the project.
Counterexample: You assign tasks to every team member and even more tasks to the problematic person. You micromanage the team.

Case 4. One of the team members is not present at the Weekly Scrum meeting.
Counterexample: You don't worry, he/she will definitely come next time. You only worry if he/she's been gone for 3 weeks.

At the end of the preparatory phase, on the third practice, students must pass a Scrum Master test, similar to Scrum.org (*Professional Scrum Training*, 2021) PSM 1 test. The test contains 25 questions. Each question offers 4 answers, of which exactly 1 is correct. Students have 20 minutes to complete the online (Canvas) test.

4.2.2. Main phase

All undergraduate students progress according to the same schedule. Scrum Master students are divided into two subgroups, and each subgroup is required to meet with the teachers every 2 weeks. The division into two subgroups is necessary, because the observations of 27 Scrum Masters cannot be meaningfully discussed during the 90-minute session per week. If a Scrum Master student needs help in a short time, he/she can join the consultation any week or ask for help in the form of a group or personal message.

The consultations usually begin with the instructor asking some guiding questions about the teams, preferably one that evokes analysis of teamwork from a different perspective. Each Scrum Master is given the floor to answer the questions or to report phenomena that he/she considers more important in his/her team at the moment. Scrum Master students are asked to discuss arising problems, exchange experiences, to form a professional community. The instructors, however, give feedback to all Scrum Master students on every meeting. The Scrum Master subgroups of 12-13 people are able to discuss what was observed in their teams over the last 2 weeks in the given 90 minutes.

In addition to discussions on meetings, Scrum Master students have to report about their team in written form every week. The written report has to be visible to all team members, thereby promoting transparency. The recommended content of the report is shown in Table 2. It contains 4 sections: basic information about the team and its weekly events; team impediments and their solutions; measurements of teamwork aspects and their explanation; and other important observations by the Scrum Master. The report has two aims: to draw attention to Scrum perspectives from week to week, and to monitor the well-being of the teams.

4.2.3. Closing phase

Software Technology practice closes with a last review where teams present the final product. Teachers and colleagues from the same practice are present.

In the last week, all Software Development in Practice Scrum Master students meet for a last time and share in a few sentences what they have learned from the course and how the course could be further developed.

4.2.4. Evaluation of the students

Undergraduate students are evaluated based on their developed game software. At the end of each Sprint (Week 4, 7, 10, 13) every team gets a sub-score. The sum of the sub-scores gives the final rate. All sub-scores are given equal weight in the final evaluation and must reach a certain minimum number. Each student receives an assessment based on their contribution to the product, which is tracked by GitLab.

Scrum Master students are evaluated according to three components. First their theoretical knowledge, which is measured by the Scrum Master test on the third practice. This counts as a quarter of the final evaluation. Another quarter of the final evaluation is based on the developer team's feedback on the Scrum

Team name:	*XYZ*
Week:	*Week 3*
Type of event:	*Planning / Weekly Scrum / Review / Retrospective*
Absent persons:	*X*
Sprint goal completeness percentage (%):	*80%*
Duration of the event:	*25 minutes*
Impediment 1:	
Originator of the solution:	
Solution:	
Impediment 2:	
Originator of the solution:	
Solution:	
Cooperation in the team:	(1=none – 5=completely satisfactory)
Explanation:	
Communication in the team:	(1=none – 5=completely satisfactory)
Explanation:	
Motivation of the team:	(1=none – 5=completely satisfactory)
Explanation:	
Procrastination of the team:	(1=very typical – 5=not typical at all)
Explanation:	
Workload sharing of the team:	(1=none – 5=completely satisfactory)
Explanation:	

Table 2. Scrum Master weekly report template

Master student. This is assessed by a questionnaire which has to be filled out by each team member anonymously at the end of the semester. The remaining half of the final evaluation is the instructor's rate on the Scrum Master's semester long activity. This covers his/her contribution to Sprint Planning, Review, and Retrospective events, detection and management of impediments in the team, continuous development of the team, and weekly brief status reports about the team.

5. Experiences

5.1. Experiences of the course implementation in 2021

We showed above how we had designed two courses to achieve our goal. 86 undergraduate and 27 master's students were affected by its implementation. Our experiences are discussed in detail below.

5.1.1. Preparational phase

Starting the semester was more of a challenge than in previous years. It had to be assured that each Scrum Master was available at the time of his/her team's practice. If this was not possible, then he/she should have been present at least on the Scrum Review practices. It was allowed to hold the Weekly Scrum meetings regularly at a different time, as long as it was convenient for the team members.

At first, the presence of the Scrum Master was strange for the undergraduates. At the beginning of the semester, we did not communicate uniformly to the undergraduates how Scrum works and what the role of the Scrum Master is. We assumed that it would be enough to assign the Scrum Masters to them, and as a first step the Scrum Masters would then explain how Scrum works. The uncertainty with which the teams welcomed the Scrum Masters was an unpleasant experience for them. We received the feedback that it would be more beneficial to prepare the undergraduates more on how Scrum works.

In contrast, master's students were satisfied with their preparation. The Scrum Master preparatory training was brief and focused, and they liked this. They also appreciated that the Scrum Master test was written on the third practice. At this stage of the academic semester, they were not overloaded, so they could focus on this test. According to their feedback, they felt they had become, by the training and the test, adequately prepared to fill their role.

5.1.2. Main phase

At the weekly Scrum Master consultations, students raised very exciting topics. They addressed the depths of teamwork and Scrum operation. We highlight some of the common issues and some unique issues:

- Mostly, at the beginning of the semester, several Scrum Masters indicated that their teams are quiet, no conversation is developing during the meetings. We presented them some icebreaker games. We drew their attention to the fact that communication can be strengthened by setting an example, and by acknowledging and praising brave communication. However, it is also a fact that each team is different depending on the personalities of the members. Not all teams are expected to reach the same vibrant communication style.

- Procrastination was the most common problem in teams. Scrum Masters were given tips to avoid procrastination, e.g., the introduction of sub-deadlines, awareness of the many harmful effects of procrastination, etc. At the same

time, we found that procrastination may be strongly related to the unbalanced workload of students caused by various other courses.

- The case of the amused, defocused team has also been raised several times. The phenomenon is that Weekly Scrums are significantly longer than they need to be because time goes by with jokes. One of the Scrum Masters chose the strategy of introducing a "Just for fun 5 minutes", where team members could experience their desire to joke. Framing jokes and social connection time seemed to us to be a good solution. There was a Scrum Master who could fit in to spend more time on his team and chatted with them longer. From the point of view of university life, we considered it useful to build relationships and share experiences between the undergraduate and master's students.
- There was a case when the Scrum Master reported poor communication of the team. When we started discussing the case, it turned out the actual problem was that a team member was not responding to any messages sent between meetings. In the end, we came to realize that poor communication is actually just a consequence – the real problem is poor engagement.
- For another team, the Scrum Master reported commitment issues. One of the team members disappeared and did not respond to messages for more than a week. The Scrum Master was clueless about how to communicate in a non-humiliating but effective way with the missing person. We introduced him to the Non-Violent Communication method. The Scrum Master tried out the method in a written message and got back a clarifying, transparent message.
- One of the Scrum Masters recognized that he had a team member, who made a very pleasant impression at the Weekly Scrums, but unfortunately did not actually work much. We advised him to talk with the member in person, confront him with the facts about his progress, but also remain open to his explanation.
- A frustrated Scrum Master reported that one of the team members had not appeared for the second time in a row at the team's meeting and claimed he had forgotten. We suggested trying to develop an individual method together with the person with the goal to avoid this issue in the future, e.g., by setting up an automatic calendar alert.
- A Scrum Master mentioned incidentally that his team works in such a way that there is an experienced and determined member who always decides

when-what to implement and the others adapt to him in everything. We drew his attention to the fact that this may not be a healthy operation, and he should assess how the other team members experience this situation. Even if the situation is comfortable for them, it puts the team in a very vulnerable position. Nor is it certain that this way the team members develop their level of self-organization. Another similar case appeared in a two-person team where one of the team members led, the other just adapted. Here again, we made the Scrum Master aware that this is not a suggested teamwork and the adaptable team member needs to be activated.

- It has been reported that some teams were afraid to ask questions from the Product Owner. We initiated their Scrum Masters to invite the Product Owner to a weekly meeting, helping to build trust between the Product Owner and the team.
- One of the Scrum Masters reported that his team was progressing well, the product has evolved, but somehow the team was quite dependent on the Scrum Master. This team seemed to be a fan of the Scrum Master, his personality, and the Scrum Master was in the center of the team. Once we became aware of this phenomena, we suggested trying out delegation. For example, asking a member to run a Weekly Scrum and only comment on it when absolutely necessary, or even handing over the coordinating role on Retrospectives and helping the coordinator like an agile coach.
- One of the Scrum Masters noticed that the documentation prepared by his team often contained misspellings. He wondered if he could improve the spellings. He was advised to report this issue to the team and work together to find a way to avoid this, e.g., pair review, introducing a responsible role for this, etc. If it fits him/her, the Scrum Master can help his/her team with correcting the documentation, but only if it has been clarified and discussed in advance together with the team. The Scrum Master's primary job is to help the team to find a way of increasing quality.

These problems might not have come up if we hadn't asked questions at the beginning of the consultations. We highlighted certain team aspects in the content of the questions, such as strengths and weaknesses of the team, level of soft skills owned, quality of Scrum events, etc. However, the goal that Scrum Master students should discuss the given topics as a community of professionals, advising each other, coaching each other, was hardly achieved. In most cases, only the teachers commented on the experiences shared by the Scrum Master. However,

students were inspired by listening to each other's challenges, experiences, successes, and the comments from the teachers. For example, delegation was tried out towards the end of the semester by more Scrum Masters. Scrum Master students also inspired each-other in creating unique Retrospective techniques as well. They learned from each other that it is worth holding a test Review, or at least discussing the details of the previous Review. Someone came up with the idea to make a member of the team responsible for ensuring the smooth operation of the version control system, which strategy worked well and was instructive for others.

In addition to training all students, the consultations were also useful to us in terms of getting an idea of how the teams are evolving. We learned, for example, that communication in teams became better as the semester progressed. Psychological safety has developed in the teams, they dared to ask more and more questions, they also improved each other's code, they came up with proactive ideas, they discussed the plans before they started working. Some teams also got to practice positive criticism on each other's work. Towards the end of the semester, the teams had a better understanding of Scrum roles. The final stage of the Sprints remained quite stressful throughout the entire semester, and the teams reached the most progress at this time.

We found that there were different types of Scrum Masters. There were those who reacted slowly, and those who reacted fast to the difficulties of their team. In our opinion, the appropriate reaction time is not trivial, as if the leader reacts immediately, it may give less room to the team to develop self-organization. If the leader reacts late, the initial problems may swell by then. There were Scrum Master students who took great advantage of the opportunity offered by their role, e.g., introduced new Retrospective techniques, tried out delegation, continuously improved the independence of their team. There were Scrum Masters who reported that they do not have much to do because their teams are very independent. In such cases, we asked them to at least try to decipher what the secret of the well-working team is, so that others could get ideas on how to develop good teamwork.

There were three cases where the operation of the team or the Scrum Master encountered serious problems:

- On a Review, two members of the same team presented two different branches. The product's features did not work in unison, as if we had seen two different products. Both the Product Owner and the Scrum Master were shocked by these phenomena. For the third Sprint, we managed to merge the two product parts and to get the two team members to work together.

- One team member hardly worked on the project. This behavior could not be corrected by several attempts. Although the team members of the Scrum Master and the Product Owner discussed the problem together in a special meeting, they were unable to find a good strategy to change this behavior. The team member still did not contribute adequately to the project. The other team members were very frustrated and dissatisfied by this. Unfortunately, this problem has not been addressed according to its weight throughout the entire semester.
- Referring to her other activities, one of the Scrum Master students did not do her job properly, she missed the Weekly Scrums and Scrum Master consultations. Luckily, her team was self-organizing and progressed without her, but we were quite disappointed.

Another unsuccessful element of our approach were the weekly reports. These were filled regularly by a negligible part of the students, mainly the explanatory parts were missing. Some of the students reported that they did not fully understand the difference of some aspects highlighted in the report schema or they found it difficult to draw the line between them. We suspect that it could have been motivating for the Scrum Masters if they received more feedback on reports during the semester. Sadly, they received meaningful feedback only at the end of the semester.

We were surprised to find that no one used the Review app. We do not know the exact reason for this. Maybe the facts that it was passed only during the semester, it did not have a nice user interface and the substantive part (i.e., the advices) was also available in pdf format may have contributed to its underutilization. Unfortunately, we neither have data on whether the tips from the pdf were used by the Scrum Masters.

We also observed that Product Owners, who were otherwise teachers, found it difficult to position themselves. They did not communicate clearly enough the acceptance criteria and were less cooperative than task allocators. We suspect this happened because they are accustomed to the role of the teacher. The transformation from another similar function may be more difficult than the case when this type of role is completely new to someone. (We observed a similar situation with a Scrum Master. He had previously worked as a lecturer at the university. He, too, was more revelatory than collaborative.) Training the instructors, which strengthens the correct behavior of the Product Owner and allows for discussion, may be a good idea. Case studies from industry could illustrate importance of a few particular behaviors. As part of the training, instructors could be asked

to create a basic Product Backlog. This could be a good exercise for learning attitudes in representing costumer's business needs, instead of valuing technical solutions only.

The Scrum Planning and Retrospectives were missing from the course schedule. There were historical reasons for this. When we first started the agile transformation of the Software Technology course, master's students were not involved as Scrum Masters. Instead, both the Product Owner and the Scrum Master role for a team were filled by the same instructor. Each instructor was responsible for 5 teams, and so the instructor could spent about 15 minutes with each team on a practice. Therefore, retrospective and planning meetings were not feasible. Instead, on occasions other than Sprint Reviews, instructors held a Weekly Scrum with each of their teams to support development processes. When master's students took over the role of the Scrum Master, we overlooked the fact that there was sufficient time on practices for Scrum Retrospectives and Scrum Plannings. These were not included in the practice schedule, we only suggested that retrospectives and plannings should be held the next week after the Scrum Reviews. As a result, we were not sure that retrospectives and plannings had actually been held every suggested week and in an effective way. In the future, we definitely make Scrum Retrospectives and Scrum Plannings a part of the course schedule, as these are two essential Scrum events.

5.1.3. Closing phase

At the final review and consultation of the semester, the undergraduate and graduate students, as well as the lecturers expressed their high satisfaction with the two combined courses. The realized software products were good, and the undergraduates were proud of their performance. The Scrum Master students reported that they enjoyed being able to try themselves in a leadership role. They wondered if there would be more courses addressing leadership roles because they would be happy to join.

5.1.4. Evaluation of the students

In Hungary, assessments of the students are realized on a scale of 1 to 5, where 5 corresponds to excellent knowledge level. The average assessment value on our undergraduate course was 4.81. The average assessment value on our master's course was 4.78. 98% of undergraduates received a value of 4 or 5. This was true for 96% of our master's course participants.

We do not have detailed data on the partial results of the Scrum Team members.

In the case of Scrum Master students, the number of points obtained was the most varied in the case of the Scrum Master test scores. The minimum was 11 points, and the maximum was 23 points out of the 25 points. Values were evenly distributed on this scale. The Scrum Master's ratings from undergraduate students were very high, with a minimum of 20 points out of a possible 25. In the case of the mid-term teacher evaluation, the minimum was 28 points, but except this one, each score was at least 41 points out of the 50 points achievable.

6. Measurements

At the end of the semester, we performed measurements on the result of our experiment using questionnaires. Students were provided with a Google Forms link and they answered the questionnaire anonymously. Intermediate data capture was dismissed to not burden students with many questionnaires during the semester. Furthermore, the majority of our questions were meaningful over a longer period of time: for example, measuring the perceptible extent of developing skills and attitudes. There are known limitations of our study. First, our results are based on self-assessment data. Second, the questionnaire may have shed less light on possible phenomena for which there were no specific questions. Third, the number of participants of the study is less than 150, so the result should be interpreted accordingly.

Below we present our most important measurements. 86 undergraduate students and 27 Scrum Master students who participated in the experiment completed the questionnaires anonymously. As an answer to almost every question, students had to select a value on a scale of 1 to 5, with 1 corresponding to "at least" and 5 corresponding to "at most".

As a control, we also completed the questionnaires with students who have not participated in the experiment. These are undergraduate students who studied within the same Software Technology (SWT) course (same goal, schedule, evaluation), except that their teams were not led by a Scrum Master student. In their case, the role of the Scrum Master was performed by the same instructor who was also their Product Owner. This group of students is hereinafter referred to as the control group. 83 people from the control group completed the questionnaire.

First, we asked students how useful did they find the Software Technology course (Table 3). There was very little difference between the evaluation of the experimental and control groups, only 0,03. However, a 4.63 average value is considered high within the 5-point scale. We think students like this course because they can work in teams, develop a larger game software and learn industry-related elements of software development, such as agile methodologies, version control systems, continuous integration and continuous delivery.

	SWT experimental group average	**SWT control group average**	**Difference**
How useful did you find the Software Technology course?	4.63	4.60	0.03

Table 3. Usefulness of the Software Technology course – A/B testing

We were also interested in how well the students understood and experienced the different elements of the Scrum methodology (Table 4). There were no major differences between the experimental and control groups in terms of measurement values, with the largest difference being 0.40. The 0.40 average difference was regarding the understanding of the Scrum roles, which was more successful in the case of the teams with the Scrum Master students. We find it logical that Scrum roles have been better understood by teams that had a Scrum Master student than by those where both the Scrum Master and the Product Owner were played by the instructor. In this case, it may have been more difficult to shed light on the different responsibilities and significance of the different roles.

The second-highest difference was in understanding Scrum events (0.33) and the third in experiencing Scrum values (0.30). Based on these, the constant presence of Scrum Master students in the team was able to clarify the course of Scrum events and the importance of Scrum values. Although the difference is even smaller in the other averages, it is worth noting that teams with a Scrum Master student in them always showed higher values. In addition, the lowest average value was 4.33, which in itself is a large value on a scale of 5. This value refers to the understanding of Scrum artifacts, which is understandable, because this element was not emphasized in our experiment. The teams did not have a Product Backlog or Sprint Backlog, they mostly used only a Kanban board as a task manager, but this was also optional.

	SWT experimental group average	**SWT control group average**	**Difference**
How much have you understood the theory of agile methodologies?	4.34	4.30	0.04
How much have you experienced the practical operation of agile methodologies?	4.36	4.11	0.25
How much have you realized your own agile role: being an active team member supporting self-organization?	4.63	4.42	0.21
How much have you understood Scrum roles (who is responsible for what in Scrum)?	4.58	4.18	0.40
How much have you understood the Scrum events (goals, correct realization)?	4.66	4.34	0.33
How much have you understood the Scrum artifacts (which one is what, how to handle it)?	4.33	4.11	0.22
How aware have you become of the importance of Scrum values (courage, openness, respect, focus, commitment) in teamwork?	4.67	4.37	0.30

Table 4. Learning Scrum on Software Technology course – A/B testing

We found it interesting to examine how the application of the Scrum methodology affected the development of students' soft skills (Table 5). We believe that the in-depth teaching of Scrum also includes the development of these skills. The average of the measured values does not differ much in the case of the experiment group and the control group. The biggest difference is 0.49 for the development of task division skills. Students who had a former Scrum Master developed better in terms of task-sharing skills. This may be due to the fact that the Scrum Master students were able to pay more attention to the operational efficiency of the team assigned to them, including the division of tasks. The smallest difference in average (0.08) and the smallest average value (3.70) came in the case of time management skills. This is consistent with the fact that we have heard many times procrastinations during consultations with the Scrum Masters. Another recurring theme was the lack of focus within the team, which may have similarly resulted in time management problems. We conclude that students felt that there

	SWT experimental group average	**SWT control group average**	**Difference**
How did your communication skills develop during your teamwork?	4.21	4.02	0.19
How did your cooperation skills develop during teamwork?	4.44	4.30	0.14
How did your organizational skills develop during teamwork?	4.34	4.06	0.28
How did your time management skills develop during your teamwork?	3.70	3.61	0.08
How did your task division skills develop during teamwork?	4.36	3.87	0.49

Table 5. Developing soft skills on Software Technology course – A/B testing

was still room for improvement in terms of time management. In our study, collaboration showed the greatest improvement, which is a promising result as this skill is very important in terms of Scrum. The improvement of both communication skills (4.21) and organizational skills (4.34), which is an important area of the application of Scrum, was generally observed by students.

We can also compare the development of undergraduate and graduate students who participated in the experiment. We can analyze the level of understanding and the level of soft skill development in the Software Technology (SWT) and Software Development in Practice (SDP) courses.

The averages were generated based on 27 Scrum Masters' completion of the questionnaire. These are compared with the results of the undergraduate students participating in the experiments (86 students). Regarding the usefulness of the courses, the results are surprisingly similar and high (4.63, 4.60). Our conclusion is that on both study levels (BSc, MSc) we managed to create useful courses (Table 6).

Based on the measurements presented in Table 7, the theoretical parts of Scrum were better understood by the Scrum Masters, and the practical parts by the members of the development teams. The theoretical aspects of agile methodologies, Scrum events, working materials were better understood by the Scrum Masters, since we talked about them a lot during the preparatory training and consultations. Interestingly, Scrum roles were understood at a similar level by undergraduate and graduate students. We, on the other hand, consider it probable that the Scrum Master students first understood it, and they succeeded in passing

	SWT – Software Technology course	**SDP – Software Development in Practice course**	**Difference**
How useful did you find the Software Technology course / Software Development in Practice course?	4.63	4.60	0.03

Table 6. Usefulness of the two redesigned courses

	SWT experimental group	**SDP experimental group**	**Difference**
How much have you understood the theory of agile methodologies?	4.34	4.64	- 0.30
How much have you experienced the practical operation of agile methodologies?	4.36	4.08	0.28
How much have you realized your own agile role: being an active team member supporting self-organization?	4.63	4.28	0.35
How much have you understood Scrum roles (who is responsible for what in Scrum)?	4.58	4.60	-0.02
How much have you understood the Scrum events (goals, correct realization)?	4.66	4.84	-0.18
How much have you understood the Scrum artifacts (which one is what, how to handle it)?	4.33	4.52	-0.19
How aware have you become of the importance of Scrum values (courage, openness, respect, focus, commitment) in teamwork?	4.67	4.56	0.11

Table 7. Learning Scrum on the two redesigned courses

this knowledge on to the undergraduate students with their presence. In terms of practical aspects, the Scrum Masters were more restrained in the appraisal of experience. The biggest difference in rating was at the level of implementing Scrum roles. In this case, the Scrum Masters rated their implementation of their Scrum role at an average of 4.28, and the team members at an average of 4.63. We think

it is clear that the role of the Scrum Master is more difficult to implement, its responsibilities and tasks are much more diverse, requiring advanced soft skills levels. This may have resulted in the Scrum Masters feeling less successful in accomplishing their Scrum role. The practical operation of agile methodologies and the importance of Scrum values were also more felt by undergraduate students. This can even be explained by the fact that they worked together more. The fact that Scrum was mostly a novelty for undergraduates could also contribute to the evaluation of experiences. Master's degree students are likely to be more rigorous and nuanced about their evaluations due to their experience.

Differences in rating soft skills development may also be related to students' previous experiences. The soft skills of master's students are probably more advanced from the start and can be developed less by one course. Nevertheless, the self-reported level of development shown in Table 8 was at least 3.52, which can be considered a significant improvement. Students' time-managing skills developed the least on both courses. On both courses, students' time-managing skills developed the least. Undergraduates reported much more progress in task division than master's students (0.80). This, in turn, may mean that master's students have transferred task-managing skills to undergraduates.

	SWT experimental group	**SDP experimental group**	**Difference**
How did your communication skills develop during your teamwork?	4.21	4.04	0.17
How did your cooperation skills develop during teamwork?	4.44	4.24	0.20
How did your organizational skills develop during teamwork?	4.34	4.04	0.30
How did your time management skills develop during your teamwork?	3.70	3.52	0.18
How did your task division skills develop during teamwork?	4.36	3.56	0.80

Table 8. Developing soft skills on the two redesigned courses

We also asked undergraduate students how much they were helped by the presence of the Scrum Master. In parallel, we also assessed how much the Scrum Master students felt they could help the undergraduates. Averages are shown in Table 9. Overall, the results showed that undergraduates most often rated the help of Scrum Masters much higher than Scrum Masters themselves. The

biggest discrepancy was in the assessment of the preparation to the Reviews, where the average of the contribution was 1.07 higher from the perspective of undergraduate students. By the way, undergraduates were the most satisfied with the quality of the Retrospectives, the handling of obstacles, and the Weekly Scrums. Based on the measurements, the Scrum Masters helped the Planning in the slightest. Based on measurements, Planning showed the least improvement by Scrum Masters' help. This may be due to the fact that Planning was not mentioned in the course schedule of the Software Technology course. In addition, Scrum artifacts did not receive extensive focus in our experiment. For example, Product Owners were not obligated to create a Product Backlog that could have been used for Planning. Estimation methods were not taught to students either. This was due to a lack of resources (a Product Owner had at least 5 teams). Overall, the lowest average for undergraduates was 4.67, which is still very high, so it turned out that it was a positive experience for undergraduates that they had a Scrum Master student to work with. Scrum Masters scored their help to the other students at 3.64. This average might simply suggest a more realistic estimate based on prior experience.

	SWT experimental group	**SDP experimental group**	**Difference**
Overall, how much did the Scrum Master help the team's work?	4.74	3.92	0.82
How much did the Scrum Master help with the Planning?	4.67	3.64	1.03
How much did the Scrum Master help with the Review?	4.79	3.72	1.07
How satisfied were you with the Retrospectives led by the Scrum Master?	4.91	4.20	0.71
How satisfied were you with the way the Scrum Master attended the Weekly Scrums?	4.86	4.64	0.22
How satisfied were you with the way the Scrum Master handled if there was any obstacle in the team?	4.87	4.48	0.39
How much did you feel that the Scrum Master constantly improved the team?	4.83	4.08	0.75

Table 9. Effect of having / being a Scrum Master

We asked the students about the benefits of combining the two courses. We cite some student responses.

What was the best thing about having a Scrum Master?

- "It gave us a good start."
- "The meetings were conducted professionally. We had someone to turn to if we got stuck in something, whatever problem we had."
- "Problems within the team have been solved very effectively, improving the atmosphere and productivity."
- "Helped stay motivated and meet deadlines."
- "He has always supported and encouraged us."
- "I could almost feel like we were on a workplace meeting."

What was the best thing about being a Scrum Master?

- "That I could lead people without command."
- "To see how the daily Scrum has improved from a must / awkward meeting to a useful and informative meeting."
- "To witness the development of the team. They managed to advance them in self-organization, planning, etc."
- "To see how the team builds itself."
- "It was great to experience working with such a motivated and enthusiastic team. Self-organization really worked. It was good to see them developing during the semester."
- "I could see the Scrum from a slightly different perspective."
- "Gaining Scrum Master experiences, overcoming challenges, leaving my comfort zone."
- "I learned that the Scrum Master does play an important role."

7. Lessons learned

From our case study, we concluded that teaching different aspects and levels of Scrum in a resource-efficient way is possible by successfully combining an undergraduate course with a master's course.

In our case, members of both courses found the linked courses useful. Averages of usefulness were rated 4.63 and 4.60 on a scale of 1 to 5. Students have understood the theory of Scrum (average level of understanding 4.34, 4.64), but

more importantly they could experience Scrum in practice (average level of understanding 4.36, 4.08), and realized a Scrum role (average level of realization 4.63, 4.28). Students also reported improvements in soft skills, such as an average 4.44 improvement level in collaboration skills, 4.21 in communication skills.

Both course's participants benefited from being associated with the other course. Undergraduate students were helped by the presence of the Scrum Masters, who gave them a good start, always supported and encouraged them. The master's students could try out the role of the Scrum Master through leading the undergraduate student teams and experienced team development, leading without command.

We have learned lessons from our implementation that have confirmed some elements of our approach and suggested changes for others. We discuss this below.

One of the challenges in linking the two courses is that the preparatory phase is more difficult than usual, as the schedules of master's and undergraduate students need to be aligned. We believe that undergraduates should be briefly and uniformly prepared for the Scrum methodology at the start (e.g., during a first lecture) in order to have a smoother and more pleasant start of working together with the master's students. In fact, this preparation could support all roles and stakeholders. Short, concise, application-focused training to prepare students for the Scrum Master role is a popular solution among master's students. According to them, it is also advantageous to write a test regarding the role already at the beginning of the semester, as they take the time to thoroughly master the theory, on which they can build later in practice. This early semester period is favorable for test writing, as at this stage of the semester students are typically not burdened with other examinations.

We consider a very good method to support the Scrum Masters in the form of a group consultation during the semester. These consultations bring up problems that are common, and thus it is important for several people to deal with them. For example: initiating communication in the team, avoiding procrastination, managing a defocused team, developing communication in the team, etc. However, individual cases also come up. These are instructive, because by discussing how they are treated, we can understand the Scrum principles and values. For example, in the case when the Scrum Master reported that his team was working successfully but still depending on the Scrum Master, or when another Scrum Master noticed spelling mistakes in the documentation and didn't know if he should correct it, we discussed the importance of the team's self-organization and independence, and the way of supporting it. The successes that the Scrum

Masters report to each other during the consultations provide encouragement and ideas. For example, our students inspired each other to invent new Retrospective techniques or delegate tasks, developing their team's self-organization.

Consultations are also beneficial from an instructor's perspective, because they allow tracking what's going on with the teams, what problems they have. Because of some difficult cases, we have learned that it would be wise to prepare in advance dealing with some critical cases, so that if they arise during a consultation, we could respond quickly. For example: a team member not working, or how to give work to a Scrum Master whose team has almost no challenge.

Our way of logging that run in parallel with consultations worked less efficiently. Maybe another template that is easier and more useful to fill out would be more efficient. Efforts could be made to improve the relationship between consecutive weekly diaries, for example, by loading the values into a column of a table and automatically generating a development curve from the values. This curve could even be reexamined at every consultation, motivating completion of the values. At the end of the semester, we could ask for an analysis of all the values collected during the semester. This could be a useful task for students as it would cause them to look at the whole team dynamic and process. This analysis could also play a role in grading the students, which would further support taking mid-term data collection seriously.

Scrum Planning was not taken seriously either. Logs completely missed to report them. Both the log template and the course schedule should highlight Scrum Planning and Review, with a precise breakdown into the appropriate weeks to ensure that they are not missed. Thus, teaching of the Scrum events would be more effective and the progress of the teams could also receive more support.

Similarly, we did not hear much about cooperation with the Product Owners. This role should be better supported. Their Scrum responsibility and attitude should be better clarified. This is not necessarily an easy task, as mostly instructors fill this role. They are accustomed to their authority in the courses taught. They have a specific and habitual style of work that is difficult to overwrite for the sake of our courses. This is actually an agile transformation issue and transformations are known to be challenging. Regarding the evaluation method of the students, we found that it resulted in too high grades. We need to find a method that would produce more nuanced grades. This could be supported by a more objective assessment of the mid-term activity in the case of the Scrum Masters.

It is common at universities that Scrum is presented to students from the perspective of the development team on a practice course. In these cases, mostly one

student can try out the role of the Scrum Master. We assured that our undergraduate students experience Scrum in practice with a similar solution, meanwhile all master's students can try out the role of the Scrum Master. We used role-based Scrum teaching, previously typical only in industrial Scrum education, now in university education. Our master's students could take a Scrum Master course free of charge, which otherwise costs approximately €1,500. Our Scrum Master course was highly practice-oriented. We believe we met all the three principles (Kropp et al., 2014) of high-level agile education: we assured personal experience by working in a Scrum Team; cooperating, social exchange and discourses on Scrum events; and discussing agile values and attitudes on weekly consultations.

Scrum Master students reported that it was interesting to experience themselves as a leader, and would be happy to take several such courses. Undergraduate students reported that Scrum Master students helped them a lot, and our measurements show that they have understood (especially) Scrum roles better than the teams without assigned Scrum Master students.

The fact that we assigned master's students to the course of the undergraduate students was also a resource optimizer for us. In this way, the instructors only had to pay attention to the role of the Product Owner, and we did not have to hire research assistants as in the case of Scharf and Koch (2013) to support teams. 5 teachers were involved in our implementation, 4 as Product Owners, 1 as Agile Coach. This means 5 teachers supported approximately 110 students, while in the case of Scharf and Koch (2013) 11 staff members supported approximately 60 students.

As a side effect, our experiment had a relationship-building effect, as all undergraduates participating in the experiment were able to get to know at least one master's student. This relationship, together with the inspiration it provides, is likely to attract more undergraduate students to enroll in master programs and strive for a higher professional qualification. This is profitable for the university as well.

At the beginning of our research project, we raised the following research questions: How can we provide undergraduate students with experience in Scrum methodology? How can we provide master's students with experience in agile leadership? How can we realize both of our previous goals in a resource-efficient way? As an approach, we redesigned and linked an undergraduate and a master's course. Implementing these courses proved to us that our approach is an answer to the previously raised questions. Our case study highlighted some weak points

of our approach as well. We examined these weak points and identified opportunities for improvement. While we consider the approach presented in this article satisfying, we believe that implementing our improvement ideas could produce even better results.

References

15th annual State of Agile Report. (2021). `https://digital.ai/resource-center/analyst-reports/state-of-agile-report`.

Agile Training. (2021). `https://www.agilealliance.org/training/`.

Damian, D., Lassenius, C., Paasivaara, M., Borici, A., & Schröter, A. (2012). Teaching a globally distributed project course using Scrum practices. In *2012 Second International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD)* (pp. 30–34). IEEE. doi: 10.1109/CTGDSD.2012.6226947

Dilts, R. B. (1996). *Visionary leadership skills: Creating a world to which people want to belong.* Meta Publications.

Hof, S., Kropp, M., & Landolt, M. (2017). Use of gamification to teach agile values and collaboration: A multi-week scrum simulation project in an undergraduate software engineering course. In *ITiCSE'17: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 323–328).

Kropp, M., & Meier, A. (2013). Teaching agile software development at university level: Values, management, and craftsmanship. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)* (pp. 179–188). IEEE. doi: 10.1109/CSEET.2013.6595249

Kropp, M., Meier, A., Mateescu, M., & Zahn, C. (2014). Teaching and learning agile collaboration. In *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)* (pp. 139–148). IEEE.

Mahnič, V. (2011). A capstone course on agile software development using Scrum. *IEEE Transactions on Education*, *55*(1), 99–106.

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

Paasivaara, M., Heikkilä, V., Lassenius, C., & Toivola, T. (2014). Teaching students Scrum using LEGO blocks. In *ICSE'14: Companion Proceedings of the 36th International Conference on Software Engineering.*

Paasivaara, M., Lassenius, C., Damian, D., Räty, P., & Schröter, A. (2013). Teaching students global software engineering skills using distributed Scrum. In *2013 35th International Conference on Software Engineering (ICSE)* (pp. 1128–1137). IEEE. doi: 10.1109/ICSE.2013.6606664

Professional Scrum Training. (2021). `https://www.scrum.org/courses`.

Ramingwong, S., & Ramingwong, L. (2015). Plasticine Scrum: An alternative solution for simulating Scrum software development. In *Information science and applications* (pp. 851–858). Springer.

Reichlmayr, T. (2011). Working towards the student Scrum: Developing Agile Android applications. In *2011 ASEE Annual Conference and Exposition* (pp. 22.1712.1–22.1712.12). ASEE.

Rodríguez, G., Soria, A., & Campo, M. (2011). Teaching Scrum to software engineering students with virtual reality support. In F. Cipolla-Ficarra, K. Veltman, D. Verber, M. Cipolla-Ficarra, & F. Kammüller (Eds.), *ADNTIIC 2011: Advances in New Technologies, Interactive Interfaces, and Communicability* (pp. 140–150).

Scharf, A., & Koch, A. (2013). Scrum in a software engineering course: An in-depth praxis report. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)* (pp. 159–168). IEEE. doi: 10.1109/CSEET.2013.6595247

Schwaber, K., & Sutherland, J. (2011). The Scrum guide. *Scrum Alliance*, *21*(19), 1–12.

Von Wangenheim, C. G., Savi, R., & Borgatto, A. F. (2013). SCRUMIA – An educational game for teaching SCRUM in computing courses. *Journal of Systems and Software*, *86*(10), 2675–2687.

Zorzo, S. D., de Ponte, L., & Lucrédio, D. (2013). Using Scrum to teach software engineering: A case study. In *2013 IEEE Frontiers in Education Conference (FIE)* (pp. 455–461). IEEE. doi: 10.1109/FIE.2013.6684866

ENIKŐ ILYÉS
FACULTY OF INFORMATICS, EÖTVÖS LORÁND UNIVERSITY,
H-1117 BUDAPEST, PAZMÁNY PÉTER SÉTÁNY 1/C, HUNGARY

E-mail: `ilyese@inf.elte.hu`

(Received December, 2021)