Scrum as a Method of Teaching Software Architecture

Gero Wedemann
Institute for Applied Computer Science
University of Applied Sciences Stralsund
Germany
gero.wedemann@hochschule-stralsund.de

ABSTRACT

In advanced software engineering courses, students profit from taking greater responsibility for and control of their own learning. Because students tend to lack knowledge about the topics they want to learn, facilitation of this learning process is not straightforward. Scrum is a well-known empirical process control model designed to manage complex activities, and is a promising method for facilitating the learning process. This idea was field-tested in a software architecture course as part of a master's degree course with a few participants. Group processes were monitored by questionnaires based on theme-centered interaction. Learning outcomes were tracked by written tests and small projects. The success of the course was compared with the courses of previous years with respect to formal evaluation and grades. Scrum helped in this course by creating a good working environment. Learning outcomes and student satisfaction were substantially better than in previous years. These results suggest that Scrum is a suitable framework for university teaching. This study may serve as the basis for future research and to inspire other instructors.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; *Systems analysis and design*; • **Software and its engineering** → *Software architectures*; *Designing software*;

KEYWORDS

Scrum, Software Architecture, Teaching, Theme-Centered Interaction

ACM Reference Format:
Gero Wedemann. 2018. Scrum as a Method of Teaching Software Architecture. In *ECSEE'18: European Conference of Software Engineering Education 2018, June 14–15, 2018, Seeon/ Bavaria, Germany*. ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.1145/3209087.3209096

1 INTRODUCTION

Facilitating students' learning at universities poses many challenges to instructors: They have to stimulate and nurture learners' interest in the topics of the course. Working and learning in groups provides many opportunities, but at the risk of ineffective group processes. Constraints such as short timeframes for classes or competition for attention and time with other courses pose additional challenges. Living learning at universities can be fostered by principles such as individualization, learning instead of teaching, handing over responsibility to the students, feedback and meta–communication, and learning by doing [8]. Although principles such as "feedback and meta-communication" and "learning by doing" are straightforward, it is not clear how "individualization" and "handing over responsibility to the students" can be implemented in software engineering classes. Students do not know enough about the subject and therefore do not know which topics and aspects of the topics may be interesting to them, how they could approach the topics, and how much time learning requires. For the students, this makes planning difficult and may intimidate and discourage them. The standard approach for instructors is to provide an initial plan for a course and allow the students to modify it. In my experience, students generally make only minor changes because they lack an overview of the topics. This counteracts the principles described above, possibly leading to low motivation for the course.

To address this problem, we applied elements of the software process management framework Scrum [10] as a teaching method. Similar approaches to learning have been tried in schools called "Agile Schools" [1] and in enterprises such as "Agile Learning" [5]. Scrum has the advantage in that it provides planning tools for complex, emergent, and fluid fields, and it is familiar to students from project management courses.

Scrum was implemented in the learning context by dividing the semester into Sprints of 4–5 weeks. At the beginning of the course, students created a Product Backlog containing all the things they wanted to learn. For each Sprint, students chose the exam type for this Sprint as a Sprint Goal and created a Sprint Backlog containing the plan for this Sprint. The potentially shippable product of each sprint were the gained competencies of the students demonstrated in the exam or a small project work after each sprint. At the end of each Sprint a Sprint Review and Retrospective was done. Each class contained elements of the Daily Scrum, such as reporting what had been done, planning the next class including modifying the Sprint Backlog, and identifying impediments. We did not define roles such as Scrum Master and Product Owner. The instructor fulfilled the tasks of Scrum Master and Product Owner. We did not use a Burn Down Chart or User Stories. These are not essential parts of Scrum as defined in the Scrum Guide [10]. As well, a general Definition of Done seemed not to be useful in this context.

As a testbed for this idea, the Software Architecture course, which is part of the revised and updated Computer Science master's degree course, was chosen. This course has four contact hours per week for 15 weeks. The master's course comprises small groups

ECSEE'18, June 14–15, 2018, Seeon/ Bavaria, Germany

ACM ISBN 978-1-4503-6383-9/18/06…$15.00
https://doi.org/10.1145/3209087.3209096

of 5–15 students. Different tools were used to study the effectiveness of the approach: At the end of each Sprint, students filled out questionnaires about their feelings toward the course and answered questions checking their understanding of the topics. The marks of the individual assessments were tracked. To compare the course with previous courses, the results of the final standard evaluation by the university and the final exam grades for the course were compared with previous years. The results of the first test run in the summer semester of 2017 are promising. They indicate that, for small groups, Scrum is a good teaching method. To gather more data, a course with the same setup will take place in the summer semester of 2018.

2 METHODS

2.1 Software Architecture course structure

The course takes place in the summer semester of the Computer Science master's degree course at the University of Applied Sciences, Stralsund. The examination regulations provide the option to select a suitable exam type. The course has six European Credit Transfer and Accumulation System (ECTS) credits. In the summer semester of 2017, only six students attended this course. These students achieved their bachelor's degree in Computer Science (four students) and Medical Informatics (two students) at the end of the preceding semester at the university and were highly motivated. The summer semester spans 15 weeks with many public holidays such as Easter in between. It was divided into three Sprints of 4–5 weeks depending on whether there were public holidays. Each week, students attended two classes of one and a half hours. Attendance was not compulsory, but five or six students out of the six enrolled students attended each class.

The description of course content and learning objectives in the study regulations covers many aspects of software architecture and allows the selection of topics that are of interest to the students. Therefore, the learning objectives regarding software architecture were defined by the students in each Sprint Planning with help and approval of the instructor. Additionally the course was designed to increase the competence of self managed learning as well as as communication and collaboration between the students as learning objectives.

In the first class, the instructor explained the content and structure of the course and presented a list of possible topics. He pointed out to the students that the list is simply an open list of ideas, that it contains more than can be done in one semester, and that it is open to topics of interest to them within the scope of the course. Students were provided with various books on software architecture. They discussed in small groups their interests and compiled a list of topics to work on this semester on a pin–board, the Product Backlog. For the three Sprints, students selected "Foundations of software architecture", "RESTful APIs", and "Cloud". Students picked only topics prepared by the instructor. After that, students selected topics for the first Sprint. The instructor decided that the first Sprint had no exam. For the other two Sprints, students chose an exam type in agreement with the instructor. It was possible to choose between a short oral examination, a short written exam, writing an essay, working on a small project, or something else suggested by the students. Students decided in both Sprints to implement a sample software project using the principles and technology of that Sprint.

The structure of each class was based on theme–centered interaction (TCI) [9] and did depend on the topics. The instructor formulated a theme for each class at the beginning. Work forms and social forms changed generally between plenum discussions and groups of 2 or 3. Enterprise patterns were taught by regular lectures with PowerPoint slides.

At the end of each lesson students and the instructor selected from the Sprint Backlog items for the next class and clarified the content and possible ways to work on them. The students defined workpackages by themselfes, if this was possible. The instructor planned the next lesson based on this material. If students were not able to define workpackage by themselves the instructure gave homework like reading assignements based on this planning using the e–learning platform.

At the end of each Sprint, a Sprint Review was held: After the second and third Sprints, the students presented their work. After all of the Sprints, the students completed two questionnaires (see Section 2.2), which had no influence on the mark. Thereafter, in the Sprint Retrospective, the students could give feedback and create a plan for things that they thought should be done differently in the next Sprint. The Retrospectives were facilitated as commonly done in agile projects [2]. Results of the Retrospectives were collected on a pin-board, which was photographed at the end of the Retrospective. Review, Retrospective, and Planning for the next Sprint occurred on the same date or on two separate dates depending on the time constraints.

Topics and materials were provided to the students through the university's ILIAS e–learning platform.

2.2 Collecting the data

We were interested in how the design of the course influences the acquisition of competencies, student satisfaction, and group processes. These questions were investigated with different methods. At the end of each Sprint, the students completed two questionnaires: One questionnaire was designed to understand changes in the group processes. This questionnaire was designed on the basis of TCI [9]. TCI describes the processes and interactions that are active in situations and groups. These are affected by "I", "We", "It", and "Globe". "I" means the individual participant in the group. "We" describes the group work interactions. "It" is the task the group was formed to deal with. Finally, "Globe" describes the conditions in which the group works. How these four factor influence each other will either enable or hinder living learning and cooperative work, transparent interactions, and growth-stimulating communication. [8:101].

The students were asked how they felt toward the group, the topics, and the course of study, on a scale from 1 (best) to 5 (worst). This is equivalent to the four factors of the TCI model. To prevent errors concerning the baseline, students were also asked how they felt at the beginning of the Sprint.

The second questionnaire was designed to check the basic understanding of the topics worked on in the Sprint on the level of remembering and comprehending conceptual and factual knowledge [6], by asking simple questions, e.g., "Name four domain logic

patterns", "How can you add semantics in RESTful APIs?" or "What are the characteristics of cloud computing?" After the second and third Sprints, application of procedural knowledge was tested by student projects. It was measured using the project mark as graded by the instructor.

In 2012–2016, the course took the more traditional form of seminars. We assessed the learning outcomes of the course by comparing the overall course grade of each student for the last years. Student satisfaction with the courses was evaluated at the end of the course using an official university questionnaire. Additionally, students gave oral feedback, which was recorded in writing by the supervisor.

3 RESULTS

The answers to the questionnaires showed that the students generally felt good and that this feeling improved during the course (Fig. 1). Students felt good ("I"). At the beginning, one student had mixed feelings (3), but at the end, no student gave a rating worse than 2. At the beginning of the course, not all students felt good toward the group ("We"). This evolved into generally very good feelings toward the group at the end of the course. At the beginning of the course, two students rated their feelings toward the course content as 2 and 3, and the rest of the course as 1. At the end of the course, all students rated their feelings as 1, and only one student as 2. During the course, feelings toward the course of study evolved from mostly positive to more mixed feelings, but at the end, students were generally positive about the course of study.

In the first Retrospective, students pointed out as positive the close relationship between theory and practical experience, flexibility, reflection of learned things, as well as group work and discussions. They demanded more practical examples. This was realized in the second Sprint. After the second Sprint, students requested a less time–consuming second project, which was taken into account in the third Sprint. In the feedback at the end of the course, students stated that this course was the highlight of the whole master's degree course. They especially liked the agile organization of the course and doing small projects instead of written or oral exams. They said that three different topics in one course was their upper limit, and might be too much. Half the students asked for an introduction to JavaScript/Node.js technology at the beginning of the course (see below).

The learning outcomes measured by the questionnaire were more mixed (Fig. 2). In the first Sprint, students found it difficult to remember what factors influence software architecture, and none was able to recall enterprise patterns [4]. In the third Sprint, students found it difficult to define the characteristics of a cloud service. All other learning outcomes were good to very good. The students chose in both Sprints to implement sample software of the technologies learned with JavaScript/Node.js, which was known to a majority of the students. In the second Sprint, the students chose to re–implement a REST interface defined with Swagger with a hypermedia API framework of their choice. In the third Sprint, students implemented a simple REST service using Amazon AWS Lambda and Dynamo DB. All projects were excellent and graded 1.0 (best grade). The students spent more time on the projects than they were required to.

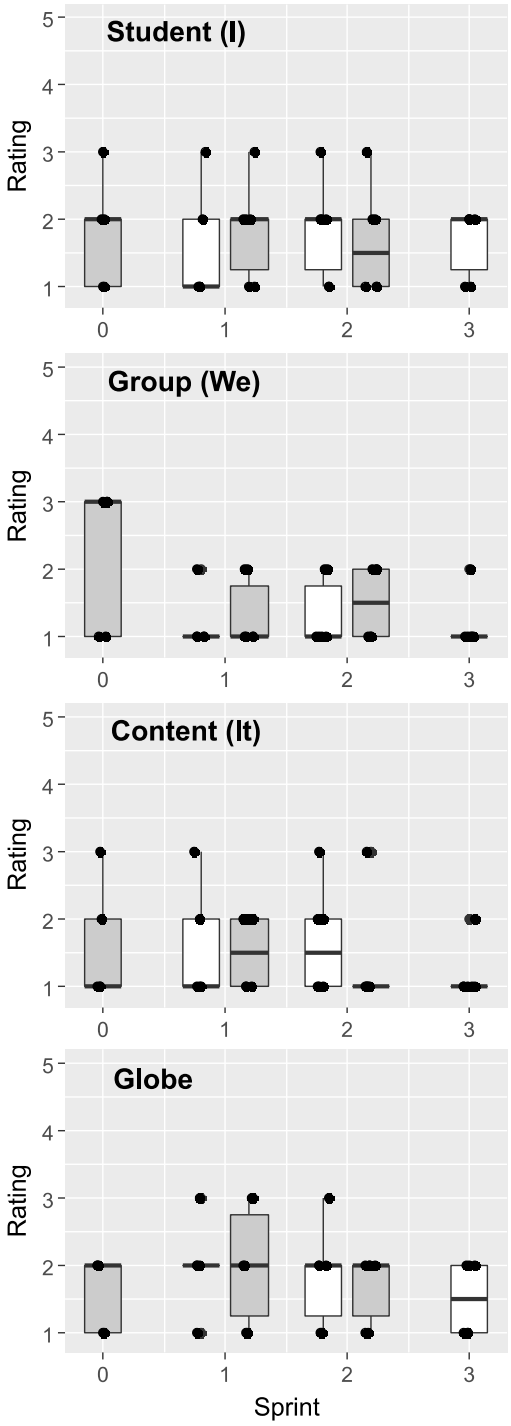

Figure 1: Evaluation of the four factors of the group as a boxplot. A rating of 1 is best, 5 worst. The central line marks the median. The lower and upper hinges correspond to the first and third quartiles, respectively [11]. Data in white boxes denote the feeling at the moment of the evaluation, and those in the gray boxes at the beginning of the Sprint.

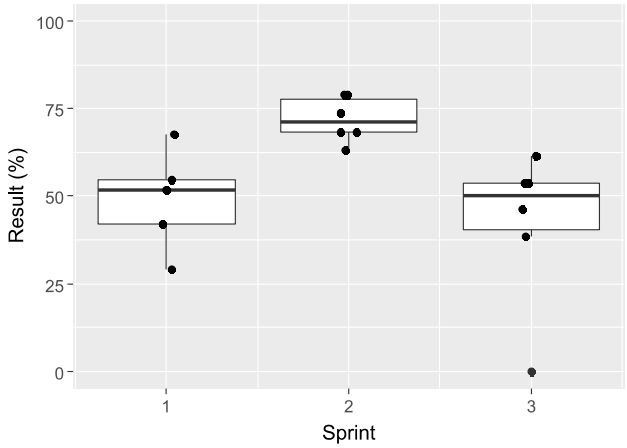

Figure 2: Results of exams testing knowledge after each Sprint as a boxplot.

In comparison with previous years, the instructor felt that the course was better. This feeling was supported by the results of the formal evaluation, where students completed the university's official evaluation questionnaire (Fig. 3 upper panel). Here, the students judged the course to be substantially better than in previous years. Additionally, the considerably better grades obtained than in the past indicate better learning outcomes (Fig. 3 lower panel).

All data are available from the Open Science Framework at the following DOI: 10.17605/OSF.IO/6VA4C.

4 DISCUSSION

The idea of applying Scrum to a learning situation in a software architecture course was successful in this first trial run. Learning outcomes and student satisfaction were better than in previous years. The group itself worked well together. However, it is too early to make generalizations from this. The idea was tried out only once, and there is no guarantee that other groups would see similar success. Only six students attended the course, which is a smaller number than in previous years. The comparison with previous years is difficult because not only was the curriculum of the master's degree changed substantially, but the course duration was also changed from two contact hours per week to four contact hours. A direct comparison is therefore difficult because many other parameters were changed. However, the good response of the students and the good learning outcomes are encouraging in proceeding with this approach.

Measuring the learning outcomes with questionnaires revealed interesting results. Enterprise patterns were taught to students by regular lectures with PowerPoint slides because no better option was available. After the lectures, the students expressed their satisfaction. Interestingly, however, in the questionnaire, they were unable to recall them. This underlines the questionability of the effectiveness of conventional lectures, as discussed in the literature, e.g. [3]. This problem could be addressed by choosing alternative learning approaches. Students also had problems remembering

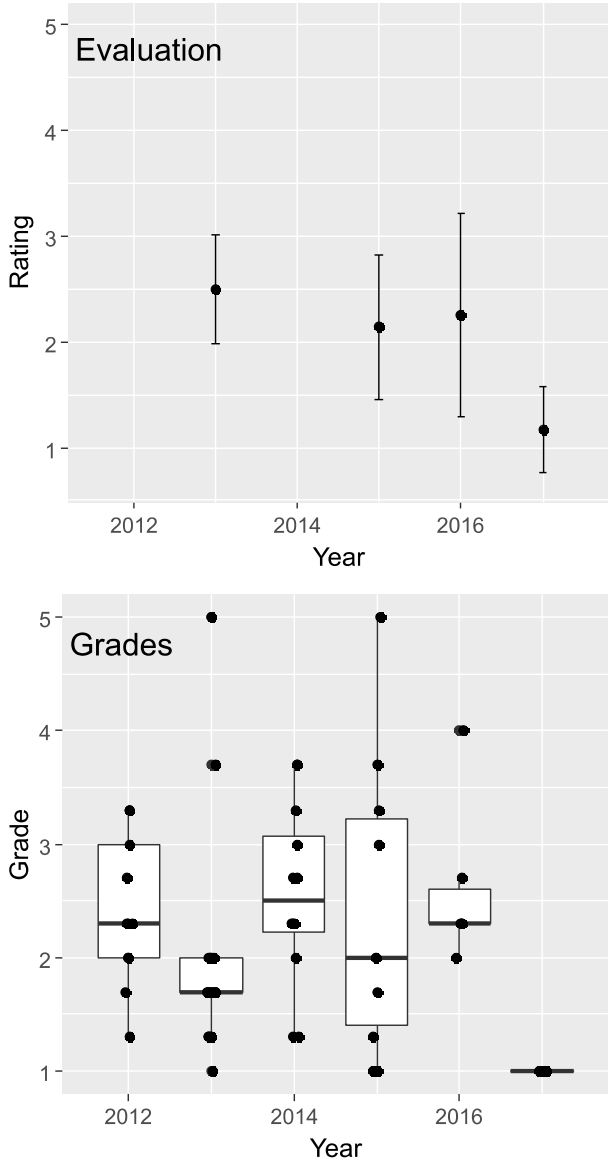

Figure 3: Comparison with former years. Upper panel: Formal university evaluation of the course by students. Shown are mean values and standard deviation. Individual ratings are not available. No data are available for 2014. Lower panel: Boxplot of exam grades. A grade of 1 is best, and a grade of 5 is worst.

which factors influence software architecture and the characteristics of a cloud service. In an interview after the course, the students explained that they found the content trivial and did not believe that they would not be able to explain it afterward. This could be addressed by alternative ways of teaching in which these seemingly trivial facts are practiced.

For the Product Backlog, students selected only topics that were prepared by the instructor. Despite the fact that students selected the topics by themselves, two out of five students were unsure about their choice. This confirms the initial observation that the students found it difficult to plan a course by themselves. Interestingly, the students did not select "Domain Driven Design" (DDD), which was the topic the instructor had least prepared. For the other topics the instructor prepared many possible backlog items while for DDD only two superficial backlog item were prepared. This underlines that it is important to prepare a range of interesting topics that the students can choose from and to ensure that they have a good enough understanding of what the possible topics specifically include. It is also very important to structure Sprint Planning carefully and to support students so that students do not develop the feeling that they are left on their own.

5 OUTLOOK

In the summer semester of 2018, the next course will be given. This will be taught by applying the same concept, with small modifications as discussed in the previous section. Moreover, additional information will be tracked: The number of attendant students will be noted each session. The instructor will make subjective estimates about the status of the group concerning the four factors of a TCI group each session, because more questionnaires may have a negative impact on student motivation [7]. It will be interesting to find out whether the positive findings of this paper will hold for the new group. The results will be presented at the conference.

ACKNOWLEDGMENTS

I thank Bertin Hoffmann, Laurens Groß, and Tilo Zülske for critical reading of the manuscript and helpful comments. I acknowledge Emma Hoyle for copyediting the manuscript.

REFERENCES

[1] [n. d.]. Agile Schools. http://www.agileschools.com/
[2] Judith Andresen. 2017. *Retrospektiven in agilen Projekten: Ablauf, Regeln und Methodenbausteine* (2 ed.). Hanser, München.
[3] L. Deslauriers, E. Schelew, and C. Wieman. 2011. Improved Learning in a Large-Enrollment Physics Class. *Science* 332, 6031 (May 2011), 862–864. https://doi.org/10.1126/science.1201783
[4] Martin Fowler. 2003. *Patterns of enterprise application architecture.* Addison-Wesley, Boston, Mass. [u.a.].
[5] Benjamin P. Höhne, Sandra Bräutigam, Jörg Longmuß, and Florian Schindler. 2017. Agiles Lernen am Arbeitsplatz – Eine neue Lernkultur in Zeiten der Digitalisierung. *Zeitschrift für Arbeitswissenschaft* 71, 2 (2017), 110–119. https://doi.org/10.1007/s41449-017-0055-x
[6] David R. Krathwohl. 2002. A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice* 41, 4 (Nov. 2002), 212–218. https://doi.org/10.1207/s15430421tip4104_2
[7] Barbara Langmaack and Michael Braune-Krickau. 2010. *Wie die Gruppe laufen lernt: Anregungen zum Planen und Leiten von Gruppen. Ein praktisches Lehrbuch* (8 ed.). Beltz, Weinheim Basel.
[8] Gerhard Portele. 1995. Gute akademische Lehre - Meine acht Grundprinzipien der Hochschuldidaktik. In *Lebendiges Lernen in toten Räumen. Zur Verbesserung der Lehre an der Hochschule*, Rüdiger Standhardt and Cornelia Löhmer (Eds.). Focus-Verlag, Gießen, 139–144.
[9] Mina Schneider-Landolf, Jochen Spielmann, and Walter Zitterbarth (Eds.). 2017. *Handbook of Theme-Centered Interaction (TCI).* Vandenhoeck & Ruprecht, Göttingen. Google-Books-ID: hyU0MQAACAAJ.
[10] Jeff Sutherland and Ken Schwaber. 2017. Scrum Guide. http://www.scrumguides.org/scrum-guide.html
[11] Hadley Wickham. 2016. *ggplot2: elegant graphics for data analysis.* Springer, Cham.