Starting out with Projects - Experiences with Agile Software Development in High Schools

Petra Kastl
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen
petra.kastl@fau.de

Ulrich Kiesmüller
Simon-Marius-Gymnasium
Simon-Marius-Straße 3, 91710 Gunzenhausen
kiesmueller@simon-marius-gymnasium.de

Ralf Romeike
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen
ralf.romeike@fau.de

ABSTRACT
School software projects, as they are common e.g. in German CS classes, traditionally apply inflexible process models, mostly an adapted waterfall model. Typically, such projects are conducted at the end of a school year. In this paper we pursue the question, if and how changing process model and time may help bringing the advantages of project based learning into play. We describe and compare practical experiences of a study with 140 students, considering four different contexts. By applying agile methods, flexibility was gained. The evaluation of the different implementations results in a more holistic and comprehensive view of projects in CSE.

CCS Concepts
• *Social and professional topics~Software engineering education* • *Social and professional topics~K-12 education*

Keywords
Agile methods; Projects in computer science education; Secondary computer science education.

1. INTRODUCTION
With the growing importance of computer science (CS) in schools around the world, methods and strategies for teaching CS in the classroom are being discussed increasingly. Even though there is mutual agreement that classroom software projects play a crucial role for teaching and facilitating an appropriate and attractive notion of CS, conducting software projects in schools remains a challenge: In Germany, projects typically are conducted at the end of a school year, when the students have acquired the competencies necessary for working on their project independently. This comes with the drawback that the pedagogical advantages of projects did not come into play until then. Also, at the end of the school year, time issues may force the teacher to shorten the project or lead to projects remaining unfinished. Additionally, traditional methods for school software projects suggest applying quite inflexible processes such as the waterfall model. However, in school flexibility is important in order to cope with the heterogeneity of contexts and the objectives of projects [11]. In this paper we describe lessons learned from four case studies with 140 students in total, from three different federal states of Germany, with the aim of addressing the aforementioned issues.

WiPSCE '16, October 13 - 15, 2016, Münster, Germany

ACM 978-1-4503-4223-0/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2978249.2978257

The interventions we describe and reflect on are designed and implemented within a design-based research process aiming to refine the theory of agile projects in CSE, which is outlined in [8]. A central aspect of the process is the close collaboration with teachers.

They contribute their practical expertise, i.e. their pedagogical content knowledge (PCK), by adapting the agile framework to the individual needs of their context. All teachers participated in the project because they saw room for improvement in their so-far waterfall-model based software projects. The problems encountered by the teachers are discussed in [8]. The analysis of their observations and experiences may help to understand if, how, when and why agile projects can support the objectives of project-based learning (PBL) better. Based on a qualitative analysis of the material of the initial iteration of the design-based research process, preliminary findings support the assumption that the emphasis on self-organized and outcome oriented effort, as well as communication, cooperative work and learning supports students' aim to acquire, apply and enhance a variety of subject-related, methodological and social competencies [8]. This practical report will especially outline the experiences providing answers to the questions why a flexible process is important as well as what the effects of conducting projects early in the learning process are. In addition, the design decisions and the evaluation and reflection of the different implementations will be analyzed with respect to the concept and attributed value of agile school software projects by teachers and learners.

In the next section, we outline and discuss traditional arguments and objectives of projects in CSE which are relevant to the design of the intervention. After that we describe the design and the course of a project which is conducted with programming novices and lasts eight months. The intervention facilitates a steady and well observable development of professional, social and organizational skills during the project. In section 4 we supplement this approach with further examples with slightly different objectives concerning programming novices and agile projects. Finally, we reflect and discuss the different approaches as well as the practical experiences.

2. PROJECTS IN CS EDUCATION
Projects traditionally are considered relevant in CSE, typically relying on professional software development practice [6, 7, 11]. However, projects in CSE are also challenging, as they require a variety of professional skills on many different levels. Therefore, they customarily are scheduled late in the learning process, when the students are expected to be confident in necessary skills such as programming. The objectives of said projects mainly focus on the application of existing professional skills: Students use analysis, problem-solving, design and implementation techniques. They

carefully plan their teamwork and they structure their course of action based on their professional knowledge by applying a well-established process model. The most common model used in Germany is an adapted waterfall model as outlined by the Swiss educationalist Karl Frey [6]. In addition to pedagogical project objectives, students are expected to gain insight into aspects central to the creation of complex software systems. With respect to problems that teachers and students experience with such linear process models in high school projects, Romeike and Göttel [13] proposed an Agile Model for Projects in Computing Education (AMoPCE) including adapted agile practices and artefacts (see fig. 1). For the design, the authors discussed agile artefacts and practices from a pedagogical perspective.

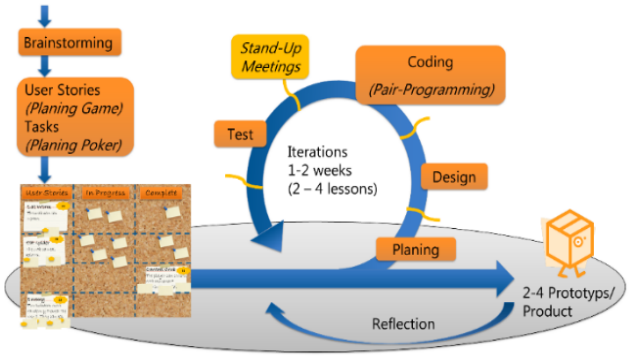

Figure 1. Model for agile projects in CSE.

Similarly, more recent curricula and publications also no longer recommend traditional linear approaches for projects in high schools. Rather, they suggest computer program design as an iterative and collaborative process [1, 2]. Students are supposed to solicit, evaluate and integrate peer feedback to develop or refine solutions and use techniques like pair programming and code reuse [1, 7, 11]. Also, pedagogical objectives, e. g. the enhancement of soft skills, especially communication, self-organization and the ability to work in teams, are emphasized [1, 3].

In the context of empirical software engineering there are two discussions we consider relevant for our work. Firstly, based on practical experience from an industry perspective agile practices and artefacts are assessed in order to identify those beneficial for project success [12]. Secondly, studies analyze the experiences in order to identify aspects which are relevant for the design of process adaptations [4], as there is general agreement that there is not the one framework that fits all projects. Also, best practice examples are considered to be important.

By now there is also a great number of publications discussing theoretical and practical aspects of agile approaches in the context of educational software development projects [5, 7, 10]. Studies refer to the traditional capstone projects as well as to projects earlier in the learning process applying different agile approaches and aiming for different skills. They mainly report positive experiences regarding the students' attitude and learning progress and varying results concerning experiences with different agile practices. However, these studies are conducted from an academic point of view focusing on still different contexts and goals than projects in secondary CS education. We are not aware of relevant publications that emphasize aspects and characteristics of agile approaches and PBL in the heterogeneity of CS high school projects.

In all interventions described here, projects are conducted in regular lessons with programming novices early in the year. However, the projects' individual designs vary considerably. They focus on different competences and on different objectives, as we will explain in the next section.

3. PROJECTS EARLY IN THE LEARNING PROCESS – FOUR CASE STUDIES

3.1 Project Set-Up and Data Collection

In the following chapters we report on the practical experiences and findings of four different experiments exploring the application of project-based learning early in the learning process. The interventions have been put into practice for up to three years by now. The teachers explained their observations and experiences in semi-structured interviews [cf. 8]. Further material is used for the evaluation in section 4.2. Photos of the project boards document the progress of the projects. Also they document the learning progress as the tasks gradually become more complex. Moreover, there is the code which shows the programming concepts and structures applied by the students. Finally, there is data of the students who elected CSE as one of their next year's subjects from the last five years.

Classes consist of about 20 to 32 students between 14 and 18 years old and there is at least one lesson (90 minutes) per week. The students' prior knowledge concerning objects, algorithms or programming is minor to non-existent.

3.2 Eight-Month PBL with Agile Methods

In this class students learn the basics of static and dynamic object-oriented modelling and programming with Java by using BlueJ. The motivation of the teacher to change the teaching methodology arose from the experience that most students were not able to create an interesting and exciting product when the project was conducted at the end of the first learning year due to limited time. Also, soft skills and organizational skills could not be fostered sufficiently in such a short period of time, even though students will urgently need these skills in their future qualification period. Therefore, the teacher extended the project duration, still applying the waterfall model. However, at this early point the students' professional knowledge was far too poor to self-organize the planning of their project. They now strongly depended on the teacher's support, which led to an exhausting situation for the teacher and was discouraging for the students, who complained about long waiting times. Moreover, often the input of new theoretical knowledge did not meet the actual needs of many students. Thus, his motivation to apply agile methods was to provide students with a structure that enables them to self-organize their learning process as well as their project-work. In the following paragraphs we will outline key teaching practices that allow the teacher to achieve this goal.

Teaching practice: Preparing the project

Before the full project started, the students worked on small problems in micro-projects. These problems were designed in such a way that the user stories[1], tasks[2] and fragments of code could be reused in the later projects with only little modifications. Also, other essential practices and techniques of agile project management were introduced.

The problems, which in the preparation phase were given by the teacher in the form of user-stories, gradually became more "open".

[1] A user Story describes a small functionality from the customers' point of view.

[2] Tasks describe what needs to be done to realize a user story from the developer's perspective.

This was to make sure that students slowly get familiar with the frequently necessary change of perspective. For example, a user-story contained the following ambiguous task: "Draw a circle that is moving in a circle". A look at the teams' different interpretations and implementations offered interesting points for discussion and reflection (see fig. 2). There were simple solutions, which could be implemented quickly, and more complex ones, which took time and required more knowledge. Later, in the full project the students will be their own "customers" and will plan and prioritize features. Therefore, they learned about aspects of time and knowledge and can use these arguments for better decision making later on.

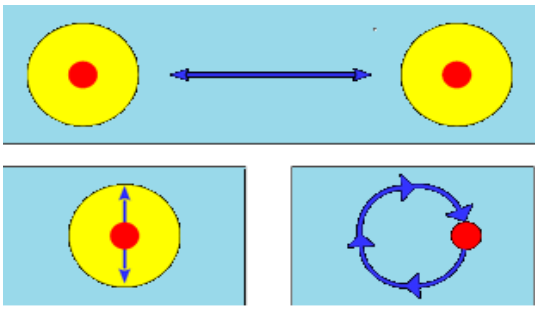

Figure 2. Implementations of "A circle moving in a circle". Arrows are added to illustrate the movement.

After about ten weeks the students were able to create simple UML class-diagrams and implement them, and they also used control structures in their Java code. They were familiar with the tools and some agile practices and artefacts. In the following eight months the students worked on agile projects and self-organized their work, as well as their learning processes, to a great extent.

Teaching practice: Building collaborative teams

There is general agreement that a person's performance and motivation is best, if he/she is interested in the topic. However, in school projects it is typical for students to join a team because their best friends are there, rather than out of interest. Here, the teacher asked the students to vote for their favorite topic. However, while they were voting they did not know that the team building will be based on the voting. This procedure resulted in teams which were mostly not the usual, well-established ones. Consequently, students had to find ways of cooperating and communicating with "new" partners.

Teaching practice: Pedagogical support and delegating responsibility

Agile projects contain playful practices for determining the project goals and requirements. However, due to limited experiences in software development at this early learning phase the students were not confident in determining achievable goals all by themselves yet. This problem was addressed by customer meetings with the teams to support them in the elaboration and specification of goals of a first stage. Similar to professional practice, the students had to negotiate with the customer (teacher), who could pedagogically influence the project by keeping the context authentic. As soon as possible, the projects' control was shifted to the teams, i. e. students became their own customers as soon as possible.

Teaching practice: Iterative project setup

According to the agile model, with each iteration students now repeatedly apply the sequential process of planning, implementation and testing. However, at the beginning, the students were neither good programmers nor did they know what to agree on in a collaborative work. This challenge was met by planning small tasks for user stories, which last for a work period of approximately 20 minutes, implementing the tasks collaboratively and meeting again for planning. Finally, teams integrated and tested their code so that they always had a working prototype at the end. As code integration was not tool-supported, this was done only at the end of each lesson in order to save time.

Usually, an iteration may not be interrupted in order to create an effective workflow: If problems occur, they should be solved in the next iteration. Here, an iteration could also be interrupted, if a problem occurred which could be solved by the team only. This facilitates students' learning by solving problems, which was more important than effective product creation. However, experience also showed that not interrupting the teams works quite well after some weeks, as students rapidly learned to plan in a goal-oriented way.

Teaching practice: New content

Starting with project-based learning early implies that large parts of the curriculum need to be covered within the project. It remains a challenge to provide such learning opportunities to students without interrupting them. Experience showed that the teams—even though they performed quite differently—often had to solve similar problems nearly at the same time. For example, they all needed one-dimensional arrays at a certain point. Then a central input phase providing theory and code fragments was scheduled at the beginning of the next lesson. If a team needed input much earlier, it was provided with individual help or asked to implement other user-stories first. If students did not come across obligatory content by themselves, the teacher, in the role of a customer, asked for certain features that would require the application of the concept in focus.

In the first year it turned out that the coordination and planning of the input-provision was not transparent enough. Therefore, the field "problems" was introduced at the project board and the students were encouraged to write down their problems and questions on sticky notes (see fig. 3). If it is obvious that the programming pair or the team can solve the problem without support, the note is reassigned to the column "in plan" and a hint is given, if necessary. If it is obvious that other teams will have similar problems soon, the team is asked to postpone the affected user-story for a certain time and a central input phase is scheduled. Otherwise, the individual pair or team is provided with input. This approach facilitated the coordination of input provision in the following years.

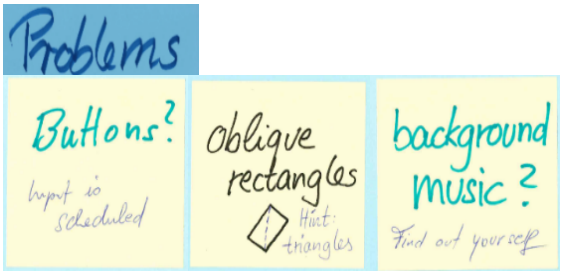

Figure 3. Typical problems and how they were handled.

3.3 Different Learners, Different Teaching Practices, but the Same Framework

The following three projects are also based on the same model (AMoPCE). All four teachers participated in the same initial workshop. Also, they all conducted their project with programming novices early in the year. However, by adapting the model according to their students' individual needs and their individual context they designed rather different interventions. In order to contrast and learn from the practical experience, the most interesting approaches are described in the following paragraphs.

Project 2: A "scripted project" was put into practice as an alternative approach to the project's preparation we described above. It combines self-organized learning with the agile framework. Students become familiar with agile artefacts like project boards, user-stories, tasks and prototypes, with practices like stand-up meetings, pair-programming[3] and even planning poker on the fly.

The teacher planned four iterations, each lasting two weeks (with 90 minutes plus 45 minutes per week). Each iteration had three stages: learning new content and doing some exercises, applying the knowledge using a Processing programming environment to build an increment to the teams' prototype, and, finally, showing and reflecting the products in plenum.

In stage one, programming pairs self-organized their work on so called "student-stories" and tasks, which are actually learning assignments designed by the teacher. In stage two, the teams worked collaboratively on ready-made user-stories and tasks which corresponded to the newly learned skills but also allowed individual creative interpretations to some extent. Finally, in stage three, there was peer feedback and feedback from the teacher. The teacher especially valued the regular reflection and feedback meetings. He observed that the frequent feedback resulted in a steady growth of students' self-efficacy.

Project 3: In this short-time project, lasting for four weeks (90 minutes plus 45 minutes a week) only, CSE novices gained practical experience with agile projects early in the year by using Scratch. The emphasis of the project was less on the students' progress in their programming skills, but more on their organizational skills. The task was to collaboratively develop increasingly elaborate prototypes of a simple game by organizing the team work in a way similar to professional agile projects. It was communicated to the students in advance that they would not be able to implement all their ideas. However, they would know how things worked and how they would have to continue. In their reflection the students mentioned that they learned how to collaborate in software development. They also stated that they experienced not only the necessity of communication and the difficulty of creating a common understanding, but also the highly motivating character of collaborative work.

Project 4: The idea of this field-tested approach is to include "scientific" aspects into the project-work even if the students are programming novices. Students did not focus on the software product or the enhancement of programming skills, but rather on the evaluation and reflection of the software development process itself. I. e., the students applied an agile framework and evaluated and reflected the method.

4. FINDINGS

Can one methodological framework successfully be of use in such a variety of very divergent learning settings? Applying a methodological framework out of the box, such as the one used for agile projects, one might expect similar outcomes even within different scenarios. Analysis of the data (cf. section 3) reveals that the teachers adapted the framework to a substantial degree in accordance with their pedagogical goals and the learners' needs. How differently the challenges of agile projects were addressed in the different settings will be discussed exemplarily in section 4.1. Experiences that occurred in a similar way in all projects will be elaborated with respect to project 1 in section 4.2. The solutions illustrate a lot of creativity and pedagogical content knowledge and may serve as an example for good teaching practice.

4.1 Variations of Agile Practices in Different Settings

Iterations for programming novices

Teachers used two different strategies for handling an iteration, depending on which one fits their context best. In one approach the iteration gets some form of even smaller sub-iteration, while in the other approach the number of challenging tasks is reduced. As we outlined in section 3.2, in project 1 the course of an iteration was adapted to the learners' skills (planning of only small tasks) and the projects objectives (enabling learning by problem solving). Additionally, further context parameters were taken into account, like used tools (no tool-supported integration) or the used programming language. In project 3 the iteration's course went according to the agile project model with a duration time for iterations of 90 minutes. However, the number of challenging tasks was reduced: firstly, students did not have to change perspectives. Both, customers' and developers' perspective, were almost identical for the implementation with Scratch due to the small user-stories. Secondly, user-stories could be pulled or postponed in order to adapt the workload of an iteration. Thus time management became rather easy.

Emphasize and structure communication

The model for agile school projects includes various practices to structure and foster communication, like the stand-up meeting in front of the project board. Students meet at the beginning of each lesson in order to recap last week's lesson.

This practice is valued and implemented by all teachers. However, teachers also added further similarly organized meetings in front of the project board. In project 1, customer-meetings and problem-solving meetings were introduced, as outlined above. Furthermore, all teachers included planning-meetings at the project board. There were product-reflections in project 2 and process-reflections in project 4. These meetings are all known in professional projects and each meeting has a different structure there. However, teachers all adapted the simple but effective structure of the stand-up meeting to most of their added meetings. The teams' project board became the typical place for any team interaction.

Handle practices flexibly and optimize the process

Comparing the projects also shows that teachers handled the agile practices differently, depending on their students' reaction and the skills they wanted to foster. This flexible handling also offered the teams the opportunity to optimize their process based on reflected experiences and personal interests (as it is done in professional projects).

For example, all teachers introduced planning poker[3] for effort estimation. In project 1, the practice was introduced after about two months, because the students then approximately have the experience they need. However, even though some user-stories were huge in the beginning, at that point the user-stories had all become quite small and in average equally sized. Planning poker turned out to be unnecessary and most teams decided to continue estimating the effort based on the number of user stories (a fact experienced by pro-

[3] A gamified technique; team members estimate by playing numbered cards face-down, then cards are revealed. If necessary, the estimates are discussed, finally there must be a consensus.

fessional developers, too). In project 2, students also became familiar with the practice of planning poker. However, since the student- and user-stories for each iteration were well planned by the teacher, taking into account slow as well as fast learners, students realized that effort estimation offers no additional benefit and therefore skipped it completely. In project 3, the students used the categories small, medium, and large to estimate the effort. This helped the students to plan their iteration without long discussions. Also, as they pulled or postponed user-stories, there was no urgent need to reflect the estimation in order to explicitly foster the skill. In project 4 however, the participating students were high performers with excellent analytical skills and quick perception. These students were fascinated by planning poker. They reflected their previous estimation and used the gained knowledge in the next iteration, comparing new tasks to already implemented ones. They also reported that planning poker helped them to achieve a common understanding in planning, as different estimates often result from a different understanding. Moreover, it supported their passive learning and fostered the ability to explain one's arguments.

Flexible choice of roles taken by the students

All teachers value the aspect that students have to consider situations from several perspectives, including not only technical details, but also customer needs and their vision of the product. Nevertheless, they also report in unison that this is really challenging for their students. They developed different approaches to cope with this challenge.

In agile projects, user-stories describe a small functionality from the customers' point of view and tasks describe what needs to be done to realize a user story from the developers' perspective. In order to plan tasks, the students have to change perspectives. In project 1 this skill is fostered starting in the preparation phase and students keep practicing it throughout the year. In a little while, students spontaneously and alternately take the customer role, starting sentences with "As a customer, I…". In the beginning, students frequently call spontaneous problem-solving meetings and learn by experience how to plan effectively and how to change perspectives. Discussions during the product test show that they also gradually understand the different tasks of a developer (planning, designing, coding), as they identify the origin of the detected problem. In project 2, the teacher provided user-stories and tasks. The students are developers, but they have to design a user manual too. Therefore, they change between the developer perspective, which is new to them, and the user perspective, which is quite familiar. This change of perspectives does not require programming skills and thus fits programming novices well. In project 3, the gap between the customer's perspective and the developer's perspective is reduced, as described above. In project 4, students reflect by themselves that the planning of tasks for the corresponding user stories requires a change of perspectives. Also, they experienced this as difficult for several reasons.

4.2 Common Experiences with Agile Practices

Findings with respect to the goals and challenges that are shared by all projects will be presented exemplarily for project 1.

4.2.1 Findings concerning the Students Learning Outcome

The overall objective of project 1 is that students learn the basics of object-oriented modelling and programming according to the curriculum. Based on the collected data it is outlined that the agile project successfully supports this goal and that slow learners are keeping up.

Programming: The pictures of the project boards and the code show that students now master more complex tasks than students did in the years before. Also, students work more decidedly and ask much less for support. Rather students work out the solution of complex tasks by themselves and thereby reuse and/or adapt code fragments they found in tutorials (see fig. 4 and 5). For example, creating one's own colors with BlueJ typically is a complex task. Some students want to create their own colors at all costs, as the given Java class comes with a fix set of colors addressed by numbers 0 to 7. Each year a team manages this task using a Java manual and when they succeed there is a shout "Yes, yepee! We did it!" and other students gather and cheer and the solution is shared. Reading the students' code also shows that they use more structures and concepts than the students did in the years before and more than the curriculum includes. I. e. concerning students' programming skills, the outcome meets the objectives.

Figure 4. Create complex shapes and use own colors.

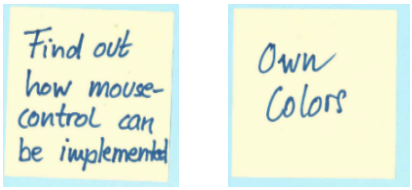

Figure 5. Examples of more complex user-stories.

Modelling: As the teacher reports, in the traditional, non-agile projects, object-oriented modelling was difficult to motivate. Furthermore, for most students its benefit remained purely theoretical. Now, students voluntarily started to draw and update class diagrams after some weeks and refer to them when they planned new user-stories or discussed possible solutions to a problem. Additionally, the teacher could observe how teams discussed the communication between objects and hence made use of dynamic modelling techniques, i.e. sequence diagrams, even before they were officially introduced, in order to better organize their planning activities. Hence, the wish of using this modeling technique arose out of the students' experiences and problems –there is probably no better way of learning what modelling is about. In comparison to regular teaching, in total there was less time spent on the elaboration of models, but students acquired a deeper understanding.

Slow learners: The teacher observed that slower learners often have problems, e.g. writing their code according to the team's model in the beginning. Since the teacher avoids intervening, eventually the students will require help with their work. Then, the teacher suggests and guides a refactoring of the code. This enables students to understand that some minor modifications may be sufficient for fixing the code. Also, this provides the slow learners with the positive experience that they too contribute to the team's product. Observing the performance of these slow learners in the course of the project shows that refactoring is an encouraging experience to most of them strengthening their self-confidence and maintaining their motivation.

4.2.2 Findings Concerning Students Attitude Towards CS

The percentage of students who elect CSE as one of their next year's subjects can be used as an indicator for students' attitude towards CSE. Due to the structure of the Bavarian High School system it is generally difficult to actually win students for CS courses. Consequently, the 11th grade and 12th grade are often taught conjointly to reach the number of students necessary for a course.

In order to outline the impact of the agile project 1 on the students' choice of subject, we describe the situation in the years 2012 to 2016. In 2012 and 2013, a waterfall model was applied in the eight-month project. However, conducting the project early during the school year had no effect. Still, as in the years before only one or two students per year, i.e. 2%, chose CS as subject. In 2014 the teacher changed over to the agile procedure which is described in project 1. The percentage of students who were interested in continuing their CS education rose to over 15% in this year and has stayed at a high level since then.

If and in what way this observation is reproducible, and also how it is connected to the introduction of agile methods, future research must show. However, similar effects are observed and reported by other teachers. All of them point out that agile projects motivate and inspire their students. Therefore, they welcome the opportunity to conduct projects with programming novices early in the year.

5. DISCUSSION

Projects can be conducted in various contexts. In place of the traditional project, scheduled at the end of a school year, teachers successfully conducted projects early in the school year, inspiring and motivating their students.

Projects can also be actively designed and shaped. In place of the traditional project, focusing on the application of existing professional skills, teachers successfully designed a variety of projects with different foci, emphasizing different pedagogical objectives and aiming for different professional goals.

For many teachers who participate in the study the idea of flexibly adapting a methodological framework was uncommon. This is in line with literature about project based learning, which yet often fails to stress the possibilities and need of individual methodological adaptations, based on the PCK and objectives of the teachers. Based on the experiences we learned that students, as well as teachers, benefit if a methodological framework is considered more as a choice of good practices than a fixed process model. In this paper we outlined several examples of how such a flexible handling can work in regular school teaching.

Summing up, the teachers describe their experiences as follows: Firstly, one can make students enthusiastic about CS as a subject by conducting agile projects, and that already early on during the learning process. Secondly, with those projects one can foster the development of abilities central to being a good software developer [9]. Also, depending on the design of the project, individual competences can be especially emphasized and facilitated: continuous learning (project 1), understanding and implementing feedback (project 2) as well as organizing teamwork (project 3) and reflection (project 4).

6. REFERENCES

[1] ACM Computer Science Teachers Association. 2016. DRAFT 2 of the CSTA K-12 CS Revised Standards. http://www.csteachers.org/page/SubmitYourFeedback. Accessed July 2016.

[2] Bell, T., Andreae, P., and Lambert, L. 2010. Computer Science in New Zealand High Schools. In Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103. ACE '10. Australian Computer Society, Inc, Darlinghurst, Australia, 15–22.

[3] Blumfeld, P. C., Solowy, E., Marx, R. W., Krajcik, Joseph S. Guzdial, Marc, and Palincsar, A. 1991. Motivating Project-Based Learning: Sustaining the Doing Supporting the Learning. Educational Psychologist 26, 3 & 4, 369–398.

[4] Boehm, B. W. and Turner, R. 2004. Balancing agility and discipline. A guide for the perplexed. Addison-Wesley, Boston.

[5] El-Abbassy, A., Muawad, R., and Gaber, A. 2010. Evaluating agile principles in CS Education. International Journal of Computer Science and Network Security 10, 10.

[6] Frey, K. 1983. Die sieben Komponenten der Projektmethode - mit Beispielen aus dem Schulfach Informatik (The Seven Components of the Project Method – with Examples from CSE). LOG IN 3, 2, 16–20.

[7] Hedin, G., Bendix, L., and Magnusson, B. 2008. Teaching Software Development Using Extreme Programming. In Reflections on the Teaching of Programming, J. Bennedsen, M. E. Caspersen and M. Kölling, Eds. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 166–189.

[8] Kastl, P. and Romeike, R. 2015. "Now they just start working, and organize themselves" First Results of Introducing Agile Practices in Lessons. In Proceedings of the 10th Workshop in Primary and Secondary Computing Education. WiPSCE '15. ACM, New York, NY, USA, 25–28.

[9] Li, P. L., Ko, A. J., and Zhu, J. 2015. What Makes a Great Software Engineer? In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 700–710.

[10] McKinney, D. and Denton, L. F. 2006. Developing collaborative skills early in the CS curriculum in a laboratory environment. SIGCSE Bull. 38, 1, 138.

[11] 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.

[12] Meyer, B. 2014. Agile! The good, the hype and the ugly. Springer.

[13] Romeike, R. and Göttel, T. 2012. Agile Projects in High School Computing Education: Emphasizing a Learners' Perspective. In Proceedings of the 7th Workshop in Primary and Secondary Computing Education. WiPSCE '12. ACM, New York, NY, USA, 48–57.