Agile Methods as a Methodology for Implementing Collaborative Constructionist Learning in Computer Science Education and Beyond

Peter Brichzin, Erasmus-Grasser-Gymnasium München, schule@brichzin.de
Petra Kastl, Jack-Steinberger-Gymnasium, kastl.petra@jack-steinberger-gymnasium.de
Ralf Romeike, Freie Universität Berlin, ralf.romeike@fu-berlin.de

Abstract

In start-ups today, almost nothing works without agile software development. In schools, agile methods may be used to implement constructionist learning by helping the students to collaboratively create meaningful artifacts. This paper reports on how agile methods can be used as a methodology for conducting school software projects and evaluates, in the light of Constructionism, how collaborative learning in school projects can be supported.

Introduction

Constructionist learning approaches share the idea that learning "happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" (Papert and Harel, 1991). In computer science education, the idea of giving students the opportunity to create meaningful software artifacts has, over time, paved the way for learning for all. Since software is usually created in collaborative projects, school software projects are typically organized in a similar way as they are in professional settings. Traditionally, school projects, similar to professional projects, proceed in sequential phases (Frey, 1983). At the beginning, the task is analyzed. From there, the requirements are defined, followed by a design phase. Only then does the implementation take place, which is tested against the requirements at the end. In school projects, however, sequentially organized projects often turn out to be difficult, since the students initially usually do not have the necessary experience and "soft skills", which are prerequisites for successful project work (Meerbaum-Salant and Hazzan, 2010). The constructionist learning theory can help us to understand the issues students have with a linear approach. If constructionist learning involves going through iterative cycles of step-by-step creation and understanding an artifact, it seems obvious to use an approach for school software in which students develop increasingly complex prototypes in order to not only create, but also to establish a better understanding of the artifact created, including the process of doing so.

Agile methods have spread rapidly throughout work environments since the 2000s, functioning as an alternative to sequential approaches in both small and large companies. Agile teams are often particularly motivated, work in a focused manner, treat each other with respect, and see mistakes as an important opportunity to grow. They determine their own path to reach their goal, reflect on it regularly and see change as an opportunity. By doing so, agile thinking and action change the corporate culture, as they are associated with a series of central values, such as open communication at eye level and self-responsibility in the team. Thus, teams and customers who have had a taste of agile team work usually do not want to go back. In agile projects, teams develop prototypes iteratively and incrementally. They test and review their interim results and regularly obtain feedback. Concrete practices such as user stories, project boards, stand-up

meetings, and team retrospectives support the teams in project organization and execution. These methods can also enrich teaching (Romeike and Göttel, 2012). Both for teachers and students, agile approaches almost always quickly produce many positive and motivating effects, because students usually succeed a lot better in organizing themselves, creating great results in joint responsibility, and achieving pleasing professional and social learning successes (Kastl and Romeike, 2018). This greatly underlines the social dimension of learning: "What is created and learned in the construction process is greatly affected by who we build with, and for whom we build" (Holbert, Berland, Kafai, 2021). However, since "creating communities around creativity and technology is hard, and success is not ensured" (ibid.), strategies are needed to support students in learning to collaborate in developing an artifact and to organize the process in a meaningful way. We will provide an overview of the agile process and explain some practices that are recommended for getting started with agile project work (Brichzin, Kastl, Romeike, 2019) in more detail in the following, since such agile practices can support constructionist learning (Kastl, Kiesmüller and Romeike, 2016; Meerbaum-Salant and Hazzan, 2010; Monga et al., 2018).

Agile Practices for Structuring and Supporting the Constructionist Learning

Agile practices help teams to act on values that are appreciated in constructionist learning as well. Core agile values include communication and simplicity, but also transparency, self-organization, and feedback. In addition, focus (devoting your undivided attention to a specific task), courage or commitment (dedicating yourself to your team to fulfill the mutually set goals) can also be emphasized.

Constructionist Learning in an Iterative Process

Instead of running through the project phases only once in a sequential manner, agile projects process the phases cyclically in so-called iterations or sprints (cp. fig. 1) using short, fixed time windows. The iterations follow one another directly, are all of the same length, and are each concluded with a functioning, incrementally growing interim result. Clear communication structures and visualization are also anchored in the process to ensure transparency. The iterative approach allows learners to grow with the project and gain confidence in project execution. Experience shows that the first iteration may be experienced as chaotic by students and teachers alike. However, the structure and practices help students learn self-organization and collaboration. The first successes are quickly visible, motivate the students and help them to regularly review the objectives. The teacher can now provide regular feedback and, if desired, integrate new learning content into the project work flow step by step to match the product development.

User Stories – Scaffolding Construction

User stories describe software requirements from the customer's point of view. They consist of a few sentences and are formulated in everyday language. They help the students to divide the extensive technical functionalities of the overall system into easily manageable parts and thus give the software development process a clear structure. Even students with a weaker technical background can contribute to this process. In this way, user stories become the basis for communication about the sub-goals, teachers can discuss them with the teams and, if necessary, can intervene at an early stage. User stories support learning in the sense of scaffolded construction.

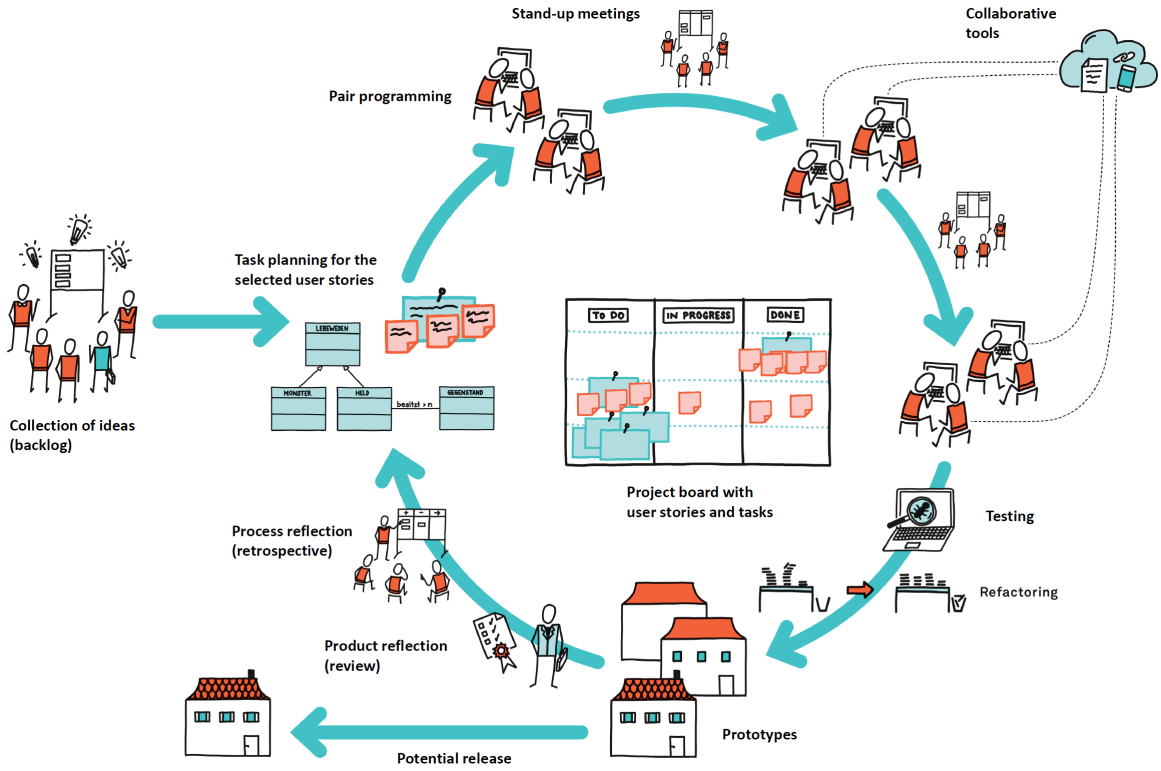

Fig. 1. A typical agile process adopted for school.

Project Board – Managing Collaboration

The project board (cp. fig. 1, center) not only visualizes the goals and task packages, but also illustrates the current work status (and thus the project progress) as well as the work distribution. The visualization provides transparency, supports self-organization, and illustrates the team's (and the individual team members') commitment. It also demands simplicity in planning and ensures focused work. Along with the prototypes, it is a basis for feedback discussions. Agile project participants report: "We're working on fewer things now, but they'll get done," and "You always know right away where you stand and what's next. You don't have to have so much on your mind and don't get bogged down so easily."

In the simplest case, the project board consists of three columns: "To Dos", "In Progress" and "Done". On the left, in the "To Do" column, the user stories (index cards) hang in descending order of priority, alongside the sub-sequent tasks determined (sticky notes), in case the user story has already been specified. During the work phase, each team member or pair selects a task, writes his name on it and places it in the middle column titled "In Progress". After completing it, it can then be moved to the column at the right titled "Done". In this way, you can see at a glance who is currently responsible for which work package and which tasks and user stories have already been completed. The active reassignment of completed tasks motivates the students, is an occasion to celebrate what has been achieved and provides the teachers with a basis for assistance.

Stand-Up Meetings – Encouraging Communication

Stand-up meetings help the group to organize itself. Like all other meeting forms, a stand-up meeting relies on openness, respect, and the courage to address, for example, mistakes or poor work performance, in addition to focus. It encourages and requires communication, brings the team to a common level of information, and gives the teacher insight into the teamwork taking place. The daily stand-up meeting is held by the teams at the beginning of each lesson at school. It replaces the usual in-class recap. Team members take turns answering three questions: "What tasks have I completed since the last meeting? What tasks do I want to work on next? Were there any problems and with what?" To ensure that only the essential information is exchanged, the meeting is held standing up and in front of the project board. Problems are only named and, if necessary, help is requested. Consequently, standup-meetings help regularly structuring the social dimension of learning.

Pair Programming – Structured Working in Pairs

Structured partner work helps students to organize collaborative work without active-passive division, to support each other and to exchange knowledge and information about concrete implementations. The practice thus ensures transparency, immediate feedback, planned and focused action, and simple solutions. Hence, it is a good method to implement structured partner work in computer science classes. The partners take on defined roles that are regularly exchanged. The driver works on the task and informs the navigator about his intentions and his approach. The navigator checks the processing, considers whether there are alternative, simpler ways of solving the problem, makes sure that the driver stays with the actual task, and addresses possible misinterpretations. In school, partner work increases self-confidence, demands a description of the procedure, initiates discussions about approaches and solution strategies, prevents mistakes and thus supports the learning process.

Reflection – Understanding Learning Progress Through Review and Retrospective

Even though it is considered an important aspect of teaching and learning processes, reflection often falls by the wayside due to time constraints. Also, the learning impulses may peter out due to the following project taking place too late for applying what has been learned. Here, the iterative approach is a clear plus. Regularly pausing after each work phase to get feedback on the product in a so-called review moves the team forward, but it also has to be learned. By comparing the goals set with what has been achieved, it promotes self-regulation skills. At the same time, it is an occasion to celebrate what has been achieved. Possible problems in the work process or in the team can be openly addressed and tackled in a retrospective. This requires courage, respect, and openness. The recurring opportunity to practice this supports the development and strengthening of team skills and prevents frustration from building up.

Discussion

Even though computer science education nowadays profits from the ideas and tools of constructionist learning, the constructionist roots were often not sufficiently taken into account while developing into a more mature subject. Instead, methods gained from practice have been applied which do not always serve the pedagogical purpose. With agile methods, professional practice and constructionist learning find themselves together again. Not only do they match the iterative character of the constructionist circle of creating an artifact and enhancing the understanding of

the artifact, they match the process of creating such an artifact as well. Furthermore, agile practices can help support the learning process by structuring, scaffolding, and providing transparency. Experience shows that students who have gained experience with agile projects in computer science lessons also use their competencies for work and self-organization in other subjects. Thus, the entire school can benefit from such collaborative constructionist learning with agile methods.

References

Brichzin, P., Kastl, P., and Romeike, R. (2019) *Agile Schule. Methoden für den Projektunterricht in der Informatik und darüber hinaus* (Agile school. Methods for project-based learning in computer science and beyond). Bern: hep Verlag.

Frey, K. (1983) *Die sieben Komponenten der Projektmethode – mit Beispielen aus dem Schulfach Informatik* (The seven components of the project method – with examples from the school subject of computer science). LOG IN, 3(2), 16–20.

Holbert, N., Berland, M., and Kafai, Y. B. (2021) *Introduction: fifty years of constructionism*. Designing constructionist futures, 1-20.

Kastl, P., Kiesmüller, U., and Romeike, R. (2016) *Starting out with projects – Experiences with agile software development in high schools*. In ACM International Conference Proceeding Series (Vol. 13-15).

Kastl, P. and Romeike, R. (2018) *Agile projects to foster cooperative learning in heterogeneous classes*. In IEEE Global Engineering Education Conference, EDUCON (1182-1191.

Meerbaum-Salant, O. and Hazzan, O. (2010) *An Agile Constructionist Mentoring Methodology for Software Projects in the High School*. ACM Transactions on Computing Education, 9(4), 1–29.

Monga, M., Lodi, M., Malchiodi, D., Morpurgo, A., and Spieler, B. (2018) *Learning to program in a constructionist way*. In Proceedings of Constructionism 2018. Vilnius, Lithuania.

Romeike, R. and Göttel, T. (2012) *Agile Projects in High School Computing Education – Emphasizing a Learners' Perspective*. In Proceedings of the 7th WiPSCE'12 (pp. 48–57). ACM New York, NY, USA.

Papert, S. and Harel I. (1991) *Situating Constructionism*. Ablex Publishing Corporation.