How Does Participating in a Capstone Project with Industrial Customers Affect Student Attitudes?

Maria Paasivaara
Aalto University & IT University of Copenhagen
Finland / Denmark
maria.paasivaara@gmail.com

Dragoș Vodă
Aalto University
Finland
dragos.voda@aalto.fi

Ville T. Heikkilä
Aalto University
Finland
v.t.heikkila@gmail.com

Jari Vanhanen
Aalto University
Finland
jari.vanhanen@aalto.fi

Casper Lassenius
Aalto University
Finland
casper.lassenius@aalto.fi

ABSTRACT

Teaching of software engineering using capstone projects has seen a steady growth over the years with overwhelmingly positive reported experiences. Discerning what students consider of value before and after a software project course, is crucial for developing a relevant curriculum. This paper reports on the affective learning outcomes of a Scrum based capstone course with industrial clients. We measured affective learning as changes in the attitudes towards the difficulty and importance of certain aspects present in software development, including technical, teamwork and customer interaction. Data was collected from 14 student teams of 7-9 members in the form of team interviews and individual surveys, with 86 students participating in the interviews and 86 valid survey answers. Our results show that students largely change their attitudes in the desired direction after the course regarding both importance and difficulty, with 57% reporting an increased understanding of the importance of collaboration and communication within the team and 44% reporting less than expected difficulty in learning new technologies/programming languages.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Software and its engineering** → **Agile software development**; *Programming teams*;

KEYWORDS

software engineering education, capstone courses, Scrum, affective learning

ICSE-SEET'18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5660-2/18/05…$15.00
https://doi.org/10.1145/3183377.3183398

ACM Reference Format:
Maria Paasivaara, Dragoș Vodă, Ville T. Heikkilä, Jari Vanhanen, and Casper Lassenius. 2018. How Does Participating in a Capstone Project with Industrial Customers Affect Student Attitudes?. In *ICSE-SEET'18: 40th International Conference on Software Engineering: Software Engineering Education and Training Track, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3183377.3183398

1 INTRODUCTION

Teaching software engineering using capstone projects has seen a steady growth over the years. Universities aim at providing education that is aligned with industry needs and better prepare students for the workplace. Agile practices, especially Scrum, are being employed with increasing frequency and success in university software project courses [2] [10].

Student learning in capstone courses, in particular when working with industrial clients, covers both technical aspects, as well as soft skills, including teamwork and communication. In addition, we have observed that participating in such a course seems to have an effect on student attitudes related to software engineering topics, as they gain experience from a simulated, yet very "real" project. While students report positive experiences with Scrum [12] and better skills related to working in software development [10] [15] as a result of participating in capstone courses, few studies have looked into the attitude and perspective changes of students regarding critical aspects of software development.

Having a proper understanding of the underlying reasons behind Agile practices is paramount to their application in the right context to avoid oblivious imitation and replication [8]. In trainings accomplished by professionals, cultivation of the 'why' is as important as the teaching of the 'how' since even apparently good approaches applied in the wrong context can lead to undesired outcomes [4].

In this paper, we report on the affective learning outcomes of a Scrum based capstone course with industrial clients run at Aalto University in Finland. Our goal in this paper is to uncover how students' perceptions of certain aspects of software engineering evolve after completing a capstone project course in software engineering. Knowing how the course affects the students' views is crucial to adapting and improving our methods, in order to provide the best possible education for the current working environment.

Furthermore, correct identification of key aspects can lead students to self-directed learning.

Following a literature review on related work in Section 2, we describe the teaching environment and the course in Section 3. We investigate the perceived difficulty and importance of selected software engineering aspects using the methods explained in Section 4 and present our results in Section 5. Finally, we discuss the results and conclude the paper in Section 6.

2 RELATED WORK

Teaching Agile methods in higher education is becoming commonplace, however, not much research exists on measuring the learning effect of the applied methods. This paper builds upon a body of 10 prior studies that were selected based on their relevance to Agile education and its application in practice. All papers offer a description of the teaching environment and method applied together with varying forms of results in terms of student feedback and project completion.

The majority of the studies present their findings in terms of student feedback regarding the course, the agile practices used or both, obtained by the means end-course surveys, and project completion. Although course feedback is essential for updating and refining content, assessment of the learning outcomes of these courses, beyond grading and student feedback, is needed. To this end, a smaller subset goes even further employing interviews, logs, staff observations and pre/post course surveys in an attempt to measure the progress and attitudes of the students.

Table 1 contains the relevant studies on project courses using agile methods. The main findings in terms of attitude assessment and reporting are summarized in Table 2.

When assessing process related feedback and attitudes, Kropp et al. report positive reception of Agile values in [5] while employed students from [6] mention their use of the learned Agile practices in their daily jobs. Mahnic et al. [11] compared the views of students to those of professional developers and found a shared understanding of the importance of teamwork and communication among team members, and good communication with the Product Owner contributing to project success. The same study revealed that accurate user story and velocity estimation were viewed as least important by both groups. Matthies et al. [12] used a ten question survey to gauge student perception of Scrum and its practices with encouraging results: nine out of ten questions had statistically significant positive answers at the end of the course. In a similar fashion, Mahnic et al. [9] employed two surveys in order to measure students' perceptions of Scrum and the agreement with its stated benefits. Results indicate a strong positive opinion towards Scrum and a statistically significant level of agreement with eight out of nine assertions of Scrum benefits. Even though the student feedback was mainly positive in these studies, a couple of studies received also negative comments: According to Kropp and Meier [5] Agile was claimed to be praised too much without mentioning its possible drawbacks. Zorzo et al. [17] received complaints that due to shorts sprints, the lectures could not advance fast enough and the early sprints were finished without actual deliverables.

Finally, student learning and attitudes has been tackled by a small number of papers. Kropp et al. [7] report on the student recognition of the benefits related to the retrospectives and customer involvement, while Rodriguez et al. [14] note the perceived difficulty of daily meetings in a distributed context. Dunlap [3] concludes that after a capstone course based on Problem Based Learning strategies, students felt significantly more confident in adressing real-world demands specific to software projects. Bastarrica et al. [1] studied the evolution of students' attitudes after completing a capstone course that followed Agile practices. The course projects were chosen from a pool of submissions coming from the industry, government or within the university and were planned to last three iterations over the 15 weeks. A total of 38 students were split into teams of 4 to 7 people to tackle projects independently after an initial lecture held by an industry expert on appying agile practices. An evaluation of student attitude evolution was performed using pre and post course surveys in which students had to rate four items: technical challenge, negotiation with the client, project planning and teamwork for both value and difficulty [1]. As the ratings had to sum up to 100, it was easy to observe which items were considered more valuable or more difficult. The results indicate that the perceived importance of soft skills grows while dropping in relation to technical challenge and the perceived difficulty of soft skills increases. Their findings show how the students learn through their mistakes of either underestimating the importance of a certain practice or overestimating their own abilities. Although the results are encouraging, no factual link has been made between the elements of the course and the learning outcomes.

Table 1: Studies on Scrum based capstone courses.

Paper	Year	Study purpose	Agile Methodology	Attitude Assessment	Attitude Reporting
[9]	2010	Course design and analysis	Scrum	X	X
[11]	2012	Identifying Scrum practices as success factors	Scrum	X	X
[17]	2013	Case study on Scrum course	Scrum	-	-
[7]	2014	Course design and analysis	Scrum	-	X
[14]	2016	Results of introducing Agile coaching	Scrum	X	X
[5]	2013	Improvement of Agile course	Scrum/XP	-	X
[6]	2016	Curriculum setup and observation	Scrum/XP	-	-
[12]	2016	Linking elements of Scrum to students satisfaction or challenges	Scaled Scrum	X	X
[1]	2017	Measuring the variation in soft skills and technical challenge	Agile practices	X	X
[3]	2005	Evolution of student self-efficacy	N/A	X	X

Table 2: Previous results on student attitude changes.

Attitude related results	References
The perceived value of addressing technical challenges DOES drop after the course while its difficulty DOES NOT. The perceived value of the negotiation with the client DOES NOT grow after the course while its difficulty DOES drop. The perceived value of project planning DOES grow after the course while its difficulty DOES NOT. The perceived value of teamwork DOES NOT grow after the course while its difficulty DOES.	[1]
Positive attitude towards Scrum and its usage after the course	[12] , [9], [5], [14]
Perceived difficulty of meetings in a distributed context	[14]
Realization of importance of teamwork and communication with the Product Owner	[11]
Acknowledgement of benefits of retrospectives and customer involvement and availability	[7]
Increased confidence in tackling software development demands	[3]

3 CAPSTONE PROJECT COURSE

In this section we briefly describe the capstone software project course at Aalto University, in Finland, as run during the academic year 2016-2017. We describe the stakeholders and their roles in the projects, the learning goals, the teaching methods, and the software development process used.

3.1 Stakeholders and Roles

In 2016-2017, we had 14 six-month projects carried out by teams of 8–9 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 of the B.Sc. students were in their second year. About half of the Scrum Masters had previously taken the same course as a development team member during their B.Sc. studies. Many of the Scrum Masters already had professional work experience, and had taken some advanced software engineering courses in their M.Sc. studies, such as software project management, requirements engineering, testing or architectures.

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 guides one or two teams in issues related to work methods. Prior to the course, the teacher identifies plenty of client candidates. There are no limitations with regard to the project domain or implementation tehcnologies. 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 his or her team. The clients participate in the course, for example, in order to get an initial version of a new software system, but they may also be looking for promising new employees. Having industrial clients means that the course projects should be able to provide the clients with valuable results, while maximizing student learning.

3.2 Learning Goals

The main learning goals of the course 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. We believe that the second year B.Sc. students, who often have no experience from large software projects, commonly expect before the course that the most difficult challenges will be related to learning and using technologies that are new to them. Participating in a real project allows them to see also many other typical challenges. We hope that the project makes them realize that learning new technologies is not so difficult, at least for those students who are pursuing their careers as software developers. However, understanding software requirements, customer collaboration, and teamwork among other things may involve difficult challenges that are important to tackle in order to complete a project successfully.

3.3 Teaching Methods

Almost all the lectures and workshops organized by the course personnel before and during the projects focus solely on teaching Scrum. Studying new programming languages and implementation tools is carried out by the teams independently, but sometimes the clients provide a technology expert who gives some help with the technologies.

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 (see e.g.,[13]) where the Scrum process is simulated by running 3-4 Sprints in which programming work is replaced by building LEGOS. Finally, the course guidelines for applying Scrum are presented during a 2-hour lecture. The guidelines require also using a couple of additional practices such as personal time tracking, and the production of three documents that describe shortly the product vision, system design and used work practices.

During the projects the teacher arranges four 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. The coaches, who help the teams with using Scrum and other work methods, meet the teams at least six times during the project, and are otherwise available to answer questions by e-mail.

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.

3.4 Software Development Process

All teams are required to use Scrum as defined in the Scrum Guide [16], 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 means that the sprints are at most three weeks long and contain at most 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 first sprint typically involves getting to know the team members and the client, crafting a product vision, creating an initial product backlog, selecting and studying technologies, and planning the initial application of Scrum and other work methods and tools.

The teams must have a Product Backlog which contains items that have a description, priority order, and effort estimate. The course requires that the description of those Product Backlog items that represent new SW features follows some published user story template. The teams must create a Product Vision with their Product Owner that shortly characterizes the "why, what and for whom" aspects of 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 instead of using, e.g., Post-It notes or Excel sheets.

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 the team is not able to meet regularly otherwise. We recommend as much collocated work as possible but the different schedules of the students mean that the degree of collocation varies.

The course requires having a Definition of Done (DoD) for product backlog items and for sprints. Furthermore the DoD must set a reasonably high criteria for unit testing and functional testing coverage, and enforce the use of some coding standard.

4 METHODOLOGY

4.1 Research Question

The paper aims to uncover how participating in a capstone project course affects student attitudes. We aim to answer the following research question:

RQ: How did the students change their attitude or perception regarding software development related topics during the capstone course?

4.2 Data Collection

Both qualitative and quantitative data were collected in the form of a survey and group interviews at the end of the course in April 2017. As the paper aims to uncover student learning, and especially how their perceptions change, we have selected what we consider the most relevant aspects needed to be developed as software engineers and inquired about their change after the course.

We collected the data at the end of the course right after each teams' final demonstration. First, the students were asked to fill in a questionnaire. In total, we received 86 valid answers to the survey. Second, we interviewed all 14 teams separately directly after each team had filled in the survey. Each interview took for about 45 minutes, ranging from 33 to 57 minutes. Altogether 86 students participated in the interviews. Participation in the survey and the interview was voluntary, and did not affect the grading. Two researchers not in responsible teaching positions conducted the interviews, one being the main interviewer and the other taking notes and asking clarifying questions.

The survey questions focused on the perceived difficulty and importance of various elements present in software development that we consider to be a learning priority. The answers were in a form of a 7-point scale rating an "Is Difficult" or "Is Important" statement from "Strongly Disagree" to "Strongly Agree" while the change rated the same aspect from "Much more difficult" to "Much less difficult". We also collected demographic data related to the students' previous professional experience and time allocated to activities specific to a software project.

In the interviews we asked the students 1) to describe the most important things they had learned during the course that would be of value in their professional life, 2) to explain the most difficult problems the team had encoutered, 3) to describe if and how their attitude towards some software development related topic had changed during the course, 4) if there had been something that they expected to learn during the course, but did not learn, 5) how the course could support better their learning, 6) what was the best thing during the course, and 7) what pieces of advice would they give to other students participating in the course in the future. In this paper, we concentrate espcially to questions three (attitude change) and one (important things learned).

For the first three interview questions we gave students a couple of minutes to think about it and write their individual answers on post-it notes (as a note for themselves) and after that we asked the students to present and explain their answers orally one student at the time. For the last four questions 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.

4.3 Data Analysis

From a total of 130 students that participated in the course, 113 passed the course and of those 97 responded to the survey resulting in a 86% response rate. Filtering incomplete and invalid answers and removing one team form the pool due to the high drop out rate, we reached a response rate of 76%, with 86 valid answers.

Regarding the change in perception on Difficulty and Importance, we had 15 statements that we analysed; 6 Difficulty related and 9 Importance related. These 15 statements can be found in Table 3. Based on previous teaching and course experience, we saw these as the subject of the biggest attitude changes, thus presenting the highest study interest.

A couple of metrics were derived from the collected data and are explained below:

Attitude change: the Attitude change scale is composed by the answers for the followed questions from the perception change in Difficulty and Importance parts and are coded in the [-3,3] range

with -3 representing "Much less difficult/important", and 3 defining "Much more difficult/important".

Desired attitude: building upon the Attitude change we define the desired attitude as an ideal view expected from the students after completing the course. Our definition of a desired attitude stems from past course results coupled with first hand experience with the software development industry and they embody the work practices and values most commonly leading to efficient work and ultimately a successful project. Since the goal was to teach the students, specifically the ones with an improper preception, that certain aspects of software development are more difficult/important than the common expectation of a person with little or no experience with real projects, a positive number indicates a shift towards the desired attitude. A single exception is present for the "Learning new technologies/programming languages" aspect, where the coding is reversed due to the opposite desired attitude, to recognize the facile nature of assimilating new technical skills.

For an overview of the change in attitude we used a sign test on the answers for the statements.

The transcribed interviews were analysed by qualitative coding after which the main points regarding each interview question were extracted to a data extraction Excel sheet. In addition, well describing quotations from the interviews were chosen to illustrate the main findings.

5 RESULTS

In this section we present the student attitude changes based on the survey answers and compare them to the interview findings. The main survey findings are presented in Figures 1 and 2, and in Table 3.

Figure 1 shows the student perceptions on the studied importance and difficulty dimensions after the course. Regarding the importance hypothesis, students agree on the importance of these aspects of software development. However, the difficulty dimension brought more dispersion of opinions. Students saw "managing the customer's expectations" as less difficult than what we expected. This can be explained by our industrial customers being very cooperative, as that was encouraged by the course personnel. In addition, the customer companies wanted to create a good image of their company in the eyes of the students as potential employers. Actually, several students reported in the interviews that the customer collaboration was one of the most important things they had learned and seemed to be happy with this collaboration in general.

Regarding the "difficulty of working efficiently in a distributed manner" the student opinions are very disperced, the median being in "somewhat agree". In the interviews several students mentioned that they had noticed to their surprice how much more efficient face-to-face work was. They had expected to be able to just split the work and then work on their own. However, they soon realized that this was not possible.

> *I was surprised at how much easier it was to discuss code and the related problems face-to-face.* — Student, Team 6

The "difficulty of collaboration and communication within the team" is quite dispersed as well. This clearly depended on the team. In some teams there were communication problems and afterwards the team improved communication, while in other teams they put effort on communication early on and the team started to work well right in the beginning. Especially these team that had had challenges brough up the importance of communication in the interviews.

> *Communication is important. In the beginning our team members did not communicate acively, which caused problems, but it improved a lot towards the end.* — Student, Team 2

Table 3: Results Summary

	Answers			
Hypothesis	Towards desired attitude	Towards undesired attitude	p	Mean value
Learning new technologies is less difficult.	38	10	$\ll 0.001$	0.45
Understanding the customer needs is more difficult.	20	25	0.551	0.01
Collaboration and communication within the team is more difficult.	26	26	1	0.06
Working efficiently in a distributed manner is more difficult.	35	16	0.01	0.31
Managing customer expectations is more difficult.	18	26	0.291	-0.15
Estimating task effort is more difficult.	40	19	0.008	0.5
Following a well-defined process when working in a team is more important.	41	8	$\ll 0.001$	0.58
Quality Assurance more important after the course.	40	5	$\ll 0.001$	0.57
Keeping the customer happy is more important.	22	6	0.003	0.24
Team-building is more important.	30	2	$\ll 0.001$	0.56
Collaboration and communication within the team is more important.	49	1	$\ll 0.001$	0.93
Communication between the team and customer is more important.	38	2	$\ll 0.001$	0.6
Development team co-location is more important.	39	2	$\ll 0.001$	0.69
Managing customer expectations is more important.	24	7	0.003	0.35
Version control is more important.	35	6	$\ll 0.001$	0.56

Value range is [-3,3]. Shifts towards the desired attitude are represented by positive changes while negative changes represent shifts towards an undesired attitude

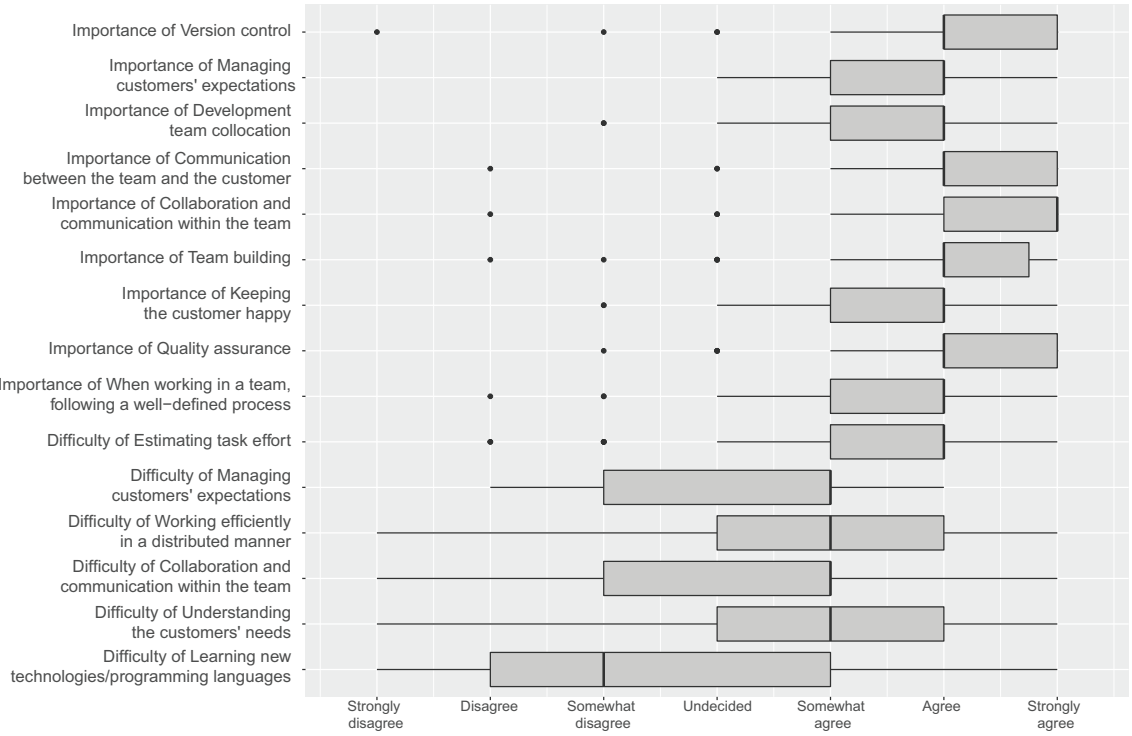

Figure 1: Post-course level of agreement with the importance and difficulty statements

Some students saw collaboration and communication to take more effort and time and being more difficult than they had expected. On the otherhand some valued the collaboration and found teamwork to be more fun than they had expected.

> *...team internal communication took surprisingly lot both resources and time.* — Student, Team 8

> *...software development in a team is fun ... I got a good feeling during the whole course that I'm studyig the corrent topic.* — Student, Team 13

Based both on the survey and interview answers "the importance of collaboration and communication within the team" was highly valued. In Figure 2. we can see that this statement has also the biggest change towards the desired attitude with 57% of survey respondents reporting an increased importance. The course seems to have worked extremely well in teaching the importance of communication.

Figure 2. shows the change in student attitude during the course. Again, regarding all importance dimensions the student attitudes change somewhat towards the desired direction, i.e., undestanding the importance of these dimensions better after participating in the course. However, the difficulty aspects have more varied changes. The student attitudes on the "difficulty of managing customer's expectations" and the "difficulty of understanding the customer's needs" show slight change towards undesired attitude, i.e., students see these actually easier than what they had expected. This might be due to our industrial customers being "easy" customers in the sense that they were easily happy with what the students produced, they wanted to give students a good experience and were committing time and effort to work as cooperative customers helping the students to both understand what they want and many of them could even advice with the technologies used. Even though these customers are maybe "easier" customers than some that the student will encounter during their professional carrier, we do not see this as negative finding, even though it was contradicting in what we expected. Instead, based on the interviews, the students found this collaboration as a good example of cooperative customers and learned the importance of close customer and Product Owner collaboration.

> *One of the most important [learning] was how you can work with the Product Owner. [...] you can iteratively collaborate with the Product Owner to reach the joint target. We did a suggestion and then they commented and then we improved and made a new, better suggestion, and in time it starts to be what they want and everybody thinks it's great* — Student, Team 6

> *I was surprised at how much they [the customer] liked everything we had done, they liked it in practice.* — Student, Team 3

One of most interesting findings based on the interviews and supported by the survey is that during the course students had noticed that learning new technologies was not as difficult as they had expected: 44% of the survey respondents reported decreased difficulty in learning new technologies / programming languages. In most projects students had to use several technologies that were totally new to them and they had to learn them from scratch.

> *...learning new technologies wasn't in the end as difficult as I maybe expected. When you got into it, it was easier than I expected. It tells that you can learn everything when you just start doing.* — Student, Team 3

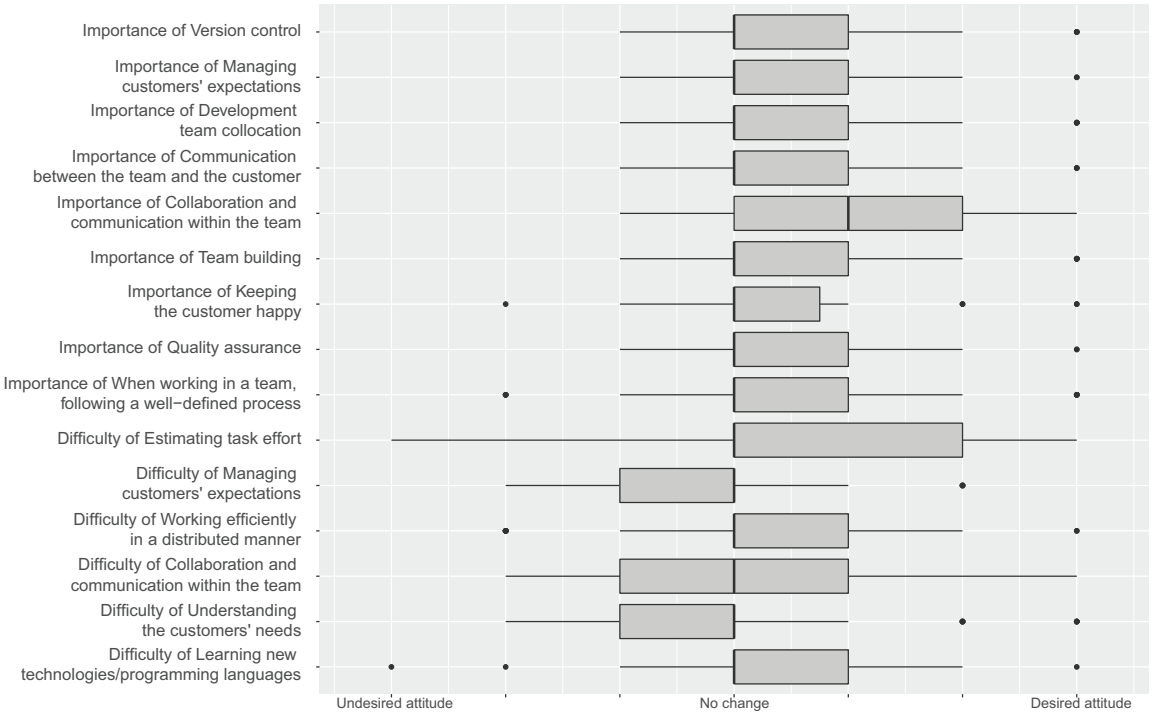

Figure 2: Level of changes during the course in the importance and difficulty statements

...during the course you learned in general how to learn new technologies. That might be more important than just Android [...] [you learn new technologies] by doing. By using it to something. Just reading detailed instructions is useless in order to be able to use it to something concrete. Concrete application is the most important... — Student, Team 3

...there were so many different technologies in this project that had to work together [...] the ability to learn new technologies developed a lot. I noticed that in the end when something new came out it was much easier to absorb it than in the beginning. — Student, Team 13

During the interviews, we asked about the most important things the students had learned that would be of value in their professional lives. As the two most frequently mentioned topics rose: 1) teamwork in a bigger team, and 2) using the Scrum framework in practice. These both were mentioned as one of the three top most important things learned by almost half of the interviewed students.

Many realized that development is actually much more teamwork had they had expected, as their previous school tasks had required mainly individul work.

Software development is teamwork more than I had imagined — Student, Team 2

They also noticed that teamwork can actualy be both efficient and fun. That was a positive surprice to many as they had been fearing much worse regarding the collaboration in a big team often with several previously unknown team mates.

The other frequently mentioned "most important things learned" were the importance of communication and collaboration both within the team and with the customer, as well as learning new technologies. In addition, the importance of several key software development aspects, like the importance of quality assurance and version control, were frequently mentioned.

As the most important attitude changes the students mentioned: the importance and value of software development tasks "surrounding" pure coding, such as the importance of quality assurance, the importance of version control, the importance of careful documentation of code, as well as the importance of planning and performing reserch on different alternatives.

You start to understand why we want tests. When the project gets so deep that one person cannot understand it all, then it's good to have tests that confirm that you can see if something breaks — Student, Team 4

Many realized for the first time that software development is actually much more than just coding and how much effort actually needs to be put on just setting up the project and communicating.

A new thing was that how much time is required to set up a common ground for everyone, like channels of communication, regular meetings, [...] because otherwise the whole thing would just collapse like a house of cards. — Student, Team 4

[my attitude changed] regarding the division of time to different areas. A lot of time went to choosing technologies and research in the beginning [...] and then in setting up the development environment — Student, Team 2

Several students also mentioned that this was the first time they understood the whole life cycle of a software development project from the set-up until delivering to the customer.

...how a real project works, this would be a kind of an introduction to how a project could work in the real world. — Student, Team 1

Some students revealed that they had been quite unsecure of their own software development skills in general before the course as they did not have real software development experience, but during the project they had understood that sofware development is not as difficult as they had expected and when you just start working you learn and can soon do much more than you had even imagined. Thus their confidence and trust on their own skills improved.

In the beginning I expected the software development to be really difficult. [...] I even doubted whether I would be able to pass the course, whether I can do anything. But in the end I first of all learned a lot and actually could do more than I had known. — Student, Team 5

Some mentioned that to their surprise being a good coder seems not to be enough in software development.

I noticed that being a good coder and knowing many technologies might not be the most important thing for the project success, but there are many other things. I could claim that over half of it is other than coding knowledge. — Student, Team 13

A few, maybe not so strong coders, were actually relieved to notice that other skills than just coding are needed and appreciated in real software development projects.

6 DISCUSSION AND CONCLUSIONS

This paper reported on the affective learning outcomes of a Scrum based capstone project course with industrial clients. We measured affective learning as changes in the attitudes towards the difficulty and importance of certain aspects present in software development, including technical, teamwork and customer interaction. The most important findings are the following:

- Student attitudes regarding the importance of all chosen software engineering aspects moved during the capstone course towards the desired direction, i.e., students became to understand the importance of these aspects better through practical experience.
- The biggest change towards the desired direction appeared in the "Importance of collaboration and communication within the team" with 57% of students reporting an increased importance regarding this aspect.
- Students found learning new technologies / programming languages somewhat less difficult than they had expected with 44% of respondents reporting decreased difficulty regarding this aspect.
- During the course, the students saw in practice that software engineering is much more than just coding. Their understanding of the importance of these other aspects of software engineering, such as teamwork, quality assurance, version control and customer collaboration rose.
- Friendly industrial customers made the students realize the importance of customer collaboration, but also that managing customers' expectations and understanding customers' need is, contrary to our expectations, slightly less difficult than the students had expected.

These results show that the course actually succeeds remarcably well in achieving the affective learning outcomes.

When comparing our results to previous studies regarding attitude changes we can find many similarities. Our study confirms the positive attitude towards Scrum and its usage after the course reported also by [12] , [9], [5], [14]. Our results showed that almost half of the students mentioned learning Scrum as one of the top three in the most important things they had learned during the course. Realization of the importance of teamwork and communication with the Product Owner reported by [11] were both highly valued by our students.

However, surpricingly, our results somewhat contradict the results by Bastarrica et al. [1] who studied the evolution of students' attitudes after completing a capstone course following Agile practices. Unfortunately their and our measures were not exactly the same, but we can do some comparison. According to their results the perceived value of negotiation with the client does not grow after the course, while its difficulty drops. On the contrary, our students found the importance of communication between the team and the customer slightly more important after the course, while our students agreed that both the difficulty of managing customers' expectations and the difficulty of understanding the customer needs slightly drops after the course. According to Bastarrica et al. [1] the perceived value of project planning does grow after the course while its difficulty does not. We did not measure this aspect in the survey, but in the interviews a few students mentioned that the importance of project planning had increased. Finally, Bastarrica et al. report that the perceived value of teamwork does not grow after the course while its difficulty does. This seems to contradict our findings as we found that the importance of collaboration and communication within the team increased and a large number of student in the interviews reported teamwork experience as highly valued. Moreover, according to our results the difficulty of collaboration and communication within the team stayed approximately the same.

Our capstone course used to be at the Master's level, but is now obligatory at the bachelor level. Therefore, the technical skills of the students starting the course are now lower than previously and almost half of the students (45% in this instance of the course) do not have working experience from software development. Even though the course is challenging for them, based on our results we can say that the students learn a lot and they can do more than they and we teachers expected. In addition, the student became more confident regarding their own skills in being able to contribute in a real software development project and being able to absorb new technical knowledge quickly. Most importantly, they change their perception on the importance of many essential software development aspects, such as the importance of using version control, the importance of proper documentation of code, the importance of testing and the importance planning. Thus, after this course they understand better what software development includes in real life. We expect that this creates a good bases, and especially a good motivation, to learn more on these aspects of software engineering in their next more advanced software engineering courses. Therefore, having this kind of "attitude affecting course" already during the second year of the bachelor studies seems actually to be a better idea than what we had originally suspected.

For future work we encourage other teachers to study attitude changes in their courses so that we could learn more on this important topic, especially as we found some contradictions when comparing our results to previous studies.

REFERENCES

[1] María Cecilia Bastarrica, Daniel Perovich, and Maíra Marques Samary. 2017. What can students get from a software engineering capstone course?. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. IEEE Press, 137–145.

[2] Daniela Damian, Casper Lassenius, Maria Paasivaara, Arber Borici, and Adrian Schröter. 2012. Teaching a globally distributed project course using Scrum practices. In *Collaborative Teaching of Globally Distributed Software Development Workshop (CTGDSD), 2012*. IEEE, 30–34.

[3] Joanna C Dunlap. 2005. Problem-based learning and self-efficacy: How a capstone course prepares students for a profession. *Educational Technology Research and Development* 53, 1 (2005), 65–83.

[4] Yuri Khramov. 2006. The cost of code quality. In *Agile Conference, 2006*. IEEE, 7–pp.

[5] Martin Kropp and Andreas Meier. 2013. Teaching agile software development at university level: Values, management, and craftsmanship. In *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. IEEE, 179–188.

[6] Martin Kropp, Andreas Meier, and Robert Biddle. 2016. Teaching Agile Collaboration Skills in the Classroom. In *Software Engineering Education and Training (CSEET), 2016 IEEE 29th International Conference on*. IEEE, 118–127.

[7] Martin Kropp, Andreas Meier, Magdalena Mateescu, and Carmen Zahn. 2014. Teaching and learning agile collaboration. In *Software Engineering Education and Training (CSEE&T), 2014 IEEE 27th Conference on*. IEEE, 139–148.

[8] Philippe Kruchten. 2007. Voyage in the agile memeplex. *Queue* 5, 5 (2007), 1.

[9] Viljan Mahnic. 2010. Teaching Scrum through team-project work: Students' perceptions and teacher's observations. *International Journal of Engineering Education* 26, 1 (2010), 96.

[10] Viljan Mahnic. 2012. A capstone course on agile software development using Scrum. *IEEE Transactions on Education* 55, 1 (2012), 99–106.

[11] Viljan Mahnic and Igor Rozanc. 2012. Students' perceptions of Scrum practices. In *MIPRO, 2012 proceedings of the 35th international convention*. IEEE, 1178–1183.

[12] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker, and Hasso Plattner. 2016. How Surveys, Tutors and Software Help to Assess Scrum Adoption in a Classroom Software Engineering Project. In *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*. IEEE, 313–322.

[13] Maria Paasivaara, Ville Heikkilä, Casper Lassenius, and Towo Toivola. 2014. Teaching students scrum using LEGO blocks. In *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 382–391.

[14] Guillermo Rodríguez, Álvaro Soria, and Marcelo Campo. 2016. Measuring the Impact of Agile Coaching on Students' Performance. *IEEE Transactions on Education* 59, 3 (2016), 202–209.

[15] Andreas Scharf and Andreas Koch. 2013. Scrum in a software engineering course: An in-depth praxis report. In *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. IEEE, 159–168.

[16] Jeff Sutherland and Ken Schwaber. 2017. The Scrum Guide. http://www.scrumguides.org/

[17] Sergio Donizetti Zorzo, Leandro de Ponte, and Daniel Lucredio. 2013. Using scrum to teach software engineering: A case study. In *Frontiers in Education Conference, 2013 IEEE*. IEEE, 455–461.