Learning from Agile Methods: Using a Kanban Board for Classroom Orchestration

Sven Strickroth[1][0000-0002-9647-300X], Melanie Kreidenweis[1][0000-0001-8659-525X] and Zora Wurm[1]

[1] Institute for Informatics, LMU Munich, Oettingenstraße 67, 80538 Munich, Germany
sven.strickroth@ifi.lmu.de

Abstract. Task-driven learning arrangements have shown to be a good learning opportunity for students and provide a viable way for differentiation and self-paced learning. However, existing approaches do not provide an overview of the current status of tasks to the teacher in collaborative settings at the same time. This overview is a crucial prerequisite for effective mentoring and guiding the students. Agile methods have shown to successfully address similar issues for planning, structuring, and visualizing the work in software engineering. In this paper a teaching method based on a Kanban board is presented which transfers the principles and advantages of agile methods to the domain of teaching for classroom orchestration. Furthermore, an offline usable digital prototype for supporting agile teaching scenarios was developed that allows teachers to easily prepare such scenarios and students to document and visualize the current work status. Lessons learned and recommendations for the agile teaching method are discussed and a preliminary study shows that the digital tool supports students and teachers on their learning/teaching path and has a good useability.

Keywords: Collaboration, Agile Teaching, Teaching Method, Teaching Support Tool, CSCL, Kanban.

1 Introduction

Orchestrating classroom activities and tasks is one of the main duties of a teacher in schools. Task-driven learning arrangements have shown to be a good learning opportunity for students. Oftentimes station teaching or other work plans are used to orchestrate tasks for single students or group work. A central advantage of such methods is to enable students to self-regulate and pace their learning [1, 2]. However, the main problem with these approaches is that the teacher cannot easily get an overview of the current status of the tasks (open, being worked on, and done), especially when several students are working collaboratively or cooperatively in multi-step learning settings. This overview, however, is crucial for mentoring and guiding the students in their learning because feedback (or different forms such as feed back and feed forward) is a driving factor for learning [3].

The approach for supporting teachers and students presented in this article relies on agile methods that are widely used in the software development industry to structure a project and provide a “simple” framework (such as Kanban or Scrum) which puts a focus on visualizing the workflow, communication, and reflection on past work amongst the team members [4,5]. The research questions addressed in this article are: (1) How can the mentioned agile principles and methods be mapped to the school context to orchestrate tasks to support self-regulated learning? and (2) How can such an agile teaching method be supported by technology? These research questions are investigated in the context on computer science classes but should be generalizable to other subjects as well. The key contributions of this paper are:

- proposal of an agile teaching method,
- proposal of a free digital support tool prototype for teachers and students,
- lessons learned of the teaching method and evaluation of the digital tool.

We applied a Design-Science approach [6] to develop the proposed teaching method, tested it in case studies, and refined it in several iterations over the last five years. During the first iterations the basic model was developed and optimized. Here, fitting agile methods for the classroom were selected and it was investigated on how to implement these (e.g., iterations, definition of finished work, selection of tasks). In further iterations several workshops were held with training teachers (mentors), teachers and (student) teachers to present and discuss the approach. Finally, a specialized digital tool was designed and evaluated to support the agile teaching method (preparation and in-class support). New ideas and lessons learned were fed back into the next iteration (for example resulting in a new variant of the method or guidelines for preparation of fitting educational material design).

The remainder of this article is structured as follows: First, the related research is discussed and the research gap is identified. Second, the general approach and the digital tool are described. Third, the lessons learned and the qualitative evaluation of the digital tool are presented. The paper ends with a discussion and conclusions.

2 Related research

In this section the related work for classroom activity orchestration, agile methods, usage of agile methods in universities/schools, and approaches for computer-supported collaborative learning (CSCL) settings are presented.

There are different teaching methods such as project work and station work to enable self-paced and self-regulated learning [1]. In the case of station work, the teacher prepares the learning content as tasks that the students should work on and spaces these out on different tables in the classroom. Then, the students can work freely on these tasks according to specific rules in a given time frame (i.e., there might be dependencies of the tasks or some tasks are optional, etc.). Project work is usually conducted in groups whereas station work can be either conducted in groups or by individual students working on the tasks. As outlined in the introduction, a drawback is that the teacher might not be aware of the progress of all students. To improve the overview for the teacher, there were approaches proposed that use special devices (e.g., called ‘Lantern’ in a project) that indicate the progress of students [7]. This requires, however, special equipment that is not common in regular classrooms.

Agile methods support the organization and execution of software development projects and are widely used in the professional sector nowadays [4]. The roots lie in the "Manifesto for Agile Software Development" which defines 12 basic principles [8]. Key features of agile approaches are continuous learning, a high flexibility, focus on the product, and an iterative procedure containing a planning and a testing/reflection phase. There are different agile frameworks such as Kanban or Scrum that can be used to structure the development, visualize workflows, and promote communication and reflection on past work amongst the team members [5]. The Kanban board approach works as follows (simplified): It is based on a board with three columns that gives an overview on the “backlog” (tasks to do), tasks actively worked on and finished tasks where tasks (e.g., symbolized as cards) “travel” from the “to do” to the “done” column using a well-defined procedure. This visualization as well as regular meetings help to orchestrate the work and improve communication between the developers. Teaching agile practices are part of the software engineering curricula in many universities worldwide [9]. Agile methods have also been used for software development projects in schools in computer science classes for learning programming and/or training agile competencies [10, 11], however, not as a generic orchestration tool or teaching method that can be applied in other subjects. There are also applications outside teaching for supporting school development [12].

Existing web-based professional software tools for agile management such as Atlassian Trello that uses a Kanban board or more generic tools such as the collaborative bulletin board Padlet turned out to be not usable in the school context: These web-based tools require an account and are not compatible to German privacy laws (based on the GDPR, especially as the students are not of legal age). Additionally, such tools are not designed for the school context as they do not allow to prepare a “board” and clone it (e.g., for several groups working independently on the same tasks), contain too many professional features, and also require an active internet connection which is not always available in schools [13].

There are also specialized tools to plan lessons and, more generally in the context of CSCL, also conduct concrete teaching scenarios. All such tools, however, are not optimized for using agile teaching methods and/or have different limitations: In general, existing lesson planning tools such as PLATON provide no specific support for specific teaching methods [14] or are limited to a specific teaching method such as Collage with the Jigsaw method [15]. Furthermore, other lesson planning and Learning Design tools such as LAMS [16] allow to model lessons with predefined activities, conditions, and groupings. During the lesson students need to log-in in LAMS and work through the steps to the end of the sequence (collaboratively) on their own computers. Hence, such tools are often restricted to scenarios in that computers are required for all students and used for most steps [14]. The very same restriction also applies to other Learning Design tools and, additionally, Learning Designs or CSCL scripts are also often not easy to model for teachers [14, 17].

Summing up: There is a research gap for the application of agile methods in generic teaching scenarios as a self-regulated learning orchestration tool which makes students’ progress visible, and availability of a tailored digital tool to support teachers and students.

3 Applying Agile Methods as General Teaching Methods

To address the identified research gap, the main idea of the proposed teaching method is to transfer the agile project management's benefits of prioritizing tasks, selecting tasks, discussing how to work on these, the self-paced working on tasks, visualization of the workflow, as well as the defined process for testing/assessing the correctness of the solution, and the reflection on the process after an iteration to the school context.

The proposed agile teaching method is based on the Kanban board and the Scrum frameworks (cf. [5, 18]). The teacher takes the role of the customer who creates the user stories (i.e., the tasks) and prepares the sprints (i.e., iterations with different sets of tasks) in which products (i.e., solutions) are developed and/or refined. The students take the role of the "developers" and can work in groups or as single students on the tasks. Each round is supposed to have an outcome that is finished (potentially shippable product increment) in the sense of a learning product. There are seven central principles for the agile teaching method (cf. [18–21]): *transparency, definition of Done, pull-principle, commitment, self-organization of the team*, *continuous improvement* and *timeboxing*.

Transparency is the key concept to visualize the workflow and the progress of the students' work. Typically, the Kanban board has three columns "to do", "in progress" and "done" that visualize the current iteration. As boards magnetic blackboards, pin boards, glass fronts, windows etc. can be used. On the board, the tasks can be symbolized by cards or post-it notes and are assigned to one of the three columns (cf. Fig. 1). Students can work in groups or as single students on the tasks by having their own board. The Kanban board provides information about the state of tasks (which students work on which tasks and the engagement of the students). This information is visible to the students who can see their own progress (and maybe the progress of other students) as well as the teacher who is able to monitor the progress from the distance without interrupting the students in their work. A range of problems can be identified easily, e.g. skipping of tasks, when students are working on a task for too long or when too many tasks are worked on in parallel. The teacher acts as a Scrum Master and can recommend splitting a task into smaller subtasks or limiting the number of parallel tasks by moving tasks back to the backlog.

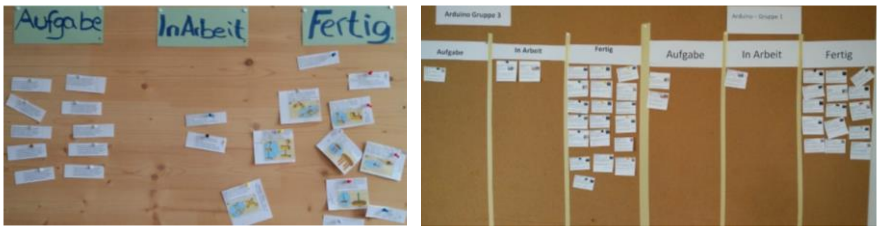


Fig. 1. Two examples of agile boards in teaching situations

The definition of Done is summarized by the agile saying "Done is Done". Tasks can only be moved to the "done" column when the acceptance criteria specified by the teacher are met. This means that there is no additional work needed to finish the task and that tasks should only move in one direction from “to do” to “done”. The definition of Done resp. the acceptance criteria for completing a task needs to be clearly specified by the teacher, e.g. bugfree/tested software, spell-checked text, practiced presentation, prepared notes for verbal contributions, etc. The clear specification of expectations and clear communication of the scope of the task help students to direct their actions and learning. The assessment on whether the acceptance criteria are met can be conducted by the teacher (as the customer) or by other fellow students (i.e., peer review).

The *Pull-Principle* allows the students to self-select a task from the backlog of the current round, taking into account the general prioritization and distinction between mandatory and optional tasks as defined by the teacher. Predetermined orders of tasks or dependencies should be explicitly labeled such as 1a, 1b, 1c. Teachers and other students can make suggestions, but it is up to the students on which tasks they individually want to work on. This freedom is supposed to be an empowering experience, especially in school settings that are often shaped by the opposite – the push-principle. The pull-principle of the tasks should encourage self-regulated learning and support differentiating instruction (e.g., by providing tasks with different priorities, “must” vs. “can” requirements, difficulty, or different sets of guiding material). Particularly, optional tasks are supposed to personalize the students’ learning experience in dependence of personal interests and ability. Whereas mandatory tasks are those which ensure that the required underlying objectives are met or artifacts are ready for following tasks/rounds. Sets of tasks for different rounds can be provided in (closed) envelopes.

Closely linked to the pull-principle is the *Commitment* [19] to the chosen task(s). This means that the students take responsibility for the completion of the tasks they chose and work on them until they fulfill the acceptance criteria (i.e., are finished). This does not mean, however, that the students cannot or should not ask the teacher or other students for help. In group work scenarios the selected task resp. the card on the Kanban board is marked with the name of the student who picked it.

Self-organization of the team addresses the agile principle that teams have to power to organize themselves. This is not only an expectation, but also a right to set limits to top-down micromanagement styles. Agile teamwork can be immensely encouraging and motivating [22]. When transferring this principle to school settings, students get an empowering permission to make their own decisions within the setting. This requires, however, certain skills which are also required in classic project work that need to be trained. Students who self-organize their team will need to address the steps necessary to complete a task, such as collecting materials, looking things up, distributing tasks within the group, or simply asking for help/direction. Once students work in this spirit of self-organization and engagement, both the sense of accomplishment and the team-building factor should increase. Still, students should regularly have the possibility to ask for guidance.

Continuous Improvement (*Inspect and Adapt! Fail Fast!* [23]) is the original key concept for the iterative-incremental agile process in contrast to waterfall model approaches. The students are supposed to start with a basic version of products/ solutions and then refine or extend these in following iterations (cf. prototyping). This iterative refinement can be achieved either by repeating the same or similar steps with more insight and lessons learned from previous attempts, taking up the continuous learning principle of the agile manifesto [8]. Hence, students start with smaller, better manageable tasks and can better work towards the goal of the tasks (maybe employing try-and-error strategies) without working towards a wrong direction for a longer time. This strategy is not limited to software prototypes/products but can be applied for a variety of products such as sketches, storyboards, mind maps, lists, experiments, or textual descriptions.

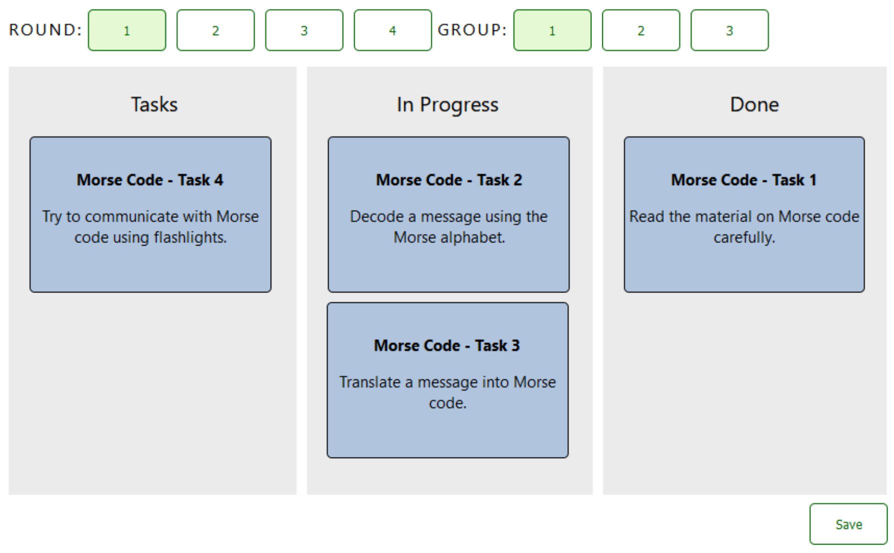
Agile frameworks such as Scrum are structured into (often so-called) sprints. A sprint (iteration or timebox) is a basic unit that represents an agreed fixed period of time between one and four weeks in which the teams work on the product [5]. Agile timeboxes are used for planning sprints and meetings [18]. Such timeboxes are a helpful concept to get things done, to structure the work, and to select adequate tasks. The iterations are called rounds here to provide a more intuitive understanding by students and teachers who are not familiar with agile software engineering. A typical timebox/ iteration can be a lesson or a fraction on a lesson that might be indicated by an alarm clock. Typically, all mandatory tasks of an iteration should be finished by the communicated end of the round and are removed from the board when a new round starts, and the new tasks are placed into the "to do" column. Optional tasks can be selected additionally by the students based on their estimation whether they fit their competencies and available time. Furthermore, timeboxes can also be used for specific tasks. The teacher can indicate a recommended time or even a strict time limit in the task description to further support students in their choice and assessment of their progress by comparing it to the time already spent. After each round there should be a short retrospective (with the teacher) in which the workflow, teamwork and achievements should be reflected to improve future learning (cf. self-regulated learning process [2]).

Based on these seven principle different variants are possible. For example, there could be restrictions of how many tasks a group or single student is allowed to work in parallel. Instead of having a Kanban board per student or group, it is also possible to have a global, shared class board in which each student/group has their "own" row (so-called "swim lanes"). This way also other students can see the progress of the other fellow students. Another alternative is to have a class board and to use it in a more traditional way to distribute tasks to groups working cooperatively on distinct tasks towards the common learning outcomes of the lesson (e.g., to discuss a topic from different viewpoints). Also, the teacher can use a Kanban board to visualize the lesson structure similar to an advance organizer [24]. All topics that are addressed within a lesson are put into the "to do" column and are moved accordingly. It is also possible to decide together with the students based on interests or pre-knowledge which topics should be added to the "to do" column at the beginning of a lesson.

4 A Digital Tool for Supporting Agile Teaching

Using blackboards, sheets of paper, sticky notes and push pins or magnets as a platform for agile teaching is not optimal. Preparing, duplicating, and storing the boards and cards is time-consuming, material-intensive, and error-prone. A digital tool can provide major advantages: Saving of intermediate results, independence of specific (class)rooms, easy preparation, and duplication of boards for multiple groups. Still, the teacher should be there to mentor the students and also to work pedagogically.

In this section the developed prototype is presented. In the current version it supports the basic variant of the agile teaching method discussed in the previous section. It is developed as an interactive web-application based on the JavaScript React framework and can be used without internet connection and without installation with modern web-browsers on various devices. The reasons for this are the oftentimes limited availability of internet in schools and privacy considerations to not store data on external servers. The application as well as the data files can simply be duplicated and stored on USB keys and distributed to students. Only one computer per group or Kanban board is needed. The prototype is open source and can be downloaded as well as the handout on https://www.tel.ifi.lmu.de/software/agileboard4teaching/.



Morse Code - Task 4
Try to communicate with Morse code using flashlights.

Morse Code - Task 2
Decode a message using the Morse alphabet.

Morse Code - Task 3
Translate a message into Morse code.

Morse Code - Task 1
Read the material on Morse code carefully.

Fig. 2. Screenshot of the student view of the digital prototype

The tool is split into two modes of operation: First, an authoring mode for teachers in which they can prepare/edit the initial Kanban boards and tasks for different groups and rounds. The tasks consist of a title, a description, inserted images, links, and attached files, and can be assigned to groups and rounds. The prepared board can then be saved to a file. Second, there is the classroom mode for the students (cf. Fig. 2). Here, the students can load the prepared data and select their group (and the current iteration; see the buttons at the top). Below there are the three columns of the Kanban board and the tasks for the selected group and iteration are displayed. By clicking on the tasks, the students can see the detailed task description and can distribute the tasks among themselves. Moving them to the “right” columns is possible using drag’n’drop. At the bottom of the page there is the option to save the current state of the board to a file for later re-usage. The simple design and minimalistic structure shall ensure an intuitive use and high usability.

5 Lessons Learned and Evaluation of the Prototype

As outlined in the introduction, we applied an iterative Design-Science approach [6] to develop the agile teaching method. During the last five years there were several workshops conducted with training teachers (mentors), teachers and (student) teachers to present and discuss the approach as well as real implementations of the teaching method in different schools by different experienced computer science teachers for different topics. This section presents general lessons learned and recommendations for the teaching approach as well as a more formal usability evaluation of the developed digital prototype.

The initial workshops used wordings and concepts more closely aligned with the Scrum framework. This caused frustration and confusion among the participants and was cited as a barrier to motivation to learn the method. Hence, more natural terms were chosen/developed for key concepts and elements (e.g., rounds vs. sprints, tasks vs. user stories, agile board vs. Kanban board).

Following workshops were held for three hours each and also applied the proposed agile teaching method. The workshops used an agile board, were segmented into rounds between the breaks and provided an overview of being a scrum master. After providing an initial introduction of the teaching method, the participants could pick optional content to be discussed from the backlog. In this way, the method was not only presented, but directly applied by the participants so that they could make their own experiences. Based on the feedback of the participants, the chosen format was suitable for the workshops, however, the presentation of the introduction should be as short as possible and the educational material should be highly simplified, e.g. as a sheet which sums up the central rules which are necessary to get started quickly. As a consequence, the complete introduction on agile methods and Scrum was drastically reduced and a short handout developed. Also, a digital tool for the teaching method was requested. Furthermore, there was much feedback from experienced teachers for guidelines on how to design tasks and additional practical examples.

In sum, the agile method was presented to and discussed with more than 100 teachers and additionally 15 teacher trainers, who train the computer science student teachers in Bavarian schools (Realschule) and around 25 head computer science teachers of a district (Fachbetreuer Gymnasium). The feedback to the agile method was mostly positive or neutral but the teachers often pointed out the long preparation time as a major disadvantage. Again, a digital tool for resolving this issue was requested that respects privacy regulations at schools. The teacher trainers also worked as multipliers.

5.1 Lesson Learned from in-class usage of the teaching method

The proposed agile teaching method could be successfully applied to various teaching settings by different teachers and different classes in real computer science classes. The ages of the involved students range from 11 to 15 years. Two experienced teachers were selected and worked with for a longer period. These two teachers developed different learning settings on the basis of the proposed agile method and were observed during the implementations in the classroom. In addition, three participants of the workshops provided explicit feedback that they applied the agile teaching method in a classroom setting successfully and rated it as useful.

The variant comparable to the advanced organizer [24] was successfully applied quite often and has shown to be a good start for using more agile methods in class.

In general, the students did not need much time for training before they could use this method – just a few minutes of introduction was sufficient. The willingness to play by the rules and to apply the agile method was high in most cases, despite that sporadically students tried to avoid working on the tasks (as in regular teaching situations). However, teachers needed some more training on how to design and use material easily and effectively in the classroom. Each task needs to be specified with enough detail to represent a “self-contained” step contributing to building a final product which is the result of an iteration and should contain activities that can be actively performed. We recommend to design tasks that encourage prototyping and using the iterations to improve done work (i.e., continuous improvement; e.g., for the development of a computer game start with an “ugly” avatar and program the main functionality and later-on design the avatar in more detail). Students then have a visible outcome/result at the end of each round (even if it is not perfect, yet) that also documents their learning. This has shown to be particularly helpful for longer projects. Furthermore, this method simplifies the monitoring of the process for the teacher.

Timeboxes have shown to be particularly helpful because teachers can plan the time similar to a traditional lesson and students get more orientation on how much time to invest in a task (e.g., for designing the avatar). There should also be time limits for choosing the tasks to work on so that not more time is spent on the management than working on the task. Tasks should have a scope of 3 to 15 minutes.

In class the visualization showed to be effective for both, the teacher and the students, to get an overview of the overall progress and also to identify students/groups lagging behind. A public board can be motivating to see other students' progress, as it can be perceived as competition among students. This can, however, also have negative effects such as demotivation for students lagging behind. Here, an alternative might be group/personal boards. The (acceptance test) procedure for moving a task from the “working on” to the “done” column, helped the students to move on to the next task(s) instead of working again on finished ones.

5.2 Evaluation of the Prototype

The digital teaching tool was evaluated for its usability on two levels: First, three experienced teachers (two computer science and one German teacher) were asked in a theoretical work-trough think-aloud study to evaluate the tool from the student perspective whether or not it is suitable in class. Second, the tool was used in a field case study in the classroom in a computer science lesson by one teacher while being observed by a researcher. In both settings semi-structured interviews with the teachers were conducted to collect impressions, feedback, and further suggestions.

During the studies no technical problems occurred and the participants (teachers as well as students) were able to use the prototype in the intended way.

In the first study, the teachers rated the tool as usable for the students for an in-class setting. Two teachers who were already familiar with the agile teaching method mentioned positively the possibilities of saving and archiving the boards as well as transportability, expandability and re-usability compared to the non-digital version. The very limited or yet unsupported possibilities for providing feedback, performing a review, and control on the teacher's end during the process were identified as improvements. Also, the wishes to synchronize the board over several computers (e.g., having a teacher dashboard) and for having an authoring mode was noted.

In the field study, the teacher stated that the tool supported her a lot because the students were able to work independently at their own pace and only asked for help when needed. The students considered the tool to be very intuitive and suitable for the task. Only one group of two had problems integrating into the new situation. Here, a sense of uncertainty could be observed but with a little support of the teacher, this group was also able to work on the tasks. All other students felt very confident in using the tool right from the beginning. The main reactions were positive excitement and interest. After the initial difficulties of the one group had been overcome, concentrated work on the tasks by all students could be observed.

6 Discussion

The development of the teaching method was carried out in the context of computer science and, therefore, mainly computer science teachers were involved. This might look like a major limitation, however, in Germany teachers teach at least two subjects. Hence, the involved teachers could also bring in their experiences from their other subjects. The authors argue that it can certainly be used at least for other STEM subjects or other learning settings outside school (e.g., workshops or vocational trainings) as well to orchestrate tasks and group work.

The proposed approach does not have the goal to teach agile methods as a computer science or software development method but to use it for classroom orchestration. It is, however, likely a proper way to introduce agile methods for computer science lessons and to teach basic agile management competencies.

The evaluation of the prototype with only one class is rather small. The main reason for this is the COVID-19 pandemic as the tool was designed to be used offline in a classroom and not in decentralized scenarios. Offline usability was a hard constraint for compliance with data protection regulations in Germany. Further evaluations are planned. However, this evaluation showed that the prototype fulfills the major requirements and has no significant usability issues.

In the studies, most students were enthusiastic about the agile teaching method and the prototype, but this could be due to the novelty effect. Also, effects of teacher personality are possible. But even when that enthusiasm declines, the benefits of structuring work and making progress visible will remain as the phases of the proposed teaching method of planning, performance and reflection match the three phases of Zimmerman's self-regulated learning process [2] as long as the tasks are prepared sensibly and allow certain choices.

7 Conclusions and Outlook

In this paper an agile teaching method for generic task-driven learning arrangements is presented. The main disadvantage of current approaches is that the teacher does not have an overview of the progress of all students. Agile methods have shown to be very helpful for structuring workflows and providing an overview in software development. The main research questions addressed are how agile principles and methods can be mapped to the school context to orchestrate tasks to support self-regulated learning and how this approach can be supported by technology.

The results indicate that the agile Kanban board and Scrum based teaching method can be used for classroom orchestration and meets the goals for differentiation and providing a quick overview over the status of the tasks of all students. Hence, agile methods cannot only be used in a (software development) project setting, but also for various products that are created by using a certain set of steps. Products can vary from simple task solutions to complex projects ranging from closed tasks such as underlining text, steps to perform in a software, and calculations to open-ended task such as simulations, software products, and essays. The approach can be used in both, collaborative and cooperative learning settings in which students are working together on a product and settings in which all students work on a product on their own.

In general, the proposed agile teaching method can be used without the developed tool, e.g. with a real blackboard and sticky cards. However, the tool allows to save the current state and resume open sessions in other physical rooms, automatically enforce restrictions, measure the time, and to document the results. An extended client-server version of the tool is planned which provides Learning Analytics capabilities such as automatic statistics, identification of hanging tasks, specific recommendations, and a dashboard for the teacher displaying all student boards at once for a quick overview – even in remote settings such as during the COVID pandemic. Further developments could include an integration into existing Learning Management Systems for digitally submitting solutions and providing feedback by the teacher. Recording learning data during use could also increase the value of the tool especially in evaluating teaching lessons and the field of educational research. However, this may only be done in compliance with the data protection guidelines at schools.

In general, agile frameworks such as Scrum may have more to offer to teaching settings. Starting from the presented agile teaching method there are more agile practices to discover (e.g., planning meetings, values, team vision) with students who are used to work with the presented method. Therefore, it might also be a good starting point to introduce more authentic agile project work in computer science.

References

1. Bönsch, M.: Die Differenziertheit der Lernprozesse. Praxis Schule 21(1), 8–12 (2011).
2. Zimmerman, B. J.: Attaining self-regulation: A social cognitive perspective. In: Handbook of self-regulation, pp. 13–41, Academic Press, San Diego, CA, USA (2000).
3. Hattie, J., Timperley, H.: The Power of Feedback. Review of Educational Research 77(1), 81–112 (2007). doi:10.3102/003465430298487
4. Hoda, R., Salleh, N., Grundy, J.: The rise and evolution of agile software development. IEEE software 35(5), 58–63 (2018).
5. Kniberg, H., Skarin, M.: Kanban and Scrum – Making the Most of Both. C4Media Incorporated (2010).
6. Von Alan, R. H., March, S. T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly 28(1), 75–105 (2004).
7. Alavi, H. S., Kaplan, F., Dillenbourg, P.: Distributed awareness for class orchestration. In: EC-TEL 2009, LNCS, vol. 5794, pp. 211–225, Springer, Heidelberg (2009).
8. Beck, K. et al.: Manifesto for Agile Software Development. Agile Alliance (2001).
9. Masood, Z., Hoda, R., Blincoe, K.: Adapting agile practices in university contexts. Journal of Systems and Software 144, 501–510 (2018).
10. Kerber, L., Kastl, P., Romeike, R.: Agiler Informatikunterricht als Anfangsunterricht. In: Proc. INFOS, pp. 211–218, Gesellschaft für Informatik e.V., Bonn (2015).
11. Kastl, P., Romeike, R.: Agile projects to foster cooperative learning in heterogeneous classes. In: Proc. EDUCON, pp. 1182–1191, IEEE (2018).
12. Philippi, D.: Agile Schulentwicklung, https://www.eduagile.de/, last accessed 2022-05-29.
13. Elfreich, H., Strickroth, S.: Bedarfs- und Anforderungsanalyse für ein mobiles, unterrichtsbegleitendes Unterstützungssystem für angehende Lehrpersonen. In Proc. MuC, pp. 210–214, ACM, New York, NY, USA (2021). doi:10.1145/3473856.3474025
14. Strickroth, S.: PLATON: Developing a Graphical Lesson Planning System for Prospective Teachers. Education Sciences 9, 254 (2019). doi:10.3390/educsci9040254
15. Hernández-Leo, D. et al.: COLLAGE: A collaborative Learning Design editor based on patterns. Journal of Educational Technology & Society 9(1), 58–71 (2006).
16. Campbell, C., Cameron, L.: Using Learning Activity Management Systems (LAMS) with pre-service secondary teachers: An authentic task. In: Proc. ascilite, Auckland (2009).
17. Koper, R.: Editorial: Current Research in Learning Design. Journal of Educational Technology & Society. 9, 13–22 (2006).
18. Gloger, B.: Scrum: Produkte zuverlässig und schnell entwickeln. Carl Hanser Verlag GmbH & Co KG, München (2016).
19. Scrum Guide, https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf, last accessed 2022-05-29.
20. Sutherland, J., Sutherland, J. V.: Scrum: the art of doing twice the work in half the time. Currency (2014).
21. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall Upper Saddle River (2002).
22. Whitworth, E., Biddle, R.: Motivation and Cohesion in Agile Teams. In: XPU 2007, LCNS, vol. 4536, pp. 62–69. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73101-6_9
23. Schwaber, K., Sutherland, J.: Software in 30 days: How agile managers beat the odds, delight their customers, and leave competitors in the dust. John Wiley & Sons (2012).
24. Ausubel, D.: The use of advance organizers in the learning and retention of meaningful verbal material. Journal of Educational Psychology 51, 267–272 (1960).