Curricula and Methods on Teaching Different Aspects of Agile Software Development

ILYÉS Enikő

Abstract. Agile methodologies are the most commonly used software development methodologies nowadays. For this reason, education of software engineering should include this topic at universities. However, the educational method of this topic has still open questions. Higher education lacks traditional curricula and methods in teaching agile software development.

Agile methodologies are based on collaboration and interaction. For this reason, high level teaching of agile methodologies requires not only teaching of theory and practice, but also development of soft skills and of an appropriate set of values. We need to find teaching methods that can develop students in all four aspects: theory, practice, skills, values. This article presents methods for developing all four aspects of agile methodologies. In addition, by describing five different course curricula, it proves these methods can be effectively incorporated into classic university courses.

Keywords: education, agile, Scrum, skills, values

1. Introduction

Agile methodologies are very popular nowadays in the software development industry. As a result, university programs in the field of software development have begun to consider teaching agile methodologies necessary.

Application and teaching of agile methodologies can be analyzed from several aspects. One aspect is the theory of these methodologies, such as what events and roles are defined within them. The practical aspect is to experience how the methodology works. Soft skills can be considered as a separate, third aspect: agile methodologies are very communication and collaboration based, so high-level applications of agile methodologies require high-level use of soft skills. The fourth aspect is the agile values, which must be continuously manifested during application of the methodology. Only this way operation of agile methodologies will be credible and effective. In case of Scrum, these values are courage, respect, openness, commitment, focus.

A common solution to teaching agile methodologies is to implement a full semester software development project in teams, most often adapting Scrum. Another approach of instructors is to teach Scrum through a simulation project.

While articles on the topic mostly discuss how the teaching or application of agile methodologies have been realized in a single course, this paper discusses education of agile methodologies in different courses of an entire software engineering educational program, which is the main contribution. At different levels of the program, in mandatory, optional, or elective courses, students can get to know agile software development. From undergraduate to graduate, 5 different courses offer an understanding of agility from 5 different perspectives. Lectures, seminars, practices, and research groups included in this package can be understood as modules, and their combination allows students to acquire in-depth knowledge about agility.

This article presents the details, methods used, aspects developed in the 5 different modules of this package. It seeks to answer the following research questions:

1. Can different aspects of agile software development (theory, practice, skills, values) be taught at university?
2. What kinds of methods can be used to teach the aspects of agile software development?
3. In what kind of curriculum can these methods be integrated?

We set as acceptance criteria that we should define at least 3 different methods for the development of each aspect. These should be incorporated into at least 3 curricula. In the case of the implemented curricula, it must be demonstrated that the four different aspects have been developed by the courses in an amount of minimum 70%. The experiment should involve at least 130 students.

Therefore, following a brief clarification of key concepts (Chapter 2), the reader can find out what teaching methods we defined to be used to teach all four aspects of agility (theory, practice, skills, values) at university (Chapter 3). In addition, he/she will see examples of 5 different course topics and implementations that successfully integrated these teaching methods (Chapter 4). The article concludes with discussion and related work (Chapter 5). All this serves as inspiration for other universities or educational institutes for teaching agile software development in a comprehensive way.

2. About agile software development and Scrum

This chapter clarifies the fundamental concepts referred to in this article.

Agile methodologies are software development methodologies that implement the principles expressed in the Manifesto for Agile Software Development: emphasizing individuals and interactions, customer collaboration, responding to change, working software over comprehensive documentation. [1]

Scrum [2] is the most commonly used agile methodology [3]. It defines 3 Scrum roles, 5 Scrum events and 3 Scrum artifacts. The Product Owner represents the interests of the customer. The Scrum Master is responsible for ensuring that workflows are implemented efficiently and along Scrum values. The development team is a self-organizing team who has all the skills necessary to implement the product. The workflow consists of sprints. The sprints are fixed work periods of 1-4 weeks during which a new increment must be created. Each sprint begins with planning, during which the Scrum team commits to a sprint goal. Then they work together and discuss their progress over a short 15-minute meeting each day - this is called the daily Scrum. At the end of the sprints sprint review takes place, ideally in the presence of the customer, where they give feedback on the result of the sprint. This meeting is also suitable for brief discussion of additional needs of the product. The sprints are closed by the sprint retrospective, where the Scrum team analyzes its operation and optimizes it. During Scrum events the different actors of Scrum need to use their soft skills a lot, for example: communication, collaboration, adaptation, critical thinking. Their manifestations should be based on the Scrum values, which are as follows: respect, courage, openness, commitment, focus.

Prioritized product requirements are stored in the product backlog, managed by the Product Owner. Some of the elements of the list are selected by the team during the sprint planning in cooperation with the Product Owner, and a sprint goal is determined. The selected items are placed in the form of tasks in the sprint backlog, which the team manages during the sprint. An increment is created when a product backlog element is implemented and matches the definition of done.

The definition of done is fixed by the team. It is evolving during the project with the intention of creating the most stable and reliable increments possible.

3. Methods for teaching agile software development

This chapter defines methods for teaching the theory, practice, skills, and attitude aspects of agility in four separate subsections. For each method, it gives a general description and then helps a deeper understanding through an example.

3.1. Methods for teaching the theory

Lecture: The instructor introduces a topic to a larger group of students, orally, sometimes with the help of tools (ex. keywords, important definitions, figures listed on slides/whiteboard). Communication in this case is mostly one-way: the instructor speaks, the students listen.

Example: The target is to teach the Scrum roles. First the instructor highlights the main idea behind each Scrum role. Then he/she explains in detail each role's responsibilities and the qualities beneficial to have for that specific role. The oral presentation is supported by the following slide set: a slide with each role's main target; two more slides about each role, summarizing which responsibilities and qualities should the people in these roles have.

Reading literature: The instructor selects a literature, and the student has to read it carefully.

Example: The target is to make students understand Scrum in detail. The instructor selects The Scrum Guide [2] as obligatory literature - this should be read by the students. During the examination, the students have to know details that were not mentioned during classes but are presented in this document.

Literature review: The students - independently or in teams - have to categorize, compare, summarize the basic ideas of the literature related to a given topic.

Example: The aim is to give insight into agile transformation. Students should create a list of causes and workflows for agile transformation based on a number of literatures.

3.2. Methods for teaching the practice

Semester-long software development project - from the developer's perspective: A group of students work on a software development project during an entire semester. Each student is solely a member of the development team (not a leader).

Example: The aim is to provide a possibility for students to experience the workflow of Scrum. Students should develop in teams of 3 a game software (for example: Adventure park) adapting the Scrum methodology. The role of the Product Owner is fulfilled by the instructor, and the Scrum Master is a senior student. The semester consists of 4 sprints, 3 weeks each.

Semester-long software development project - from the Scrum Master's perspective: A student supports a development team during an entire semester long, taking on the role of the Scrum Master.

Example: The target is to provide a possibility for students to experience the servant leadership of Scrum. A senior student as a Scrum Master supports a team of 3 undergraduate students who need to develop a game software in one semester. The Scrum Master student holds the weekly Scrum and at the end of the 3-week sprints he/she prepares the team for the review, coordinates the retrospective and the planning.

Semester-long software development project - from different perspectives: A group of students work on a software development project for a full semester. Each student takes turns in the group as a developer and Scrum Master.

Example: The target is to provide students the possibility to experience both the developer and the Scrum Master roles. Students develop a small administration tool in teams of 3. The project is implemented in 3 sprints using an adaptation of Scrum methodology. During each sprint different students from the teams take on the roles of Scrum Master, Product Owner, and developer.

Monitoring of several software development teams - from the perspective of an agile coach: A student monitors the operation of several development teams and helps them with reflections, advice.

Example: The target is to give multiple insights about how Scrum teams work. Students in teams of 3 use Scrum to develop a software product. At the end of each 3 week-long sprint a senior student visits all teams one by one. He/she coordinates a Retrospective, where they reflect on their strengths and weaknesses. He/she supports them with reflections and advice addressing their impediments.

Situational exercises: Students simulate a situation as part of the lesson, inspired by the industrial use of the methodology, as part of a lesson. The goal is to understand and then elicit the correct operation of the methodology in a given situation.

Example: The target is to give the possibility to experience the course of a daily Scrum. We assign roles to 5 students from the class. Among the characters is a Scrum Master and 4 developers. The characters have different motivations and characteristics: The Scrum Master wants to validate Scrum values; developer 1 explains a lot and in detail; developer 2 is defocused because of a personal issue; developer 3 is ashamed to admit that he is stuck with his task; developer 4 likes to chat. The task is for students to simulate the daily Scrum of this team.

3.3. Methods for developing agile skills

Reflection: When a student looks back at a practical situation of which he or she was an observer or participant and analyzes it - alone or with the help of others. (Analysis can also be supported by sharing some criteria first.) He/she examines what was appropriate, what needs to be improved about the situation, and develops specific improvement ideas.

Example: The target is to learn how to communicate during a daily Scrum. After playing a situational exercise (see previous point), students look back at how they manifested themselves. If this is not considered appropriate, they get back into the situation and this time try to behave in an other, more appropriate way.

Schemas and examples: Presentation of manifestations that illustrate the proper functioning level of soft skills. If possible, sharing schemes that help to reach a similar level.

Example: The students work in a team of 3. One of the students does not appear at the meetings for more than a week, nor does he respond to messages. The team is insecure and frustrated. They

don't know how to solve this difficult situation correctly. The target is to learn how to communicate in these kind of difficult team situations. The instructor introduces the schematic operation of the Nonviolent Communication [4] (observations, feelings, needs, requests) in the class and then illustrates it with an example. It encourages students to use the scheme to write a message to the missing student.

Consultation: When the student, together with the instructor, analyzes a situation in order to learn how to behave in it. The student receives personalized recommendations from the instructor for self-improvement.

Example: A Scrum Master student reports that members of his team communicate timidly. The target is to teach the Scrum Master student how to support his team in gaining more courage. Advice listed: play icebreaker games, lay down common rules, give positive feedback, set the example of brave communication, reflect on what you see to the team (e.g. "I see you are timid"), etc.

3.4. Methods for teaching the values

Discussion: Students have to argue about what would be the appropriate behavior in a given hypothetical situation.

Example: The target is to help students understand the value of self-management. The instructor describes the following hypothetical situation to the students: A part of a team indicates to the Scrum Master that the division of tasks between them is not satisfying and one team member barely works. The Scrum Master takes control over the situation, distributes tasks proportionately among team members. The question is: Did the Scrum Master do the right thing? The instructor collects the pro and contra arguments and highlights the values behind them.

Reflection: When a student looks back at a practical situation of which he or she was an observer or participant and analyzes it - alone or with the help of others. (Analysis can also be supported by first sharing some criteria.) He/she examines what values have been manifested in that situation, what needs to be changed to manifest the agile values even more.

Example: Target is to teach Scrum values and detect their presence on Scrum event. Students work in teams of 3 on a semester-long project using Scrum. Following the first sprint, the instructor asks students to analyze their Scrum Retrospective in terms of how much the Scrum values (respect, focus, commitment, openness, courage) were present.

Consultation: When the student, together with the instructor, analyzes a situation in order to see how agile values were present during it. The student receives personalized recommendations from the instructor for improving the manifestation of agile values.

Example: Target is to teach that the presence or absence of Scrum values deeply effects teamwork. Students work in teams of 3 on a semester-long project using Scrum. The Scrum Master reports that one team member rarely communicates with other team members. He reports that this is despite the fact that he has created new communication channels. The instructor draws the attention of the Scrum Master to the fact that it is possible that the lack of communication is just a symptom. Maybe he has to deal with commitment, openness, courage issues. He should explore the deeper causes of missing communication through questions, listening, reflection.

4. Curricula for teaching agile software development

This chapter shows how the teaching methods discussed above can be incorporated and combined in 5 different courses. In each case, the course details, the weekly schedule, the way of implementing the teaching methods in the course and experiences related to the implementation, and students' feedback on the course are presented.

4.1. Software technology (practice)

Course details

Dedicated to: undergraduate students, fourth semester

Duration: 90 minutes / week

Type: mandatory

Goals: The aim of the course is to provide students with an experience of the entire project lifecycle and collaboration on developing a larger software project.

Developed aspects of agile: practice, skills

Weekly schedule:

Week	Action during class
1	Preparation
2	Weekly Scrum
3	Weekly Scrum
4	Review
5	Weekly Scrum
6	Weekly Scrum
7	Review
8	Weekly Scrum
9	Weekly Scrum
10	Review
11	Weekly Scrum
12	Weekly Scrum
13	Final Review

Teaching methods used:

Semester-long software development project - from the developer's perspective: Students must develop a strategic game software in teams of 3 in any programming language. At the beginning of the semester, they receive a short (15-minute) presentation on how the Scrum methodology works. Then they have to use an adaptation of Scrum. The semester is divided into four 3-week sprints. The role of the Product Owner is played by the instructor, the Scrum Master is a senior student, or also the instructor (in case we do not have sufficient senior student taking this role). Every 3 weeks the teams present the results of their sprint to the class. In the intermediate

weeks the teams do a weekly Scrum. During the sprints, students should work together in a self-organizing way. They need to distribute the tasks, share their impediments and work together to find solutions. After the Sprint Reviews, the teams have to hold a Retrospective at an extra meeting (not during class) with their Scrum Master, an Agile Coach, or on their own - with the help of a scheme. At the end of the semester, they can learn about the Scrum methodology in detail in a 90 minutes lecture and compare it with their experiences during teamwork.

Experiences: Experience has shown that students enjoyed teamwork. Most of the teams were enthusiastic all along, many definitely liked to create a game software. In the beginning it was harder for them to manage the tasks, nor did they understand the essence of their meetings. They kept reporting to the Scrum Master during daily Scrums, instead of cooperating. As the semester progressed, these problems disappeared in most teams. The most stubborn mistake was procrastination. Few teams managed to outgrow this.

Reflection: There was room for reflection during retrospectives. Retrospectives on the other hand, are not part of the class, they have to take place during separate, extra meetings.

Experiences: Reflection was taken more seriously in the teams, which had a Scrum Master student or an agile coach student. In this case improvement ideas were invented, some of them were proven to be useful. The other teams could use reflection questions in the form of a questionnaire, but we do not know if this was effective. This meeting gives opportunity to find ways to improve their soft skills and their understanding of Scrum values.

Students' feedback on the course: During 2021, 146 students participated in the course, of which 99 completed a feedback questionnaire at the end of the semester. 72% of the students considered the course absolutely useful, 20% mostly useful. Students rated different aspects of their team work on a scale of 1 to 5, where 5 means "absolutely". For the question "How successful have you been in realizing your own agile role: active member of development team?" the average of the scores was 4.31. Other averages: 4.65 - self-management of the team, 4.36 - motivation of the team, 4.64 - communication of the team, 4.74 - performance of the team, 4.68 - well-being in the team.

Students rated how much the course improved the four different aspects of agile software development. The average values were as follows: 4.35 - development of the theoretical aspect, 4.29 - development of the practical aspect, 4.92 - development of the skills, 4.29 - development of the attitude - all of which are considered high. It is striking that the degree of theoretic and attitudinal development is highly rated, although in this curriculum we did not stress this particularly. This may be caused by the fact that most of the students first encountered agility and Scrum at this point in their lives.

Some textual feedback were: "It was a great experience to work in a team.", "It was useful that I got acquainted with GitLab and CI tools.", "The most profitable was that I was introduced to agile software development.", "I gained insight into how software development works in an industrial environment."

4.2. Project management in IT (lecture)

Course details:

Dedicated to: graduate students, first semester

Duration: 90 minutes / week

Type: mandatory

Goals: The aim of the course is to make students understand the operation of their workplace environment, to work in project teams or lead them.

Developed aspects of agile: theory, values

Weekly schedule:

Week	Topic of the lecture
1	Basic concepts: project, project diamond
2	Corporate culture
3	Organizational models
4	Project life cycle 1.
5	Project life cycle 2.
6	Scrum methodology 1.
7	Scrum methodology 2.
8	Personality models
9	Customer reception at the highest level
10	Effective team
11	Leadership models
12	Change management
13	Burnout prevention

Teaching methods used:

Lecture: The Scrum methodology is presented on weeks 6 and 7. The following subtopics are taught with the help of a slide show: Waterfall model; Manifesto for Agile Software Development [1]; roles in Scrum, their responsibilities, advantageous skills for taking them, in the case of taking them; Scrum events, their typical course, typical mistakes; Scrum artifacts, their goals, properties, examples; Agile metrics; summary picture of Scrum operation; Scrum values; Agile leadership.

Experiences: Experience has shown that students have understood the basics of the Scrum methodology. On their exam they had to answer the simpler questions of a basic Scrum Master certification - everyone passed the exam.

Discussion: At certain points of the presentation of the Scrum theory some situations are thrown in to be discussed. These are: "We use the Waterfall model. What could be the disadvantage of this?"; "We use planning poker on our sprint planning. What could be the benefit of this?"; "Scrum values are strongly present in our team. How could this be manifested during our Scrum events?"; Students can share their opinions, arguments.

Experiences: Approx. 30% of the students got involved in the discussions and asked more questions. The discussions broke the monotony of the predominantly one-way communication, sparking attention. On the exams the questions concerning Scrum values were answered well by 70% of the students.

Students' feedback on the course:

During 2020, 77 students participated on the course, of which 33 completed the university's course feedback questionnaire. 91% of respondents gave maximal score on the structure of the course. 88% reported it was maximally helpful for their professional development. 91% stated the classes

were as interactive as possible. This can be considered a particularly high rating for a lecture, probably due to the discussions integrated into the classes. Some textual feedback were "Thank you for this course! It was a very good experience!", "It was the most enjoyable lecture of my studies so far."

4.3. Scrum Master training (seminar)

Course details:

Dedicated to: undergraduate and master students, any semester

Duration: 90 minutes / week

Type: elective

Goals: The aim of the course is to assure appropriate knowledge for taking the Scrum Master role and to develop the necessary skills and attitude.

Developed aspects of agile: theory, practice, skills, values

Weekly schedule:

Week	Project	Lesson	Other methods
1	Preparation	Introduction to Scrum	
2	Weekly Scrum	Scrum Roles; daily Scrum	Situation - A Daily Scrum, Reflection
3	Weekly Scrum	Scrum events; Retrospective techniques	Situation - Daily Scrum with Product Owner, Reflection, Schema – NVC [4]
4	Review, Retrospective, Planning		Reflection - First sprint
5	Weekly Scrum	Retrospective techniques, software tools	Reflection - First Retrospective, Schema - Retro
6	Weekly Scrum	Agile metrics	Situation - Scapegoating in the team, Reflection
7	Weekly Scrum	Scrum artifacts	Situation - A team split in two, Reflection
8	Review, Retrospective, Planning		Reflection - Second sprint
9	Weekly Scrum	Empiricism	Reflection - Second Reviews, Scheme - Review, Situation - Senior developer against agile, Reflection
10	Weekly Scrum	Scrum values, agile leadership	Situation - New Scrum Master, Reflection
11	Weekly Scrum	Scrum Master test	
12	Final Review, Final Retrospective		Reflection - Third sprint
13		Evaluation	Consultation - Final evaluation

Teaching methods used:

Lecture: Most of the classes include an approx. 20-minute lecture.

Experiences: With this solution, students learned the basics of Scrum during the semester, while time remained to apply other teaching methods as well.

Reading literature: It is obligatory to read The Scrum Guide [2].

Experiences: The end-of-semester test showed that students have known its content but knew better the theoretical parts that were mentioned during classes.

Semester-long software development project - from different perspectives: During the semester students work in teams of 3 on an administration software. The role of the Product Owner is performed by a senior student or the instructor. All students try out both the Scrum Master and developer roles. There is a total of 3 sprints during the semester. On the closing class of the sprints, students have 30 minutes per team for a review, retrospective, and planning.

Experiences: Experience has shown that during the semester students became more and more aware of the course and purpose of each Scrum event and implemented them more and more appropriately. According to their feedback they failed to truly gain experience in the role of the Scrum Master, which they practiced only for a sprint. This may also have been caused by the fact that the teams were small (3 people, including the Scrum Master). The 30 minutes allocated for a team's review-retrospective-planning triple usually proved to be too short. They would have liked to try out more retrospective techniques. By the end of the semester the products were successfully completed, but some students stated that it would not have been necessary to have such a large amount of software-development to understand how Scrum works. Moreover, perhaps a non-software development simulation project would have been enough.

Situational exercises: There are 6 situational exercises during the semester. To prepare and play them, approx. 10 minutes are needed. Students receive their own role descriptions and details of the situation via email. They don't know each other's role descriptions. Situational exercises become more and more difficult, requiring more and more theoretical knowledge, advanced skills, developed attitude. Each time someone else plays the role of the Scrum Master, and those who were not given an active role receive observational aspects. The daily Scrum situational exercise simulates an average daily Scrum of an average team that has talkative, quiet, defocused, etc. members. The second situational exercise is also a daily Scrum, but with the presence of the Product Owner. The role of the Product Owner is to distract the team by telling stories from the management. In the third exercise the point is that several members of the team blame a single team member for the failure of the sprint. The question is how the Scrum Master handles this. In the case of the "A team split in two" exercise, one issue has to be decided. Half of the team would like to choose option A, the other half option B. The Scrum Master has to deal with this situation. In the fifth exercise the topic is the resistance of the team's most experienced senior developer against teamwork. The challenge is to find a way to deal with his resistance. In the last exercise the group simulates the first meeting of a new Scrum Master and his team. There are also skeptical and optimistic members of the team. The task of the Scrum Master is to start a good cooperation with his new team.

Experiences: The experience was that students got involved in their roles - although they had to implement it online because of Covid-19. As the semester progressed, the Scrum Masters were more able to handle different characters (e.g. quiet, talkative, etc.). The students liked the situational exercises, found them useful, even more of them would have been welcome.

Reflection: Reflection is present several times during the course. Primarily, situational exercises are always followed by reflection. Students who were given observer roles are the first who reflect on the situations. After that each actor is able to tell what they experienced.

Experiences: The time allocated for reflection was at least 10 minutes, but it always turned out to be too short. Soft skills and Scrum values were in the focus of the reflections and students formulated ideas to develop this aspect of Scrum.

Secondly, the Scrum retrospectives are a reflection on each sprint. The teams look back, analyze the sprint along aspects chosen by the Scrum Masters and search for ways to optimize their processes.

Experiences: The development in operation of the teams was observable during the semester.

After closing the sprints, there is a reflection on the quality of the reviews and retrospectives. The goal is to learn how this can be lead on a high quality.

Experiences: During the semester, sprint ceremonies got better and better - more focused, more concise, smooth.

Schemas and examples: After seeing that the first sprint reviews and retrospectives were not satisfactorily effective, schemas of a retrospective and a review were presented.

Experiences: After the presentation of these schemas, these Scrum events have become smoother and more professional.

Consultation:

Experiences: A Scrum Master turned to advice with the problem that one of his team members does not work, neither responds to messages. The team did not know if they can count on him, which makes the team insecure. The Scrum Master was introduced to the method of NVC [4] and was given an example of how he could communicate with the problematic team member. The Scrum Master tried out NVC through posting on message board and the missing student responded. The problematic student officially and peacefully left the team, which helped the team to deal with the situation. At the end of semester, students received feedback one by one on their strengths and weaknesses in order to have the possibility to become good Scrum Masters. The students were very grateful for the personal feedback.

Students' feedback on the course: During 2021, 9 students participated in the course, of which 7 completed a feedback questionnaire at the end of the semester. 28% of the students (2 students) were completely satisfied with the training, 72% (5 students) were mostly satisfied. 42% (3 students) found the situation exercises absolutely useful, 58% (4 students) mostly useful. According to their votes, situational exercises contributed mostly understand Scrum roles and their responsibilities (6 votes), team dynamics (6 votes), handling problems (5 votes) and Scrum values (5 votes each). In textual feedback, several students indicated that they really liked that the exercises were based on real situations. Situational exercises were mentioned as the best part of the course by 72% (5 students). However, project work was not popular within this course. 58% (4 students) indicated that the work was too much, 42% (3 students) stated that change is needed regarding project works in future semesters. On the question "How useful did you find the project work?", the following scores were received: 28% (2 students) - a little, 14% (1 student) - fairly, 42% (3 students) - mostly, 14% (1 student) - absolutely.

Students rated how much the course improved the four different aspects of agility on a scale of 1 to 5, where 5 means "absolutely". The average values were as follows: 4.57 - development of the theoretical aspect, 4.71 - development of the practical aspect, 4.28 - development of the skills, 4.57 - development of the attitude - all of which are considered high. A textual feedback: "I really enjoyed the course. It is good that we can acquire such up-to-date knowledge at university. Keep it up! Lots more! :)"

4.4. Software development in practice (practice)

Course details:

Dedicated to: master students, second semester

Duration: 90 minutes / week

Type: optional

Goals: The aim of the course is to simulate an industrial software development situation. In 2021, each student played the role of the Scrum Master in a semester-long software development project, realized by a team of 3 undergraduate students.

Develops the following aspects of agile: theory, practice, skills, values

Weekly schedule:

Week	Action during class	Action during the week
1	Theoretical preparation	Meeting the Product Owner
2	Theoretical preparation	Team building, Planning
3	Consultation - Subgroup A	Weekly Scrum, preparation to Review
4	Consultation - Subgroup B	Review, Retrospective, Planning
5	Consultation - Subgroup A	Weekly Scrum
6	Consultation - Subgroup B	Weekly Scrum, preparation to Review
7	Consultation - Subgroup A	Review, Retrospective, Planning
8	Consultation - Subgroup B	Weekly Scrum
9	Consultation - Subgroup A	Weekly Scrum, preparation to Review
10	Consultation - Subgroup B	Review, Retrospective, Planning
11	Consultation - Subgroup A	Weekly Scrum
12	Consultation - Subgroup B	Weekly Scrum, preparation to Review
13	Closing	Final Review, Final Retrospective

Teaching methods used:

Semester-long software development project - from the Scrum Master's perspective: Every student of the Software development in practice course is the Scrum Master of a team from the Software technology course. During the semester, they have to support the team, participate in their daily Scrums, Reviews, coordinate Retrospectives and Plannings a total of 4 times.

Experiences: The students liked the Scrum Master role and got more and more involved. Because Retrospective and Planning did not fit in the timeframe of the classes (there was no time left after the Reviews), it was often difficult to schedule a separate meeting for them. There were about 4-5 Scrum Master students who had greater difficulties with their team, e.g. lazy team member, split team, misunderstandings in the team. Typically, the Scrum Masters hoped the situation would resolve on its own and found it difficult to admit that there is something wrong. At the same time about 20 Scrum teams worked well, the Scrum Masters reported better and better communication, cooperation, focused meetings and successes. About 3 Scrum Masters have tried out delegation by the end of semester and held creative retrospectives.

Lecture: The semester begins with two 90-minute lectures, which briefly presents Scrum events, roles, products, Scrum values, Agile leadership. During the presentation of the theory, we also discuss exactly how to implement them in the particular case of the course.

Experiences: After the two preparatory classes, the students all passed a theory test and were able to begin their work with their teams. Later, only a few questions arose regarding Scrum's theory.

Reading literature: The Scrum Guide [2] is a must read at the beginning of the semester. There are questions in the test that can be answered only by reading the document.

Experiences: The students were less likely to answer the questions, but still everyone passed the test.

Discussion: The two preparatory lectures at the beginning of the semester are enriched by small discussion questions. For example: "Our team complains that they do not understand what the job is. We, as Scrum Masters, contact the Product Owner, ask him what the job is and then explain it to the team. Is this a good approach?", "A team member is missing from a meeting. We are not worried, she will definitely come next week. Is this a good attitude?" etc.

Experiences: The students got involved in the discussions, made several arguments, and the solution always tipped in the right direction.

Schemas and examples: During the theoretical preparation, students receive schemas regarding situational leadership model, how to ask questions, how to give feedback, Scrum values as compass in decisions.

Experiences: Later, during the discussions, it was observable that the students thought along the Scrum values. Some Scrum Masters reported that they used the feedback giving scheme and found that it was a success.

Reflection: After the theoretical preparation, during each class a subgroup of the students reports on what they experience in their teams. Sometimes they do it based on reflection aspects, such as "What is the biggest weakness of your team?", "How has your team developed till now?" etc.

Experiences: Students thought over the aspects and sometimes formulated things that were new to themselves. Often, they articulated aloud what they would do differently next, and later reported that they had tried and succeeded.

Consultation: During classes, the instructor can provide personalized advice to the student.

Experiences: For example, one Scrum Master reported that his team is very enthusiastic but very dependent on him, always waiting for him to decide, organize etc. He was advised to try to delegate a daily Scrum. The student later reported that he had tried it and it was a good experience for the whole team that they were capable of self-management. In another case, a Scrum Master repeatedly reported that he is frustrated because his team is quiet, restrained. He was advised to try to find out what is behind this, e.g. lack of courage, openness or commitment, or it is simply the dynamics of his team. Later, the Scrum Master reported that he realized Scrum teams can be different, even quiet, restrained, and that does not necessarily mean there is something wrong. Overall, it was obvious that students are happy to receive personalized advice and find them helpful. Moreover, those who only witness the consultation, also learn from the other Scrum Master's situation.

Students' feedback on the course: During 2021, 25 students participated in the course, of which 24 completed a feedback questionnaire at the end of the semester. 66% of the students considered the course absolutely useful, 25% mostly useful. Students rated how much they could be helpful in their teams on a scale of 1 to 5, where 5 means "absolutely". The average values were as follows: 3.91 - helpful in general, 3.58 - helpful in planning, 3.70 - helpful in preparing reviews, 4.29 - helpful

on retrospectives, 4.62 - helpful in weekly Scrums, 4.45 - helpful in clearing impediments, 4.08 - helpful in constantly developing team processes. For the question "How successful have you been in realizing your own agile role: a leader who supports and serves a self-organizing team?" the average of the scores was 4.29.

Students rated how much the course improved the four different aspects of agile software development. The average values were as follows: 4.66 - development of the theoretical aspect, 4.12 - development of the practical aspect, 3.92 - development of the skills, 4.62 - development of the attitude - all of which are considered high. Some textual feedback were: "The course gave a good chance to try the theory in practice.", "Now I see the Scrum Master's point of view", "I learned the Scrum Master does play an important role.", "I learned it is not easy to be a Scrum Master."

4.5. Agile Research Team (research project)

Course details:

Dedicated to: master students, all semesters

Duration: 90 minutes consultation / week

Type: elective

Goals: The aim of the Agile Research Team is to study research results on the topic of agile methodologies, practices, principles and carry out new experiments.

Developed aspects of agile: theory, practice, skills, values

Research topics:

Topic	Students	Since
Developing methods for supporting the Scrum Master training	2	1, 5 year
Developing methods for supporting effective Scrum Retrospectives in education	1	1, 5 year
Analyzing a new hybrid agile methodology	1	1, 5 year
Analyzing the impact of agile transformation on software developers	1	1 year

Teaching methods used:

Reading literature: When someone joins the research team, he/she definitely has to read some important literature. These include the Manifesto for Agile Software Development [1] and the Scrum guide [2].

Experiences: Experiences had shown that this has built a common understanding of basic concepts and later everyone could join the discussions.

Consultation: Each student chooses a research goal after enrollment. Our weekly meetings are actually joint consultations in which everyone reports on what they have achieved since the last meeting, what impediments they have met, what plans they have until the next meeting. The instructor and also other students give advice.

Experiences: With this method the team has reached to publish 4 articles in conference proceedings and 5 dissertations are going to be presented this semester.

Discussion: Sometimes, there are spontaneous discussions on meetings.

Experiences: Students argue with each other about what questions or answers are relevant to a particular Scrum-related questionnaire that some of them creates, for example. Based on this, more nuanced questionnaires were created than those originally compiled by a single student.

Literature review: Some of the research topics are literature focused.

Experiences: The student with the topic "Analyzing the impact of agile transformation on software developers" firstly explored the causes, course etc. of the agile transformation from literature. Based on this, he was able to create a detailed, exciting questionnaire for his research. The student dealing with topic "Analyzing a new hybrid agile methodology" also started with the collection of literature on hybrid methodologies, and then was able to describe a completely new hybrid agile methodology he invented and used before.

Monitoring of several software development teams - from the perspective of an agile coach: The student with the topic "Developing methods for supporting effective Scrum Retrospectives in education" visited a few Software technology teams at the end of each sprint.

Experiences: She tried out different retrospective techniques with them. She gave tips on correcting their weaknesses, based on the literature and her own experiences – which grow from sprint to sprint. She was able to build an entire collection of tips on handling various typical team weaknesses, which she also incorporated into one application. This app aims to automate Scrum Retrospectives. It uses a form to assess the teams situation and then gives tips based on it.

Reflection: Two students with the topic "Developing methods for supporting the Scrum Master training" attended the training as Product Owners. In addition, they have witnessed all the situational exercises. After the lessons, we always reflected separately on what they had seen and learned from the situation, how it could be done differently.

Experiences: The two students came to deeper and deeper understanding of Scrum and were able to perform the role of the Product Owner quite well.

Students' feedback on the course: Students reported on how they benefited from being a member of the Agile Research Team: "I was able to study closely the topic of agile values, which basically lacks literature.", "I saw agile software development from many perspectives by following the research of my colleagues.", "We gave each other good ideas. We were a small creative community, and we were catalyzing each other's work."

5. Discussion

As the curricula and experiences described above demonstrate, the four aspects of agile software development can be taught at university.

Teaching of the theory of agile methodologies could be incorporated in project management related lectures, as in the case of our Project Management in IT course. Such lectures make possible to compare agile methodologies with other methodologies, which leads to a better understanding of them. Teaching of theory can be well complemented with teaching of values by starting discussions during lecture. Discussions help deepen values as students come up with supporting arguments. The discussions also break the monotony of the lectures.

The practical aspect of an agile methodology can be taught on a practice course, by using it for managing software engineering projects of teams of students. There are several examples of this teaching approach in the literature, according to [5] this is the most common form of teaching agile methodologies. Cases differ mostly in assigning the roles. Product Owner role is commonly taken by an instructor or other expert, such as in cases [6,7,8,9,10], but there are also examples of Product Owner students [12,13]. In the case of the Scrum Master role, the opposite is more typical, so mostly students fill the Scrum Master role [6,7,9,11,12]. However, there are also examples where the instructor fills the Scrum Master role [8,13]. If an instructor fills a Scrum role, it is usually advantageous, because he/she fills it more authentically due to his/her experience. If a student fills a Scrum role, experience is not guaranteed, but in return we give the student the opportunity to gain experience.

In our case, assigning the Product Owner role to the instructor (in case of each course presented in this paper) seemed to be a good solution, as the evaluation of the products at the end of the semester was also his task. This way he could authentically represent the product requirements. On the Software Technology course, students played only the role of developers, and the Scrum Master role was played by senior students from the Software engineering in practice course. We consider this a good solution because – being their first teamwork at the university – undergraduate students needed help in managing teamwork. Assigning the Scrum Master role to senior students was an efficient solution, because in this way the teams have got more attention and senior students gained more experience in Scrum. Student teams who had a senior student as their Scrum Master assigned an average value of 4.65 on how much the Scrum Master student helped their work. Senior students who were able to try themselves in the role of the Scrum Master gave an average rate of 4.60 on how useful this opportunity was for them. In addition, they felt that they could help their team in an average of 3.91. The three high values further validate the success of connecting the Software technology and the Software development in practice course. However, we had to be aware that some Scrum Master students tend to wait too long for their teams to solve problems among themselves, others may intervene too soon and not support self-sufficiency enough, as mentioned in [12]. We supported the operation of student Scrum Masters with providing weekly consultations with an instructor, in line with recommendation form [5]: "one of the instructors should still play the supervisor Scrum Master for the whole process". We considered the connection of the two courses very successful because we have very few human resources at our university and several practical courses. [12] presents a successful method of teaching the practice of Scrum that they claim need a large staff (up to 11 supervisors). We managed to achieve something similarly effective by connecting the two courses, and even saved significant resources in terms of university staff.

According to the literature, another frequent way to teach the practice of agile methodologies is to play educational games. Some of these are presented in [14,15,16,17]. There has been no example of this at our university yet, but in the case of the Scrum Master course, we should seriously consider this option. The feedback was the students were the least convinced by the efficiency of the semester-long project among all the methods used on this course. They considered the effort needed too high compared to the lessons learned. [5,18] point out that Scrum games are suitable for demonstrating the practical operation of Scrum in short time. Due to little time required, we could consider making a short Scrum simulation even on the Project Management in IT lecture in some creative way, e.g. paper city construction [19]. This game is cheap compared to Lego based Scrum games [14].

The basic theory and practice of Scrum can and should be taught on mandatory courses because of their popularity. However, we should not forget about teaching agile skills and values too. Our

experience has shown that if we want to teach in depth and strengthen all four aspects then optional or elective courses are more suitable.

The importance of teaching agile skills and values are discussed in multiple articles of Martin Kropp and Andreas Meier [7,18,19]. In their Agile Competence Pyramid model, Agile Values are placed at the highest level of being agile [7]. They defined three principles for teaching agile comprehensively: 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 [18]. We believe keeping these three principles in mind while designing a course is in line with developing all four aspects of agile methodologies.

In our case, the Scrum Master training and the Software engineering in practice integrated all three principles through semester-long project works, reflection, discussions. Besides, it developed all four aspects of agile methodologies. On both courses, the same theoretical material was presented, and the same literature was obligatory to read. The difference is that in case of the Software development in practice course, the knowledge was passed on in form of two 90-minute lectures, while the Scrum Master training was divided into short, 20-minute lectures. Both solutions proved to offer a very good understanding of Scrum theory (averages higher than 4.50). As part of the Scrum Master training, 6 situational exercises and projects of 3 ensured that students become familiar with the practice of Scrum. Students found the situational exercises very useful, but the project-work less so. They said the teams were too small and no real lessons were drawn. The time invested into projects was also too much. In comparison, we received a lot of positive feedback on the method where students played only Scrum Master roles of project teams of another course – like in case of the Software development in practice course. In addition, there were many opportunities for consultation, reflection and discussion. Students found this solution very useful. In the case of Scrum Master course, students could observe 6 typical industrial Scrum situations closely through situational exercises. In the case of the Software development in practice course, students could observe only one team's functioning closely - but this was more real, not a simulation. Both agile practice experience and soft skill development were rated to be more noticeable in the case of the Scrum Master course (4.71 - practice, 4.28 - skills), and they received high ratings in the case of the Software development in practice course as well (4.12 - practice, 3.92 - skills). We think the difference is caused by the use of situational exercises in the Scrum Master course, which have put students in difficult real-life situation, thus developing their knowledge of real-life Scrum effectively. Agile values development was rated in a similar way: 4.62 in the case of the Software development in practice course, 4.57 in the case of the Scrum Master training course. The advantage in favor of the Software development in practice course could be due to more time allocated for discussions and consultations.

Based on the similarly high rates, we consider both the Scrum Master training and the Software development in practice course adequate for teaching all four aspects of agile software development. There were two students who participated on both courses. They agreed the two courses were good one by one, and they strengthened each other as well. A textual feedback was: "I think it is worth completing several courses, especially Scrum Master training and Software engineering in practice, because together they give a much more comprehensive picture of Scrum."

Thus, we consider agile methodologies could be taught most widely on elective and optional courses, the research team is more suitable for teaching in depth. In case of Agile Research Team, each student has his own research topic. We can choose different methods of helping them based on their topic, e.g. reviewing, assigning the role of an agile coach or Product Owner. Every student can be supported by consultation, which also supports all other team members in learning about agile at the same time.

Courses teaching various aspects are best placed in such a way that, proceeding from undergraduate studies to graduate studies, previously taught material is always supplemented in depth or variety. In our case students meet agile software development first in the Software technology mandatory course. Here they gain basic knowledge about Scrum roles, events, artifacts, and experience being a developer in a Scrum team. Secondly, they deepen their theoretical knowledge on agile software development in the Project management in IT course, which is a mandatory course for first year graduate students. This course is enriched with discussions to give insight into agile values. If students get interested in Scrum by these obligatory courses, we provide them optional or elective courses to expand their knowledge. The Scrum Master training and the Software engineering in practice courses both focus on developing skills and values of agile software development. Besides that, they highlight the perspective of the Scrum Master. If a student wants to gain even more knowledge on agile software development, he/she can join the Agile Research Team. This allows students to thoroughly study what the most inspiring part of agile software development is for them.

As for some of the curricula and conclusions of this article, it is important to note that we can consider them validated only in a modest way due to low number of feedback. For example, in case of the Scrum master training only 7 people filled out our feedback survey.

We considered heavy focus on developing skills and attitudes the essence of Scrum Master training, which is most effective on small courses. However, because of the small number of participants, it is very difficult to collect an amount of data that can be considered sufficient to validate the curriculum. The current version of the Scrum Master training presented in this article was attended by 9 people and 7 people provided feedback on the course. In terms of validation, however, it may be interesting that a previous version of this course had been attended by a total of 2x9 students in previous years, and the feedback pointed in a similar direction. The course differed in that there was only one situational exercise throughout the semester, and slightly more theoretical material was delivered in the form of a lecture. 16 students provided feedback on the previous version of the course. The averages of the evaluations were as follows (notation: aspect {c = current course average, o = old course average}) satisfaction with the course {c = 4.28; o = 4.12}; usefulness of project work {c = 3.42; o = 3.50}; usefulness of situational exercises {c = 4.58; o = 4.60}. In the text-based feedback, 42% (3 students) mentioned that they would change the project work of the current course, and 37.5% (6 people) in the case of the old one. Thus, it can be seen that although we have a small number of measurements, they point in one the direction, pointing out the weak (project work) and strong points (situational exercises) of the curricula.

Another interesting observation is that the satisfaction related to the courses may have been influenced by students' prior knowledge of Scrum and agility.

There seems to be a tendency for students who had practical experience on Scrum to vote lower values for several aspects of the courses. For example, in the case of the Scrum master course (current (2021) and old one (2020) as well), we asked students what experience they had with Scrum before the course. We call beginners those who knew something about Scrum theory at most and advanced ones who have had some practical experience with Scrum before. Roughly half of students were beginner, half were advanced. The averages were as follows (notation: aspect {b = average of beginners, ad = average of advanced}) satisfaction with the course {b = 4.63; ad = 4.00}; usefulness of project work {b = 3.91; ad = 3.16}; development of the theoretical aspect {b = 4.58; ad = 4.26}; development of the practical aspect {b = 4.65; ad = 4.19}; development of the skills {b = 4.58; ad = 3.87}; development of the attitude {b = 4.20; ad = 3.66}. The biggest difference is in the assessment of the usefulness of the project work (0.75), but there is also a big difference in the overall satisfaction with the course (0.65). In the case of Software technology

course, we can also see that development along different aspects was rated by the students very high (at least 4.29 in all cases), although this course did not involve many different methods. Thus, it emerges that the curricula presented in this article are mostly successful for beginners, and the methods defined in the article also develop beginners the most.

Since the experiments took place in 2020 and 2021, we cannot miss the question of how the Covid 19 might have influenced it - although this is not the focus of this article. At our university, from mid-March 2020 to present, the education took place only online (because of the Covid 19 restrictions). We used MS Teams and realized live online teaching on all courses presented.

We can give a measurement-based estimate on the effect of Covid 19 only on the courses whose structure have not changed since 2019. This holds only for the Project management in IT lecture. In 2020, for the first time, the lecture took place online, in the form of an MS Teams meeting. If we look at the data, we can see that students rated the online course a bit higher than its offline version: (notation: aspect {on = online course average; of = offline course average}) structure of the course {on = 4.88; of = 4.50}; professional development by the course {on =4.81; of = 4.15}; interactivity on the course {on = 4.81; of = 4.55}. The author's own conclusion was that in an online environment, students appreciated even more if one lecture was interactive. Some students preferred to write in a joint chat rather than comment on the discussions in a video voice call. Slightly fewer people took part in the discussions, but they were still involved. Students also said they were able to learn more through a course because they could replay the lesson recordings. For the other courses, we can only rely on the verbal feedback of the students or the information provided on the Sprint Reviews. Some students, especially in the case of teamwork, missed face-to-face meetings. At our call, several students organized weekly 1-3 hour long online coding sessions together and they admitted it helped them a lot. Short, daily meeting - like weekly Scrums - were considered by many to be logistically easier to organize online.

Thus, the shift to online teaching caused by Covid 19 at first seemed to be a challenge. But after all, our experience was that by finding the right online tools, we were able to achieve the same performance as before on the courses presented.

Overall, we discussed how agility can be taught at university. The uniqueness of this article compared to the others is that it first defines separate methods to teach the four aspects and then shows how they can be incorporated into different curricula or an entire university program. Compared to other articles, which usually present the implementation of a single curriculum, this article presents many methods and their embedding and combinability in different curricula. Based on the measurements related to the implemented curricula, the methods appear to be successful. However, their success is recommended to be confirmed by further measurements.

6. Conclusion

Teaching of the four aspects of agile software development can be realized in classic university courses,

Teaching of theory can be integrated into many types of courses. Discussions can complement teaching theory and thus ensure understanding of agile values. The practice of agile software development can most effectively be taught by semester-long projects and situational exercises, in practices and seminars. Reflections on practical situations and consultations can amplify skills development. Choosing an inspiring topic related to agile software development and working in an agile research team can be beneficial in various ways.

Incorporating teaching of agile software development into multiple types of courses in various ways is a winning strategy because it gives students opportunity to master agile software development.

In the introduction of this article, we defined three questions:

1. Can different aspects of agile software development (theory, practice, skills, values) be taught at university?
2. What kinds of methods can be used to teach the aspects of agile software development?
3. In what kind of curriculum can these methods be integrated?

In response, we defined 14 methods for developing the four different aspects, with at least 3 methods for each aspect. In the field of practice development, these are in line with those prevalent in the literature. The methods were inserted into 5 different curricula, which we implemented. We checked their effectiveness with a total of more than 130 students. The results showed that the students found that they had developed along the four different aspects on the courses. In the case of every aspect at least an average of 70% development were measured. These achievements provide a positive answer to our question that agile methodologies can be taught at university. The number of involved students is moderated, so it is recommended to collect additional responses, possibly not based on self-assessment. However, in the field of teaching agility, measurement methods are a challenge in other research as well.

7. Acknowledgment

The research project was supported by the European Union and co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

Bibliography

1. *Manifesto for Agile Software Development* (2001) https://agilemanifesto.org/
2. K. Schwaber, J. Sutherland: *The Scrum Guide* (2020) https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf
3. *14th Annual State of Agile Report* (2020) https://stateofagile.com
4. M. Rosenberg: *Nonviolent Communication*. Puddle Dancer Press, Encitas (2015)
5. V. Mahnič: *Scrum in software engineering courses: an outline of the literature*. Global Journal of Engineering Education, 17 (2015) 2. World Institute for Engineering and Technology Education (2015) 77-83
6. D. Damian, C. Lassenius, M. Paasivaara, A. Borici, A. Schröter: *Teaching a globally distributed project course using Scrum practices*. CTGDSD 2012, Zurich (2012) 30-34 DOI: 10.1109/CTGDSD.2012.6226947

7. M. Kropp, A. Meier: *Teaching agile software development at university level: values, management, and craftsmanship*. CSEE&T 2013, San Francisco (2013) 179-188 DOI: 10.1109/CSEET.2013.6595249

8. V. Mahnič: *A capstone course on agile software development using Scrum*. IEEE Transactions on Education, 55 (2012) 1. IEEE (2012) 99-106

9. M. Paasivaara, C. Lassenius, D. Damian, P. Raty, A. Schröter: *Teaching students global software engineering skills using distributed Scrum*. ICSE 2013, San Francisco (2013) 1128-1137 DOI: 10.1109/ICSE.2013.6606664

10. S.D. Zorzo, L. De Ponte, D. Lucrédio: *Using scrum to teach software engineering: a case study*. FIE 2013, Oklahoma City (2013) 455-461 DOI: 10.1109/FIE.2013.6684866

11. G. Rodríguez, A. Soria, M. Campo: *Teaching Scrum to software engineering students with virtual reality support*. Lecture Notes in Computer Science 7547 (2012) 140-159

12. A. Scharf, A. Koch: *Scrum in a software engineering course: an in-depth praxis report*. CSEE&T 2013, San Francisco (2013) 159-168 DOI: 10.1109/CSEET.2013.6595247

13. T. Reichlmayr: *Working towards the student Scrum - developing agile Android applications*. ASEE Annual Conference & Exposition 2011, Vancouver (2011) 159-168

14. M. Paasivaara, C. Lassenius, V. Heikkilä, T. Toivola: *Teaching students Scrum using LEGO blocks*. ICSE 2014, Hyderabad (2014) 382-391 DOI: 10.1145/2591062.2591169

15. C.G.Von Wangenheim, A.F. Borgatto: *SCRUMIA - an educational game for teaching SCRUM incomputing courses*. Journal of Systems and Software, 86 (2013) 10. Elsevier (2013) 2675-2687

16. S. Ramingwong, L. Ramingwong: *Plasticine Scrum: an alternative solution for simulating Scrum software development*. Lecture Notes in Electrical Engineering, 339 (2015) 851-858

17. J. M. Fernandes, S.M. Sousa: *PlayScrum - a card game to learn the Scrum agile method*. VS-GAMES 2010, Braga (2010) 52-59 DOI 10.1109/VS-GAMES16805.2010

18. M. Kropp, A. Meier, M. Mateescu, C. Zahn: *Teaching and learning agile collaboration*. CSEE&T 2014, Klagenfurt (2014) 139-148 DOI: 10.1109/CSEET.2014.6816791

19. S. Hof, M. Kropp, M. Landolt: *Use of gamification to teach agile values and collaboration: A multi-week scrum simulation project in an undergraduate software engineering course*. ITiCSE 2017, Bologna (2017) 323-328 DOI: 10.1145/3059009.3059043

Authors

ILYÉS Enikő

Eötvös Loránd University, Faculty of Informatics, Department of Software Technology and Methodology, Hungary, e-mail: ilyese@inf.elte.hu

About this document

Published in:

CENTRAL-EUROPEAN JOURNAL OF NEW TECHNOLOGIES IN RESEARCH, EDUCATION AND PRACTICE

Volume 3, Number 2. 2021

ISSN: 2676-9425 (online)

DOI:

10.36427/CEJNTREP.3.2.2378

License