Abstract - In order to prepare students for increasing use of agile methods in industry, teaching these methods is becoming an important part of the Software Engineering curricula. At the University of Ljubljana Scrum has been systematically taught since 2009 in the framework of the software engineering capstone course. The paper describes the course content and analyzes results of the survey that was performed among students with the aim of identifying those practices that students perceive most important for the success of Scrum-based software projects. Students’ opinions on 12 typical practices representing possible success factors are described and compared to opinions of professional developers in order to find out similarities and differences in their perceptions. Both groups of respondents identified team-work and communication among team members, as well as good communication with the Product Owner, most important. Students also stressed the importance of strict adherence to the notion of “done”, while professional developers ranked third Sprint Planning Meetings and maintenance of Sprint Backlog. Accuracy of user stories and velocity estimation was rated least important by both groups of respondents.

I. INTRODUCTION

Numerous agile software development methods [1] have appeared in the last 15 years that – in contrast to disciplined plan-driven approach advocated by the quality models – “value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan” [2]. According to the Agile Adoption Rate Survey [3] performed by Dr. Dobb’s Journal in 2008, agile teams report significant improvements in productivity, quality, and stakeholder satisfaction, and reasonable improvements in cost. A similar survey conducted by VersionOne [4] additionally reports an enhanced ability to manage changing priorities and significantly improved project visibility. On the other hand, the same survey has revealed that the lack of experience with agile methods, and the conflict between the company’s culture and core agile values are the leading causes of failed agile projects.

Due to the increasing use of agile methods in industry in the last few years, the teaching of agile methods for software development has become an important issue [5]. Although some universities are starting to teach courses on agile methods, these are still rather new [6]. Since core software engineering courses mainly teach the traditional plan-driven approach, the capstone course seems to be an appropriate place for the introduction of agile software development.

An extensive study of 49 capstone projects [7] using different software processes (ad hoc, MIL-STD-498, IEEE 1074, Team Software Process, and Extreme Programming (XP)) reports several benefits of introducing the agile approach in place of the traditional plan-driven processes. The agile approach is more appealing to student teams, and project success, measured by how well the finished software met customers’ expectations, surpassed that of previous projects using non-agile software processes. Coupal and Boechler [8] report a similar experience comparing student projects developed following an agile approach to their previous projects developed in a traditional way. Successful introduction of agile methods in capstone courses is also reported in several other studies, for example [6], [9]–[11].

State of agile surveys conducted by VersionOne since 2006 indicate that Scrum [12, 13] is the most widespread agile method and its share is permanently increasing (see Fig. 1). In light of the above, a new software engineering capstone course was designed at the University of Ljubljana, Slovenia, that introduces Scrum as a framework for the planning and management of students’ projects. The course ran for the first time in the Academic Year 2008/2009, and as well as teaching students agile software development through team-work on two industry-relevant projects, also paid special attention to students’ perceptions about the learnability and usability of Scrum. An empirical evaluation [14] has shown that students enjoyed learning Scrum and successfully grasped its main strengths. Their perceptions of Scrum improved from Sprint to Sprint and were in statistically significant agreement with anecdotal evidence about Scrum’s benefits reported in the literature.

In order to help students to better understand the problem of agile estimation and planning, and to improve
their skills, the course was upgraded in the Academic Year 2009/2010, with the project work being designed as an observational study providing data for empirical evaluation of students’ skills with special emphasis on user stories estimation, release planning, fulfillment of scope, and velocity tracking [15, 16]. The course was conducted in a similar way in the Academic Year 2010/2011, further emphasizing the use of user stories as a means for requirements specification [17], and agile project planning and management [18].

In order to obtain students’ opinions, several surveys were preformed during the last three years. An important part of the survey that was conducted at the end of Academic Years 2009/10 and 2010/11 was focused on students’ perceptions of 12 typical Scrum practices in view of their impact on the success of Scrum projects. Proper execution of each practice was treated as a possible success factor and students were asked to evaluate its importance using a 5-point Likert scale. The same survey was later conducted among a group of developers at the largest Slovenian media house, which used Scrum in a project of rebuilding the web site of its daily newspaper with the highest circulation. This paper presents the results of both surveys and compares opinions of students with those of professional developers.

The remainder of the paper is organized as follows: Section II contains a brief description of the capstone course. Section III describes 12 typical Scrum practices the respondents had to rate. The results are presented in Section IV and discussed in Section V. Section VI provides a conclusion.

II. COURSE DESIGN

The course is designed to teach Scrum in a near real-world environment, augmenting the Scrum method with user stories, release planning, and velocity tracking. Students are required to work in groups in order to develop a quasi-real project on the basis of user requirements provided by a domain expert playing the role of the Product Owner. The course lasts 15 weeks and is taken by Computer Science students in their last (eighth) semester. It is allocated 7 ECTS 1 points; therefore, the expected workload of each student (including contact hours) is between 175 and 210 hours.

The first three weeks serve as a preparatory Sprint (Sprint 0) before the start of the project. During Sprint 0 students are given 12 hours of formal lectures on agile principles, Scrum, and the use of user stories for requirements specification and iteration planning. These three weeks are also used to prepare the development environment and acquaint students with the initial Product Backlog, containing a set of prioritized user stories for the project they are going to develop. Each story contains a short description and a set of acceptance tests that must be used to demonstrate that the story is correctly and fully coded. Students are grouped into teams of four, responsible for the development of the required functionality. Each team estimates the stories using planning poker [19] and prepares the release plan.

The rest of the course is divided into three Sprints, each lasting four weeks. Strictly following the Scrum method, each Sprint starts with a Sprint planning meeting at which student teams negotiate the contents of the next iteration with the Product Owner, and develop the initial version of the Sprint Backlog. During the Sprint the teams have to meet regularly at the Daily Scrum meetings and maintain their Sprint Backlogs, adding new tasks if required and updating data on work spent and work remaining.

At the end of each Sprint, the Sprint review and Sprint retrospective meetings take place. At the review the students present their results to the instructors, while at the retrospective meeting students and instructors meet to review the work done in the previous Sprint, giving suggestions for improvements in the next. After three Sprints the first release should be complete and delivered to the customer.

Since user stories provide only a rough description of required functionality all teams are strongly encouraged to communicate with the Product Owner in order to resolve possible disagreements over the implementation as soon as possible. When a team finishes a story the Product Owner is asked to evaluate its implementation. The Product Owner strictly enforces the concept of “done”, rejecting all stories that do not conform to user requirements. If the shortcomings are not removed by the end of the Sprint a new story must be defined in the Product Backlog requiring the completion of missing features in one of the remaining Sprints.

The course is conducted as a case study; therefore, students are required to provide data on their initial effort estimates, the amount of work spent and the amount of work remaining. At the beginning of each Sprint they are encouraged to re-estimate their velocity and the remaining user stories in order to obtain a more realistic plan for future iterations. Instructors compare students’ plans with actual achievement and analyze whether students’ estimation and planning abilities improve as they gain more knowledge of Scrum and a better understanding of the user requirements. An upgraded version of the Agilo for Scrum project management tool [20] is used to support the collection of all required data.

There is no formal final exam, and the students’ grades are determined on the basis of the amount of Product Backlog accomplished, the quality of software and documentation developed, the fulfillment of release and Sprint plans, and the instructor’s judgment on how well the team worked together, maintained the Sprint Backlog, and kept to schedule.

In the Academic Year 2009/10 the course was attended by 52 students, while in the Academic Year 2010/11 the number of students was 51. Both times 13 student teams were formed that were required to develop a Web-based student records information system covering enrollment, examination applications, examination records, some statistical surveys, and a special module for the maintenance of all data required.

---

1 European Credit Transfer and Accumulation System
for the proper functioning of the system (i.e., the maintenance of various code tables, lists of required and optional courses, data about teachers of each course, etc.). Strictly following Scrum rules, each team acted as a self-organizing and self-managing Scrum team collectively responsible for the implementation of the required functionality.

The initial Product Backlog consisted of 60 user stories. 55 stories described the required functionality for four different user roles (student records administrative staff, students, teachers, and data administrator), whereas five stories described constraints that had to be obeyed (e.g., the system had to enable remote access to data through the Internet, all outputs should also be printable, and so on).

The initial Product Backlog was prepared by a teacher who had considerable experience in developing university information systems, thus being able to provide realistic user requirements and play the role of Product Owner. The stories were divided into four groups on the basis of priority. Among the 55 stories describing the required functionality there were 21 “must have”, 4 “should have”, 3 “could have”, and 27 “won’t have this time” stories. The teams were asked to implement at least all “must have” and as many as possible “should have” stories for the first release. If they planned to be more productive they could also include the “could have” and even some of the “won’t have this time” stories. However, the “won’t have this time” stories were specified merely to illustrate the desired functionality for the next release.

III. SCRUM PRACTICES AS SUCCESS FACTORS

After attending the course students were asked to evaluate how much the proper execution of each Scrum practice contributes to the success of a Scrum project. In order to identify the most important success factors they had to rate the following items:

Quality of requirements specified in the Product Backlog: Scrum requires the maintenance of the Product Backlog. Product Backlog consists of a set of user stories describing the required functionality and constraints of the new system. Each user story must contain a short description and a list of acceptance tests to confirm that it has been developed with the functionality the customer had in mind when he/she wrote the story. The Product Owner is responsible for the contents, prioritization, and availability of the Product Backlog. The quality of Product Backlog can significantly affect the success of a Scrum project.

Good communication with the Product Owner: Scrum requires all details regarding a user story implementation to be clarified through conversations between the Product Owner and developers. At the end of each Sprint, the Product Owner also decides whether a story has been fully implemented or not. It is important that the Product Owner provides timely answers to questions on user stories details, and makes quick evaluations of work being done.

Good ScrumMaster: Scrum prescribes the role of ScrumMaster instead of the traditional project manager. The ScrumMaster is responsible for managing the Scrum process so that it fits within an organization’s culture and still delivers the expected benefits, and for ensuring that everyone follows Scrum rules and practices. He must protect the team by making sure they do not over-commit themselves to what they can achieve during a Sprint, and resolve impediments brought up by the team during Daily Scrum meetings.

Team-work and communication among team members: Scrum requires self-organizing, self-managing, and cross-functional teams that are collectively responsible for implementation of the committed functionality. Therefore, it is important to establish team-work and good communication among team members.

Accurate user story estimations: Scrum requires development teams to estimate the effort required for the implementation of each user story. Stories are usually estimated in story points using the planning poker technique. Estimates are owned by the team and serve for preparation of release and iteration plans. Inaccurate estimations can cause cost overruns, delayed projects, improper resource allocation and/or lost contracts.

Release Planning: Scrum requires the Release Plan to be created by allocating stories to Sprints strictly considering their priority. The total number of story points allocated to each Sprint must not exceed the team’s planned velocity. The Release Plan helps the Product Owner and the whole team decide how much must be developed and provides an estimate of the approximate duration of a project. It serves as a guidepost toward which the project team can progress. Without the concept of a release, teams move endlessly from one iteration to the next [21].

Accurate velocity estimation at the beginning of each Sprint: Scrum requires Scrum teams to estimate their expected velocity at the beginning of each Sprint. Accurate velocity estimates enable Scrum teams to take the right decision about the amount of work they can accomplish. Measuring and monitoring velocity through several iterations improves the accuracy of estimates, thus contributing to better Sprint planning and smooth running of a project.

Sprint Planning Meetings and maintenance of the Sprint Backlog: Scrum requires each Sprint to start with a Sprint planning meeting. At the Sprint planning meeting the Product Owner and team get together to agree upon user stories to be implemented over the next Sprint. The Product Owner presents the highest priority Product Backlog and the team selects how much of what is desired it can implement. Then the team develops the Sprint Backlog, i.e., a list of tasks that must be performed to deliver committed functionality by the end of the Sprint. The tasks in the list emerge as the Sprint evolves and should be divided so that each takes roughly 4 to 16 hours to finish.

Daily Scrum Meetings: Scrum requires that the development team meets every day for a 15-minute Daily Scrum meeting. Daily Scrum meetings serve as a means
of empirical process control in order to assure visibility, inspection, and adaptation. Each team member must report what he has done since the last meeting, what he will do till the next meeting, and what impedes him from performing his work as effectively as possible. Daily Scrum meetings must not be for reporting to the ScrumMaster, but for the team members to inform each other about the current state of the project.

Concept “done”: Scrum requires that at the end of each Sprint the Product Owner accepts only those stories that are potentially shippable. It is preferable to have a small number of completed stories than to have a slightly larger number of stories all incomplete. Strict enforcement of the concept of “done” contributes to the awareness that the code must be fully tested, integrated, and resistant to user errors in order to be used in practice.

Sprint Review Meetings: Scrum requires a Sprint review meeting to be held at the end of each Sprint. At the Sprint review meeting the team presents what was developed during the Sprint to the Product Owner and any other stakeholders who want to attend. This meeting is intended to review the work completed and provide directions for the next Sprint. Through Sprint review meetings customers see on-time delivery of increments, and obtain frequent feedback on how the product actually works.

Sprint Retrospective Meetings: In order to assure continuous improvement of the development process, Scrum requires a Sprint retrospective meeting to take place at the end of each Sprint, i.e., after the Sprint review meeting and prior to the next Sprint planning meeting. At this meeting, the ScrumMaster encourages the team to revise its development process to make it more effective for the next Sprint.

IV. RESULTS

Students rated the aforementioned practices using a 5-point Likert scale. Grade 1 indicated that the practice was not important at all, and grade 5 indicated the highest importance. Grade 3 represented a neutral opinion, indicating that the practice was considered neither important nor unimportant. In total, the authors obtained 94 answers (51 in the Academic Year 2009/10 and 43 in the Academic Year 2010/11).

Since the authors wanted to compare students’ opinions with opinions of professional developers, the same survey was also conducted among 6 developers at the media house working on the project of rebuilding the website of their major daily newspaper. The aim of the project was to introduce completely new design using Drupal content management system [22] as the base for implementation. Developers at the media house were surveyed at the end of the fourth Sprint of their project. Both, students and the media house developers used Scrum for the first time, thus having approximately the same amount of experience with agile methods. The only exception was one of the media house developers who attended the capstone course in the Academic Year 2008/09 and proposed the media house management to use Scrum in their project.

Results of both surveys are presented in Table I (students) and Table II (professional developers).

V. DISCUSSION

Comparison of Tables I and II reveals several similarities, but also some differences between the students and professional developers. Both groups of respondents

- agreed on the two most important success factors, i.e., team-work and communication among team members, and good communication with the Product Owner;
- stressed the high importance of the concept “done”;
- had similar opinions about the role of the ScrumMaster, Sprint review and Sprint retrospective meetings, and Release planning;
- rated accurate user story estimation and accurate velocity estimation least important.

On the other hand, the students and professional developers had different opinions about

- the importance of the clarity of requirements specified in the Product Backlog;
- Sprint planning and Daily Scrum meetings.

A. Two most important success factors

The survey has shown that the two most important factors affecting the success of Scrum projects are: (1) team-work and communication among team members, and (2) good communication with the Product Owner.

TABLE I. STUDENTS’ OPINIONS REGARDING THE IMPORTANCE OF SCRUM PRACTICES (N=93)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Factor</th>
<th>Mean</th>
<th>Std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Team-work and communication among team members</td>
<td>4.80</td>
<td>0.43</td>
</tr>
<tr>
<td>2</td>
<td>Good communication with the Product Owner</td>
<td>4.79</td>
<td>0.44</td>
</tr>
<tr>
<td>3</td>
<td>Concept “done”</td>
<td>4.33</td>
<td>0.88</td>
</tr>
<tr>
<td>4</td>
<td>Clarity of requirements specified in the Product Backlog</td>
<td>4.29</td>
<td>0.68</td>
</tr>
<tr>
<td>5</td>
<td>Sprint Review Meetings</td>
<td>4.24</td>
<td>0.71</td>
</tr>
<tr>
<td>6</td>
<td>Good ScrumMaster</td>
<td>3.99</td>
<td>0.92</td>
</tr>
<tr>
<td>7</td>
<td>Sprint Planning Meetings and maintenance of Sprint Backlog</td>
<td>3.87</td>
<td>0.82</td>
</tr>
<tr>
<td>8</td>
<td>Daily Scrum Meetings</td>
<td>3.83</td>
<td>0.98</td>
</tr>
<tr>
<td>9</td>
<td>Release planning</td>
<td>3.81</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>Sprint Retrospective Meetings</td>
<td>3.76</td>
<td>0.89</td>
</tr>
<tr>
<td>11</td>
<td>Accurate velocity estimation</td>
<td>3.61</td>
<td>0.88</td>
</tr>
<tr>
<td>12</td>
<td>Accurate user story estimation</td>
<td>3.60</td>
<td>0.85</td>
</tr>
</tbody>
</table>
This in line with agile principles that (1) the most efficient and effective method of conveying information to and within a development team is face-to-face conversation, and that (2) business people and developers must work together daily through the project. These principles emphasize synchronous human communication rather than communicating through documents, and require a customer representative to be available in order to provide feedback and to answer the questions of the development team [1]. Authors’ experience with the capstone course [15] has shown that the role of the Product Owner is crucial for the success of a Scrum project. He/she communicates the vision of what is to be developed and defines the criteria by which it will be judged. A nonresponsive Product Owner can cause unproductive work periods, which make iteration planning more difficult or even impossible. Additionally, the Product Owner must be knowledgeable enough about Scrum to be able to write, maintain, and prioritize user stories. On the other hand, the Product Owner must not interfere in the management of teams, redefining the scope or goals of a Sprint, or trying to add new requirements once a Sprint has started.

### B. Concept “done”

The notion of “done” was also rated very high by both groups of respondents, especially students. Although it was their first Scrum project, it seems that they immediately accepted one of the most important agile values, i.e., that the progress of the team is measured in working software. Producing large paper documents and reporting that the code is 80% complete give a false impression of project progress; therefore, the developers should “rely on the honesty that comes with running code” [21]. Only those stories that are brought up to the useful, real-world level which can survive an encounter with users can be considered “done” by the Product Owner and contribute to the velocity calculation.

### C. The role of ScrumMaster

The importance of having a good ScrumMaster was ranked sixth by both groups of respondents obtaining average rates 3.99 and 3.83, respectively. Although still considered important, it seems that this role is less exposed than the role of the Product Owner. Strictly following the Scrum principles of self-organizing and self-managing teams, the ScrumMaster acts merely as a facilitator, thus giving an impression of less importance. He is not the team leader, but acts as a buffer between the team and any distracting influences. A key part of the ScrumMaster’s role is to protect the team and keep them focused on the tasks at hand.

### D. The role of Scrum meetings

Both groups of respondents had similar opinions regarding the importance of Sprint review meetings and Sprint retrospective meetings, but differed quite a lot in their ratings of Sprint planning meetings and Daily Scrum meetings.

Students rated Sprint review meetings to be very important (rank 5, mean 4.24), while the other meetings received average grades between 3.76 and 3.87, which means that these meetings are also considered important, but less than other Scrum practices. While the authors agree that Sprint review meetings are important for reviewing the work completed and providing directions for next Sprints, they attribute a slightly lower grade of other meetings to the fact that Computer Science students prefer technical activities and often perceive meetings as an unproductive waste of time. Nevertheless, it is worth noting that students rated all meetings except Sprint planning higher than professional developers.

Professional developers attributed most importance to Sprint planning meetings, which were rated as high as the good communication with the Product Owner (rank 2-3, mean 4.33). It seems that the professional developers are more aware of the fact that a well prepared and maintained Sprint Backlog is a prerequisite for smooth running of a Sprint. The rates and ranks of Sprint review and Sprint retrospective meetings are similar to those of students, while the rate and rank of Daily Scrum meetings are much lower.

Quite surprisingly, the importance of Daily Scrum meetings was rated higher by students (rank 8, mean 3.83) than by professional developers (rank 10-11, mean 3.17). It seems that students better grasped the fact that Daily Scrum meetings are one of the most important Scrum practices providing continuous insight into project activities and serving as a means for immediate detection and resolution of possible impediments. Nevertheless, both groups of respondents ranked this practice quite low, which is in conflict with the widespread use of Daily Scrum meetings in industrial environment. According to survey [4], daily standup meetings are used by 82% of respondents, thus being the second most used agile technique (iteration planning is used by 83% of respondents).
E. Release planning

In spite of the fact that a release plan provides a context that allows iterations to combine into a satisfying whole, the importance of such a plan was rated rather low especially by the developers working at the media house. The low rate can be partly justified by Cohn’s warning [17] “not to put too much faith in the release plan” since it only represents an approximate duration of a project. Nevertheless, the Scrum users should bear in mind that without a release plan they do not have a clear picture about what is likely to be developed and in what timeframe.

F. Accuracy of estimates

Accurate user story estimation and accurate velocity estimation were rated least important by both groups of respondents. Although agile methods advise not to put too much effort into plans, but to apply an ongoing iterative approach to planning (putting too much effort into plans can lead to the common misbelief that the plans must be right), the authors think that both groups of respondents underestimated the impact of accurate user story and velocity estimates on iteration and release plans.

G. Clarity of requirements specified in the Product Backlog

The students attributed much more importance to the clarity of requirements specified in the Product Backlog than professional developers. During the capstone course many of them were suspicious of the use of user stories for requirements specification finding user stories not precise enough to adequately describe the desired functionality. It seems that the students were less aware of the fact that the essence of the agile approach is not in writing detailed requirements specification, but in acquiring missing details through conversations with the Product Owner and end users.

VI. CONCLUSION

Agile software development methods represent an important teaching challenge. Authors’ experience has shown that the capstone course is an appropriate place for the introduction of these methods. Such a course can provide students with a realistic simulation of professional experience enabling them to better grasp the most important agile concepts. The survey that was conducted among students at the end of Academic Years 2009/2010 and 2010/11 showed that they mastered the most important practices, and that their perceptions are mostly in line with those of professional developers. In some cases (e.g., with regard to importance of Daily Scrum meetings) the authors found their opinions even more appropriate.

The authors agree with students’ opinions that teamwork and communication among team members, as well as good communication with the Product Owner are most important. On the other hand, the survey revealed some practices that were underestimated (e.g., Sprint retrospective meetings, velocity estimation and user stories estimation) and should deserve more attention when teaching forthcoming generations.

REFERENCES