Do High and Low Performing Student Teams Use Scrum Differently in Capstone Projects?

Maria Paasivaara, Jari Vanhanen, Ville T. Heikkilä, Casper Lassenius, Juha Itkonen, Eero Laukkanen
Department of Computer Science
Aalto University, Helsinki, Finland
Email: {*firstname.lastname*}@aalto.fi

***Abstract*—In this paper, we present how student teams applied Scrum in their capstone projects and compare how the Scrum usage differed between the high and low performing teams. 16 student teams of 7–9 persons were taught Scrum during a lecture and a 4-hour Scrum simulation game, after which they applied Scrum in their capstone projects developed for external industrial customers during a six month period. We collected data by surveys (98 responses) and by interviewing all 16 teams at the end of the course. Our results show that the frequency of Scrum practice usage does not differ significantly between the low and high performing teams. However, the interviews showed that the high performing teams applied Scrum practices more thoroughly, e.g., by taking better advantage of the retrospective meetings and Daily Scrums and working more face-to-face, while low performing teams paid less attention to the proper way of applying the Scrum practices.**

***Index Terms*—Scrum; capstone project; software engineering education**

I. Introduction

During recent years, agile methods, such as XP [1] and Scrum [1] have become increasingly popular in the software industry, as companies want to reap the claimed benefits of these methods, e.g., improved performance and fast response time. As university level teaching aims to provide trained professionals to the industry, the software engineering curricula must respond by providing students with knowledge and experiences on how to use the agile methods in practice.

Scrum is one of the most popular agile methods in the industry [2]. Even though the basic principles and ceremonies of Scrum are easy, their application is not, as the developers of Scrum state on the first page of the Scrum Guide [3].

We have taught Scrum at Aalto University for several years, both using traditional lectures combined with expert visitors from industry, and by employing a Scrum LEGO simulation game [4] in classroom settings. In 2014 we adopted Scrum in our capstone project course.

Our contribution is two-fold: 1) We present how we designed the capstone course to teach students Scrum in practice and 2) we describe differences in the use of Scrum between high and low preforming project teams.

II. Related literature

We believe that the most common method for teaching Scrum in tertiary education has been the traditional lecture model (see e.g. [5]). Recently, Scrum has gained popularity as the development life-cycle model of software engineering education capstone projects [6], [7], [8]. During these courses the students usually emulate small industrial software development projects within the limits afforded by the educational setting. Industrial software companies can be recruited to act as customers for the projects to add realism to the course [6], [7], [8].

Rico and Sayani [6] found that the students should learn agile methods before trying to apply them during a capstone course. According to their descriptive case study, having to learn the agile methods during the capstone course caused notable confusion and slowed down the development progress.

Using Scrum in software engineering courses has been discussed in a literature survey [9], which identified seven studies where Scrum was applied on capstone projects. The author states that it is important to provide students with practical experiences, and define the roles carefully. In particular, it is important to have a non-student to be the Product Owner, but students can work as Scrum Masters.

III. The Capstone Project Course

During the winter of 2015-2016, we used Scrum in our capstone course for the second time. Below, we briefly describe the course, and how we applied Scrum.

A. Learning Goals

Student teams develop real software for real clients. The main learning goals are: 1) understanding the common challenges involved in commercial software development projects, 2) being able to apply Scrum and other work methods and tools in a real project, and 3) learning new technologies. Having industrial clients means that the course projects should be able to provide them with valuable results, while maximizing student learning.

B. Stakeholders and Roles

In 2015-2016, we had 16 six-month projects carried out by teams of 7–9 students. Each team consisted of 7–8 B.Sc. level computer science students and a Scrum Master, who was typically a software engineering Master's student. Most students are very motivated to take the course due to its' practical nature and aim for the highest grade. As the course is mandatory for the B.Sc students, some of them naturally aim only at passing the course. All B.Sc. students must allocate 225 hours for the project work, which is monitored through time reporting. The Scrum Masters can opt for project sizes between 100–175 hours based on the number of credits they want.

The course staff consists of the course teacher and several coaches. Each coach typically guides two teams in issues related to the work methods. Prior to the course, the teacher identifies client candidates. There are no limitations with regard to the project topics as long as they have a realistic scope. Only one team can get each topic. Most of the clients are from the industry, and a few from the university. A client representative works as a Product Owner for each team.

Grading is based on the project work and the whole team, including the Scrum Master, receives the same grade. All projects have three project reviews, after which the coaches and the clients evaluate their teams: the client the intermediate and final results, and the coach the Scrum usage, work methods, and final results. The clients' evaluations indicate their satisfaction with regard to their expectations. The course personnel gently guides the evaluation if a client's expectation level clearly differs from those of the other clients. The coaches' evaluations of the final results compare the scope and quality of the delivered software among the projects as they have seen several projects over the years. The coaches will also tune the points slightly if there are some factors that make a project clearly easier or more difficult than a typical course project.

C. Teaching Scrum

Before the projects begin we teach students Scrum basics during a 2-hour lecture, and arrange a 2-hour workshop for the Scrum Masters on their role. All teams participate in a 4-hour Scrum LEGO simulation game where the Scrum process is simulated by running 3–4 Sprints in which programming work is replaced by building LEGOS, as described in [4]. Finally, the course guidelines for applying Scrum are presented during a 2-hour lecture. During the projects the teacher arranges six experience exchange sessions where representatives of the student teams can share their work practice or development process related problems and discuss potential solutions with other teams, the teacher and a visiting industrial professional.

D. Application of Scrum

All teams are required to use Scrum as defined in the Scrum Guide [3], but also considering the exceptions mentioned in the course instructions. If a particular Scrum requirement fits poorly into the project, the team may propose changes. The enforced process framework helps the teams define their development process quickly, and aims to ensure that all teams get practical experience on using software engineering practices that are aligned with the educational goals of the course.

Each team must have at least six Sprints, which are about three weeks long, containing 40 hours of effort per student. The first Sprint is spent setting up the project, but thereafter the students are expected to deliver software increments.

The teams must have a Product Backlog which contains items that have a description, priority order, and effort estimate. The teams must create a Product Vision with the PO that shortly characterizes the project. Furthermore, they must have a Sprint backlog that contains all identified tasks with a name/description and an effort estimate. Both the product and the sprint backlog must be managed in a dedicated backlog management tool. The teams must arrange sprint reviews, retrospectives and planning events, but due to the small effort allocated for each sprint and the busy schedules of the Product Owners, the course recommends combining them into a single day event. The course requires having a Scrum stand-up meeting at least once per week, as typically students spend 0.5–2 days on project work per week. On-line stand-ups are allowed if team is not able to meet regularly otherwise. We recommend as much collocated work as possible.

IV. Methodology

A. Research Question and Data Collection

In this paper, we aim at answering the following research question: *How did the application of Scrum differ between high and low performing teams?* To this end, we collected both quantitative and qualitative data.

At the end of the course, after the final demonstrations, the students were asked to fill in a questionnaire based on their knowledge and experiences during their project. In total, we received 98 answers to the survey, which is available on-line[1]. The questions relevant to this research asked to choose how often (Q1–Q2) or frequently (Q3–Q10) the team used the listed 10 agile practices and to estimate time spent co-located with more than 3 other team members (Q11).

We interviewed all 16 teams directly after the final demonstrations. Each team was interviewed separately for about 40 minutes. The interviews were voluntary, and did not affect the grading. Three researchers not in responsible teaching positions conducted the interviews. Two researchers were present in each interview, one being the main interviewer and the other taking notes and asking clarifying questions. In these open-ended interviews we asked the teams to 1) describe how they had applied Scrum in their projects, 2) explain the most challenging aspects of the project and the best experiences, 3) give advice to next year's student teams, and 4) comment on the Scrum Lego game. We did not pose the questions to any individual team member but to the whole team, thus anybody present was free to answer. Regarding all teams, all or most members present participated in the discussions. There were no big open disagreements in any interview. All interviews were tape recorded and transcribed by an external company.

B. Data analysis

We selected the high performing teams by taking the four teams with the highest scores (T1a: 44.7, T1b: 44.7, T3: 44.4, T4: 43.0) and the low performing teams by taking the four teams with the lowest scores (T13: 38.6, T14a: 38.3, T14b: 38.3, T16: 36.3). The maximum score based on the project results only was 45 points (30 from the client and 15 from the coach).

The responses to Q1–Q11 were analyzed using the team-wise median. The responses to how often the team used the product (Q1) and Sprint backlog (Q2) and how frequently the

[1] http://bit.ly/scap-data

team did effort estimation (Q5), had hands-on demo with the PO (Q8) and had the PO available to the team (Q10) were analyzed as ordinal variables. According to Scrum guidance, release planning (Q3) is conducted once in a release project and Sprint planning (Q4), Sprint review (Q7) and retrospective (Q9) are conducted once a Sprint. Since the teams did not work full-time, the teams were instructed to conduct Scrum stand-up meetings (Q6) at least once per week, which we consider the minimum for regular stand-up meetings. These Scrum practice variables (Q3, Q4, Q6, Q7, Q9) were converted into categorical variables based on whether the team had conducted the events approximately according to the Scrum guidance ("Scrum" category), considerably more often than the Scrum guidance ("More often" category), considerably less often than the Scrum guidance ("Less often" category) or never.

The transcribed post-course interviews were analysed by qualitative coding after which the main points were extracted to a data extraction sheet. The codes used included, e.g., Sprint length, Sprint review, retrospective, daily Scrum, backlog, definition of done, collaboration with the customer / Product Owner, team communication, communication tool, backlog tool, challenge, best on course, advice, and Scrum game.

V. Results

A. Scrum practices

Based on the quantitative data, all analyzed teams followed the Scrum practice in Sprint planning (Q4), and Sprint review (Q7). One low performing team (T14b) conducted retrospectives (Q9) less often than the Scrum practice. The rest of the teams followed the Scrum practice regarding retrospectives. One high performing team (T1b) never did release planning (Q3) and three (T1a, T3, T4) did release planning more often than the Scrum practice. Two low performing teams (T14a, T14b) followed the Scrum release planning practice and two (T13, T16) did release planning more often than Scrum recommends.

There were no statistically significant differences (Mann-Whitney U, asymptotic significance, 2-tailed) between the high and low performing teams regarding their way of working with the product backlog (Q1), $U = 4.00, p = .22$, and Product Owner (Q10), $U = 3.50, p = .16$, nor in the frequency of the Scrum stand-up meetings (Q6), $U = 4.00, p = .24$, effort estimation (Q5), $U = 5.00, p = .32$, or had hands-on demo with the PO (Q8), $U = 6, p = .31$.

There was a statistically significant difference (Mann-Whitney U, asymptotic significance, 2-tailed) regarding the use of a Sprint backlog (Q2) between the high and low performing teams, $U = 0.00, p = .01$. One low performing team (T16) answered "usually", three low performing teams (T13, T14a, T14b) answered "almost always", while all four high performing teams (T1a, T1b, T3, T4) answered "always".

Based on the interviews, the high performing teams took Scrum practices more seriously. They were more eager to experiment with new ways of working and improved their ways of working somewhat systematically. The low performing teams had a slightly different mindset. Their main focus was to get the project done and they did not pay much attention to the process. They sometimes omitted or conducted superficially some Scrum practices. These differences were most notable regarding the retrospectives, collocated working sessions, daily Scrums, and effort estimation.

According to the interviews, the usage of retrospective meetings differed between the high and low performing teams regarding their length, the practices used, the number of improvement actions identified and implemented, as well as the experienced usefulness of the retrospectives. Three high performing teams reported having regular one hour retrospective meetings, while the length of the retrospectives of the low performing teams ranged between 5–30 min. Three high performing teams tried out different retrospective methods, e.g., root cause analysis or "six thinking hats", while the lowest performing teams used post-its or simple discussions. Both high and low performing teams changed their ways of working based on the retrospectives. The high performing teams reported several big changes that they successfully implemented based on the retrospectives. The low performing teams reported improvement ideas, but were not able to implement them all. The high performing teams found the retrospectives to be more important than the low performing ones, as illustrated by the following quote from a high performing team:

> *Without retros we would not be here, we changed the working practices a lot [based on retros], e.g., meeting a lot face to face. We would have stood still without them.*

Based on the interviews, high performing teams paid more attention to learning in order to get each team member able to work on the system comprehensively. Subsequently, there would be several persons that understand specific areas of the system. In one high performing team, the team members took turns having the role of "Sprint leader" for one Sprint. In another high performing team, the developers switched between the front-end and the back-end in the middle of the project to give everybody a chance to learn both. Low performing teams often assigned features and tasks to developers who were "the best at performing that specific task". A member of a low performing team could concentrate on developing one specific component during the whole project, which easily caused them to become bottlenecks.

Even though there was no statistically significant difference between the teams in arranging stand-up meetings, in interviews the high performing teams reported arranging stand-up meetings more often than the low performing teams. One of the low performing teams reported having stand-up meetings meetings once per week, while the rest did not arrange regular stand-up meetings at all. Three high performing teams arranged most of their stand-up meetings during collocated working sessions. Two of these teams arranged stand-up meetings 1–2 times per week, while the third one had it three times per week. The third team used an online chat tool when not meeting face-to-face. The following comment from a high performing team illustrates the benefits from these meetings:

> *The biggest advantage [of stand-ups] is that they encourage to tell about blocking issues. If a team does not talk, there might*

be blockages, for which someone might know an easy solution. It [stand-up] encourages to bring them up.

In the interviews, the low performing teams did not report benefits from performing estimation. They mentioned the following estimation related issues: not estimating systematically and estimates differing hugely from the realized effort. The high performing teams reported using planning poker regularly for estimation, while only one low performing team used it regularly. The high performing teams reported many benefits from planning poker, including the following: becoming better at estimating, raising a lot of useful discussions and everybody being able to express their honest opinions. This is illustrated by the following quote from a high performing team:

The greatest advantage of it [planning poker] was that it caused a lot of discussion, on what we can do and why we cannot do something. The numbers could come out of the blue [...] but the following discussion was good.

The Sprint length ranged from one week to three weeks, two weeks being the most common. Two of the high performing teams had 3 week sprints while all low performing teams had 2 week sprints. The low performing teams explained that short sprints help to synchronize work and create deadlines, which makes work faster. Those high performing teams that had three week sprints found that length suitable for them, as well.

Both high and low performing teams arranged face-to-face Sprint change meetings with their Product Owner in every Sprint as instructed by the course. A Sprint change meeting normally included a review of the previous Sprint, a retrospective and the planning of the following Sprint. According to the interviews, the high performing teams had slightly longer Sprint change sessions ($\approx$ 3 h) than the lower performing teams ($\approx$ 2 h).

B. Team collocation

Although collocation is not a Scrum practice per se, it is highly recommended by Scrum guidance. Based on the quantitative data, the amount of high collocation varied quite notably between the teams. The most collocated team (T1a) spent, on the average, more than 80% of their time collocated with four or more people, while the least collocated teams (T4, R13, R16) spent 10–20 %. However, there is no statistically significant difference (Mann-Whitney U, asymptotic significance, 2-tailed) between the high and low performing teams in regards to the percentage of high collocation, $U = 6.00, p = .56$.

Interviews supported that the highest performing teams worked more collocated than the lowest performing teams. Of the three highest performing teams two performed almost all work in the same space during two weekly sessions (around 10 hours / week), and one team changed to two weekly collocated working sessions after having challenges.

It's really important that we work in the same space. In the beginning we did some work distributedly, but that was so unefficient. We achieved almost nothing. [...we would recommend to work collocated] as much as possible.

Of the four lowest performing teams only one had two regular weekly face-to-face working sessions, while two teams normally had only one shorter (2-4 hours) working session per week and one team met face-to-face only in bi-weekly Sprint change sessions.

VI. Discussion and Conclusions

In this paper we explored whether high and low performing teams applied Scrum differently in a capstone project. All teams followed the course instructions quite well and used the Scrum practices required, which explains why only a few statistically significant differences in the practice usage were identified. However, the interviews revealed that the high performing teams learned to apply the practices more thoroughly receiving more advantage to their teams, e.g., they improved their ways of working by proper retrospectives and had efficient communication due to frequent Scrum stand-up meetings and face-to-face working sessions, while the low performing teams easily dropped practices (e.g. stand-up meetings) or did not use them efficiently (e.g. retrospectives).

The qualitative differences in Scrum practice use between the high and low performing teams might be explained by student motivation: good teams emphasized learning, not just passing the course, and also aimed for a good grades by applying Scrum properly, as that was part of the grading criteria.

A limitation of this study is that we analyzed only eight teams, which makes receiving statistically significant results unlikely. Thus, our results give an interesting direction to more thorough future research. In addition, selecting the high and low performing teams only by the scores can be seen as a limitation. However, this was the best available method for differentiating between the teams in this case.

Future research should dig deeper into these findings, e.g., by studying if the proper usage of Scrum practices improves the project results and how we should support students to really learn to use Scrum practices.

References

[1] C. Larman, *Agile and iterative development : a manager's guide*. Boston, MA: Addison-Wesley, 2004.

[2] VersionOne, Inc, "7th annual "state of agile development" survey," http://www.versionone.com/pdf/ 7th-Annual-State-of-Agile-Development-Survey.pdf, 2013.

[3] K. Schwaber and J. Sutherland, "The scrum guide – the definitive guide to scrum: The rules of the game," http://www.scrum.org/Scrum-Guides, October 2011.

[4] M. Paasivaara, V. Heikkilä, C. Lassenius, and T. Toivola, "Teaching students scrum using lego blocks," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 382–391.

[5] P. Kruchten, "Experience teaching software project management in both industrial and academic settings," in *Proceedings of the 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2011)*, 2011, pp. 199–208.

[6] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in *Proceedings of the 2009 Agile Conference (AGILE '09)*, 2009, pp. 174–179.

[7] V. Mahnič, S. Georgiev, and T. Jarc, "Teaching scrum in cooperation with a software development company," *Organizacija*, vol. 43, no. 1, pp. 40–48, 2010.

[8] M. Persson, I. Kruzela, K. Allder, O. Johansson, and P. Johansson, "On the use of scrum in project driven higher education," in *Proceedings of the World Congress in Computer Science, Computer Engineering and Applied Computing*, 2011.

[9] V. Mahnič, "Scrum in software engineering courses: an outline of the literature," *Global Journal of Engineering Education*, vol. 17, no. 2, 2015.