Int. J. Engng Ed. Vol. 26, No. 1, pp. 96–110, 2010
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Teaching Scrum through Team-Project Work: Students' Perceptions and Teacher's Observations*

VILJAN MAHNIC
Faculty of Computer and Information Science, University of Ljubljana, Trzaska 25, 1000 Ljubljana, Slovenia. E-mail: viljan.mahnic@fri.uni-lj.si

In order to prepare students for the increasing use of agile methods in industry, teaching these methods is becoming an important part of the Computer Science and Software Engineering curricula. So far most of the attention has been devoted to Extreme Programming and its practices, but there is not much reported about teaching Scrum, in spite of the fact that Scrum is one of the most widespread agile methods. To fill this gap, a course was developed at the University of Ljubljana that not only teaches Scrum through a capstone project, but also serves as a study about the learnability and applicability of Scrum. This paper describes the course details and analyses students' perceptions and teachers' observations after running the course for the first time in the Spring semester of the Academic Year 2008/09. The student surveys showed that students were overwhelmingly positive about the course and confirmed the anecdotal evidence of Scrum's benefits as reported in the literature.

Keywords: software engineering education; agile methods; Scrum; capstone project

1. INTRODUCTION

AGILE METHODS FOR SOFTWARE DEVELOPMENT [1, 2] emerged as a reaction to documentation driven, heavyweight software development processes that are typical for the traditional disciplined approach advocated by software quality models such as CMMI [3]. According to the *Manifesto for Agile Software Development* [4] these methods 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. Some methods that are inspired by these values include Extreme Programming [5], Scrum [6–8], Adaptive Software Development [9], the Crystal family [10], Feature-Driven Development [11], Dynamic Systems Development Method [12], and Lean Software Development [13].

Experience has shown that adopting agile methods improves management of the development process as well as customer relationships [14], decreases the amount of overtime, and increases customer satisfaction [15]. In the last few years several successful implementations of agile methods in private companies as well as in the public sector have been reported in the literature, e.g., [16–20]. According to the Agile Adoption Rate Survey performed by *Dr. Dobb's Journal* in 2008 [21], agile teams report significant improvements in productivity, quality and stakeholder satisfaction, and reasonable improvements in cost. A similar survey conducted by VersionOne [22] additionally reports significantly improved project visibility and enhanced ability to manage changing priorities. However, the latter survey underlines that the (in)ability to change the organizational culture, the general resistance to change, and the lack of personnel with the necessary agile experience are the greatest barriers to further adoption of agile practices in software development organizations.

In order to prepare students for an increasing use of agile methods in industry, teaching these methods is becoming an important part of the Computer Science and Software Engineering curricula. So far most attention has been devoted to teaching Extreme Programming (XP) [23–26] and its practices, particularly pair programming [27–29] and test-driven development [30–32]. Some authors also report their experience in teaching various combinations of XP practices, e.g., continuous integration and test-driven development [33], test-driven development and refactoring [34], and pair programming, test-driven development and refactoring [35]. With regard to the teaching of other agile methods, there are far fewer reports in the literature. Reed [36] describes an agile classroom experience with the Lean Software Development method, while Van Til *et al.* [37] developed an interdisciplinary elective course to teach lean principles from a broader, manufacturing-oriented perspective.

Most attempts at introducing agile methods are still limited to elective courses and capstone

* Accepted 25 November 2009.

projects that take place after the students have mastered a traditional plan-driven approach to software development. However, due to the increased use of agile methods we can expect that teaching these methods will be incorporated into the core Software Engineering courses, thus giving agile and traditional plan-driven approach equivalent treatment. An example of such a course is given in [38].

Although agile methods are gaining popularity in industry some people still view them as a backlash to software engineering and compare them to hacking [2]. In order to eliminate doubts and provide industry with advice that is grounded in research findings there is an urgent need to empirically assess the applicability of these methods further. Dingsøyr *et al.* [39] claim that the current state of theory and research on particular agile development methods is in the nascent phase according to classification described in [40], while only XP and its pair programming and test-driven development practices might be characterized as intermediate.

In this situation student projects can also serve as case studies providing useful information about the learnability and applicability of particular agile development methods. Students' perceptions of agile methods are analysed in [41], while [42] and [43] describe formal experiments comparing the agile approach with traditional software construction. Dybå and Dingsøyr's systematic review [44] identified 33 empirical studies on agile software development, of which nine (27%) were conducted in a university setting. This review revealed that the studies almost exclusively investigated the XP method. Consequently, one of the clear findings was that the coverage of the research area should be increased, placing more focus on management-oriented approaches such as Scrum, which Dingsøyr *et al.* [39] consider to be an example of an area where there is a large gap and that should be given priority.

In the last few years the use of Scrum has increased significantly. Annual surveys on the state of agile development [22, 45] have shown that Scrum, Scrum/XP Hybrid and XP are the three agile methods that are followed most closely in practice. In 2007 37% of respondents used Scrum, 23% Scrum/XP Hybrid, and 12% XP. In 2008 the share of Scrum increased to 49.1%, Scrum/XP Hybrid was used by 22.3%, and XP by 8.0% of respondents. Therefore, teaching Scrum is becoming an important issue if we want to prepare students for the increasing needs of industry. A knowledge of Scrum is crucial because for most companies the adoption of Scrum is the first step on their way towards agility. Using Scrum they establish a framework into which the XP technical practices can be incorporated easily and more efficiently. This means that Scrum and XP complement each other. Scrum introduces project management practices that ensure transparency, inspection and adaptation, while XP provides engineering practices that cover technical aspects of software development.

Following these facts we have decided to introduce Scrum into the final Software Engineering course that the undergraduate students of Computer Science at the University of Ljubljana take in their last (eighth) semester. The course is designed as a capstone student project with the aim of not only teaching students the Scrum method, but also of contributing to research into Scrum by exploring students' perceptions and find those items that influence the students' satisfaction with work on a Scrum project.

In the next section we describe the aims of the course and research questions that we wanted to answer. We then describe the course details, students' perceptions, and teachers' observations. The balance between the amount of coaching and self-organization is discussed in order to help other educators to evolve and/or apply our method of teaching Scrum.

2. AIMS OF THE COURSE AND RESEARCH QUESTIONS

The course first ran in the Spring semester of the Academic Year 2008/2009 with the following aims:

1. to teach Scrum in a close to real world environment strictly following the method as it is described in [8];
2. to explore the perceptions of Scrum from the students' perspective;
3. to measure the performance of the Scrum-based development process using the metrics model proposed in Mahnic *et al.* [46, 47].

Considering the first aim, the course required students to work in groups to solve a real problem. Since students had mastered traditional methods of software development, fundamentals of data bases and information systems in previous courses, there were only three weeks of formal lectures at the beginning of the course to provide them with the missing knowledge of agile methods and explain the rules of project work. Students were given a choice between two projects: Project A consisted of the development of a project management tool for monitoring the Scrum-based software development process, while Project B was defined in co-operation with a software company and consisted of the development of a simplified version of a hospital information system. A detailed description of both project settings is given in the next section. Our hypothesis was that students will prefer learning through practical project work in agile teams than formal lectures. According to experience and recommendations reported in the literature (e.g., [48–51]) this kind of learning (especially if performed in partnership with industry) contributes significantly to the acquisition of professional skills such as teamwork, planning and organizing, communication,

commitment, co-operation and adaptability. For this reason we also expected the course to be useful for students' employability, allowing them a smooth transition from study to a working environment.

With regard to students' perceptions, our aim was not only to find out how they perceived the course as a whole, but also to analyse their opinions about particular Scrum concepts and identify those Scrum practices that significantly affect satisfaction with the work on a Scrum project. Additionally, we wanted to ascertain how students' opinions match the anecdotal evidence about Scrum benefits reported in the literature. In order to obtain the necessary data for qualitative analysis, several surveys were performed that are described in detail below. With regard to students' perceptions of individual Scrum concepts and Scrum benefits, we posed the null hypothesis that their attitude towards Scrum and its benefits was neutral, but we hoped to find a statistically significant tendency towards the positive side. The paired sample t-test was used to determine how students' opinions about Scrum change with time as they gain more knowledge and practice with the method, while Pearson's product-moment correlation was used to identify similarities between patterns of students' responses to different items and find practices that contribute most to satisfaction with the work on a Scrum project.

With regard to the third aim, we paid special attention to the collection of data on work spent on each task, in addition to the amount of work remaining that was already prescribed by the original method. These data were used by the teacher to compute several process performance indicators (viz. work effectiveness, schedule performance index and cost performance index) as proposed in [46] and [47]. These indicators helped the teacher to monitor progress and react immediately in case of deviations from the plan. They were also presented to students in order to give them rapid feedback about their performance and encourage them to work consistently rather than procrastinate. Our hypothesis was that collecting data for measuring performance does not hinder the agility of the method, but contributes significantly to monitoring the development process.

3. COURSE DETAILS

In order to obtain meaningful results, one of the main challenges of the course was to simulate a real world environment. For this reason two meaningful, practical projects were defined, one of them in co-operation with one of the largest software development companies in Slovenia. The company readily accepted our invitation because they expected the results of the study to be useful in making decisions about the adoption of Scrum in the company. Both projects followed the Scrum process as closely as possible with some minor changes because the students could not work on the project every day due to other academic commitments. Details are given in the subsections below.

In the Academic Year 2008/2009 the course was attended by 31 students who were grouped into seven teams. Four teams worked on Project A, while three teams worked on project B. The amount of work was defined considering the rules of the ECTS. Given the fact that the course is allocated 7 ECTS points, the expected workload of each student (including contact hours) was between 175 and 210 hours.

3.1 Project Settings and Role Playing

Since there was no collocated user representative working permanently with student teams, both projects were chosen from problem domains with which the students were familiar and they could develop the desired functionality without the constant help of the real customer. Nevertheless, the teacher played the role of the customer of Project A and a representative of the co-operating company (a graduate student of Computer Science with experience in building health-care information systems) represented the customer of Project B. The 'customers' attended all lab hours and were available via email to answer students' questions regarding the required functionality of both systems.

In accordance with the principle of self-organization, students were given the opportunity to decide on who they would work with, and which platform and tools to choose for the implementation. According to Scrum rules each team of students played the role of a Scrum Team that was collectively responsible for the success of each iteration and of the project as a whole. The teacher was the ScrumMaster of all teams responsible for the Scrum process and for teaching Scrum to everyone involved in the project. The teacher also played the role of the Product Owner representing the interests of all stakeholders of Project A. Similarly, the representative of the company that specified Project B acted as the Product Owner of Project B.

Both projects were planned to be finished in two iterations (i.e., Sprints in Scrum terminology). The Scrum method in its original form prescribes a Sprint length of 30 calendar days and requires the Team to meet every day for a 15-minutes Daily Scrum meeting. However, because of scheduling problems and other course commitments it was impossible to expect students to work on the project every day. We resolved this problem by prescribing two Daily Scrum meetings per week: one in the presence of the ScrumMaster and the Product Owner took place during the lab hours on Monday, while the other had to be organized by each team independently on Thursday. After each Daily Scrum meeting an updated version of the Sprint Backlog document had to be sent to the teacher. A complete schedule of all Daily Scrum

ID	Product Backlog Item Description	Priority	Initial Estimate	Adjustment Factor	Adjusted Estimate
1-1	Maintenance of projects data (create/update/delete a project description)	1	20	1.50	30
1-2	Maintenance of developers data (create/update/delete developer's details)	1	15	1.00	15
1-3	Maintenance of a Product Backlog (create/update/delete a Product Backlog item)	1	60	2.00	120
1-4	Maintenance of a Sprint Backlog (create/update/delete a task)	1	60	2.00	120
1-5	Maintenance of a metrics table (create/update/delete a metric description)	1	15	1.00	15
1-6	Maintenance of impediments data (create/update/delete an impediment record)	2	15	1.20	18
	Sprint 1		**185**		**318**
2-1	Recording metrics at the beginning of each Sprint (at Sprint planning meeting)	2	40	1.20	48
2-2	Recording metrics collected at Daily Scrum meetings	2	40	1.20	48
2-3	Printing a Product Backlog	3	20	1.50	30
2-4	Printing a Sprint Backlog	3	20	1.00	20
2-5	Displaying work effectiveness and earned value indicators	3	40	1.40	56
2-6	Recording metrics at a Sprint review meeting	4	40	1.10	44
2-7	Recording metrics at a Sprint retrospective meeting	4	30	1.00	30
2-8	Displaying other indicators (fulfillment of scope)	4	30	1.00	30
2-9	Project documentation	5	40	1.00	40
	Sprint 2		**300**		**346**
	Release 1		**485**		**664**

Fig. 1. The initial Product Backlog of Project A.

meetings as well as other meetings (Sprint planning, Sprint review and Sprint retrospective) was prepared in advance and given to the students.

3.2 Product Backlog

At the beginning of the course the Product Owners prepared the initial Product Backlog for both projects containing a list of prioritized requirements divided into two Sprints. The initial Product Backlog for Project A is shown in Figure 1. The rows are the Product Backlog items, separated by the Sprint and Release subheadings. For each Product Backlog item an initial estimate of the work amount (in hours) was provided, multiplied by the adjustment factor taking into consideration the complexity of individual requirements as well as providing room for student teams to adjust the estimated effort to their capabilities and previous experience in practical project work. Beside the initial Product Backlog the students were also given a short description of each requirement and a draft data model (entity–relationship diagram) outlining the database design of each project.

3.3 Maintenance of the Sprint Backlog

Each iteration started with a Sprint planning meeting in order to define the contents of the next Sprint and to develop the initial version of the Sprint Backlog. In the first half of the meeting the Product Owners discussed the requirements and effort estimates with student teams giving them the chance to decide on the subset of the Product Backlog items to be chosen for implementation. After agreeing on the contents of the Sprint, the second half of the meeting was devoted to the development of the Sprint Backlog. It was up to each team to work out how to turn the selected Product Backlog items into an increment of potentially shippable product functionality. Strictly following Scrum rules, the instructors who played the roles of the Product Owners and the Scrum-Master did not interfere, but just observed and answered questions asking for further information. At the end of the Sprint planning meeting each team had to hand the initial version of its Sprint Backlog to the teacher.

During the Sprint the teams had to meet regularly at the Daily Scrum meetings and maintain their Sprint Backlogs, adding new tasks (if required), omitting tasks that proved to be unnecessary, and splitting the tasks that were initially defined too roughly into more detailed ones. For each valid task data about the hours spent and the hours remaining had to be recorded and the updated Sprint Backlog had to be sent to the teacher after each Daily Scrum meeting.

Figure 2 shows an excerpt from the augmented spreadsheet application that was given to students for the maintenance of the Sprint Backlog. For each task a student had to be chosen to be responsible for it and the task status had to be maintained. Classification of tasks according to their type allowed the tracking of the amount of different kinds of work during each Sprint, while 'Hours Spent' and 'Hours Remaining' sections represent the amounts of work spent and work remaining (in hours) to be recorded during each Daily Scrum meeting. Column 0 in the 'Hours Remaining' section corresponds to the initial estimate of each task. Since the tasks in the Sprint Backlog emerge as the Sprint evolves, when each

Task ID	Task Description	Responsible	Task Type	Task Status	Task Valid		Hours Spent							Hours Remaining						
					From	Till														
1	2	3	4	5	6	7	1	2	3	4	5	6	...	0	1	2	3	4	5	6	...
						Total							...							...

Legend

4 – Task Type	*Analysis, Design, Coding, Testing, Documentation, Rework, Other (Specify)*
5 – Task Status	*Not started, In progress, Completed, Omitted, Moved into next Sprint, Split/Divided, Other (Specify)*
6 – Task Valid From	*Day of the Sprint when the task was defined (the consecutive number of the corresponding Daily Scrum meeting)*
7 – Task Valid Till	*Day of the Sprint when the task became not valid (the consecutive number of the corresponding Daily Scrum meeting)*

Fig. 2. Worksheet used for the maintenance of the Sprint Backlog.

task was defined and when it became invalid (if omitted or split into more detailed ones) must be known. Only valid tasks are taken into consideration when performance indicators such as the amount of work remaining, schedule and cost performance indexes are computed.

Considering the principles of self-managing and self-organizing teams advocated by Scrum the instructors did not interfere in the distribution of tasks between team members and the estimation of effort. However, the ScrumMaster strictly monitored the maintenance of the Sprint Backlog and asked all teams to provide data about work spent and work remaining in time.

3.4 Sprint Review and Sprint Retrospective Meetings

The Sprint review and Sprint retrospective meetings took place at the end of each Sprint.

At the Sprint review meetings the students presented the results of their work to the instructors (playing the roles of the Product Owners and the ScrumMaster) and representatives of the cooperating company, who commented on the solutions and asked for changes when necessary. While the main purpose of the Sprint review meeting in Scrum is the demonstration of the new functionality to end users, in our case this meeting also had an educational component: students could compare different solutions and user interfaces for the same requirements and learn from the experience of other teams.

At the Sprint retrospective meetings students and instructors met to review the work in the previous Sprint, giving suggestions for improvements in the next one.

4. STUDENTS' PERCEPTIONS

Students' opinions regarding Scrum and the course as a whole were collected by two questionnaires. The first questionnaire dealt with the use of Scrum in our project (e.g., the adequateness of the Product and the Sprint Backlog, the roles of the Product Owner and the ScrumMaster, etc.) and was used at the end of each Sprint to evaluate the students' experiences for each iteration separately. The second questionnaire served as a final survey at the end of the semester to obtain a general evaluation of the course as a whole as well as for gathering students' opinions about the benefits of Scrum that are usually reported in the literature. In addition, formal discussions about the strengths and weaknesses of Scrum were conducted during the Sprint retrospective meetings that were organized after each Sprint.

4.1 Quantitative Analysis at the End of Each Sprint

The survey conducted at the end of each Sprint contained ten questions on a 5-point Likert scale, grade 1 representing the most negative (strong 'No') and grade 5 the most positive opinion (strong 'Yes'). The survey was anonymous and was answered by 30 out of the 31 students enrolled in the course. One questionnaire that was not filled in completely was excluded from further analysis. In order to test the extent to which student perceptions are consistent the intraclass correlation coefficient (ICC) was computed using the absolute agreement type of the two-way random effects model [52]. The average measure reliability ICC values for Sprint 1 and Sprint 2 were 0.763 and 0.743, respectively. Since by convention the ICC values above 0.7 are considered acceptable [53] we proceeded with the analysis, assuming that the survey data were reliable enough to substantiate appropriate conclusions.

Results of the survey are gathered in Table 1. Columns Sprint 1 and Sprint 2 represent the average grades (the mean), standard deviation, and the results of a one-sample t-test for each Sprint separately. In the last column the results of the paired sample t-test are shown to compare the difference between students' opinions after each Sprint.

The one-sample Student's t-test was used to determine how much students' perceptions deviate from the null hypothesis that their attitude to Scrum was neutral, where the arithmetic mean value of all questions would be equal to 3. Results

Table 1. Results of the survey at the end of each Sprint

		Sprint 1			Sprint 2			
		Mean	Std. dev.	One-sample t-test (t-value)	Mean	Std. dev.	One-sample t-test (t-value)	Paired sample t-test (t-value)
1	**Clarity of requirements specified in the Product Backlog** Was the Product Backlog clear enough? Did the short description of each requirement suffice to understand what the Product Owner really wanted?	3.21	0.98	1.14	3.87	0.90	5.28**	3.91**
2	**Effort estimation** Were the estimates of workload provided in the Product Backlog adequate?	2.97	0.98	–0.19	3.80	0.81	5.44**	4.25**
3	**Maintenance of the Sprint Backlog** Was it clear how to maintain the Sprint Backlog and provide data requested by the spreadsheet application?	3.72	0.75	5.19**	4.30	0.79	8.96**	4.31**
4	**Administrative workload** The administrative work requested by the Scrum method does not represent a significant additional workload.	3.24	1.15	1.13	3.27	1.08	1.35	0.49
5	**Co-operation with the ScrumMaster** Was the co-operation with the ScrumMaster adequate?	4.07	0.92	6.24**	4.33	0.71	10.27**	2.20*
6	**Co-operation with the Product Owner** Was the co-operation with the Product Owner adequate?	3.79	0.90	4.74**	4.03	0.85	6.66**	1.44
7	**Co-operation with other Team members** Was the co-operation with other Team members adequate? Does the Scrum method encourage co-operation?	4.00	1.04	5.20**	4.10	0.92	6.53**	0.46
8	**Workload** Was the amount of work required on the project adequate?	3.76	0.99	4.14**	3.73	0.78	5.12**	0.00
9	**Satisfaction with the work on the project** Are you satisfied with the work on the project?	3.72	0.80	4.89**	3.80	0.92	4.74**	0.96
10	**Satisfaction with the Scrum Method** Is the method useful? Would you recommend the method to other software developers?	3.72	1.07	3.66*	3.87	0.94	5.07**	0.50

* $p < 0.05$; ** $p < 0.01$.

in Table 1 show that 7 out of 10 hypotheses were rejected after Sprint 1, and 9 out of 10 hypotheses were rejected after Sprint 2. In all these cases the student grades were significantly higher than 3; therefore, we can accept the alternative hypothesis that students on average had a positive opinion about the Scrum method and its use in our course.

The paired sample Student's t-test enabled us to analyse the before–after effect of the Scrum method by comparing the average grades after Sprint 1 and Sprint 2. All grades except one increased after the second Sprint, four of them (see questions 1, 2, 3 and 5) with a statistically significant difference. This shows that the students' attitude towards Scrum improved as the students gained more experience.

Students' opinions were most positive with regard to questions 5, 6 and 7 concerning co-operation between all parties involved in the project. We were gratified that the co-operation with the ScrumMaster received the highest grade (see question 5). On the one hand it indicates that the course was well prepared and conducted, and on the other it shows the importance of good coaching when agile methods are introduced to novices. Students were also very satisfied with the co-operation with their team-mates, thus clearly indicating that Scrum contributes to successful teamwork (question 7). We believe that this grade would be still higher if we did not have problems with two non-participatory students. The co-operation with the Product Owner also received a high grade (question 6).

At the beginning students had some questions about the maintenance of the Sprint Backlog and about providing metrics required by the spreadsheet application, but the initial problems were quickly resolved and the grades (although not bad in the first Sprint) rose significantly in the second Sprint (question 3). Nevertheless, the

answers to question 4 show only a weak agreement with our opinion that the administrative work required by the Scrum method does not represent a significant additional workload. This question was further discussed during the Sprint retrospective meetings.

Since the students were not given a detailed requirements specification, but just the Product Backlog containing a prioritized list of requirements and a short textual description of each requirement, we were afraid that they would have problems with understanding what the Product Owner really wanted (question 1). This was partly true during the first Sprint (although the average grade is still on the positive side of the Likert scale), but the situation improved significantly during the second Sprint through better communication between the students and the Product Owner. This shows the importance of a customer representative being a member of the development team.

After the first Sprint the students had a neutral opinion regarding the adequacy of the initial effort estimation provided in the Product Backlog (question 2). It is interesting to note that during the Sprint some student teams decreased the initial estimates and reported their tasks to be completed in less time than planned at the beginning. However, the Sprint review meeting revealed that their solutions were merely prototypes that worked properly if the user entered correct data, but failed in the case of mistakes. After explaining again that each application should be fully tested and resistant to misuse before being considered 'done' the attitude of these students towards effort estimation changed as indicated by the increased grade after Sprint 2.

The answers to question 8 are also positive and show that the majority of students agreed with the amount of work required. However, this is the only question with a lower grade after the second Sprint. We think that this is a consequence of the fact that (considering the experience from the first Sprint described in previous paragraph) we paid more attention to strict testing and usability of the code developed, thus giving students the feeling of an increased workload. Nevertheless, the difference between the two grades is too small to be significant.

The answers to the last two questions show that the majority of students was satisfied with the work on the project and the use of Scrum. Most of them found the method useful and would recommend it to others. This is in line with the findings of some other studies, e.g., [36, 41] describing the students' enthusiasm with regard to agile methods.

Pearson's product-moment correlation was used to identify similarities among patterns of responses to different questions. Table 2 shows a significant positive correlation between question 9 and questions 1, 3, 4, 5 and 6, leading to the conclusion that the overall satisfaction with the work on a Scrum project depends on the clarity of the requirements specified in the Product Backlog, clear rules for the maintenance of the Sprint Backlog, appropriate administrative workload, and good co-operation with the ScrumMaster and the Product Owner. The positive correlation between questions 3 and 5 also indicates that the ScrumMaster plays an important role in facilitating the maintenance of the Sprint Backlog. The high correlation between questions 5 and 6 must be interpreted cautiously since the roles of ScrumMaster and Product Owner were played by the same person on project A. Nevertheless, a good ScrumMaster can contribute a great deal to improving the co-operation between the Product Owner and the development team. Finally, the positive correlation between questions 1 and 3 shows that a clear specification of requirements in the Product Backlog is an important prerequisite for efficient maintenance of the Sprint Backlog.

Surprisingly, there is a significant negative correlation between questions 7 and 8 that is difficult to explain. Detailed analysis of students' comments concerning these two questions revealed some extreme cases. On the one hand, there were some students who rated their co-operation with other team members excellent, but meant that the final polishing of application details in order to produce a shippable code free of bugs required too much work. On the other hand there was a student who rated workload excellent, but was very unsatisfied with the teamwork due to a slacker in his team.

Table 2. Pearson correlations (r)

		1	2	3	4	5	6	7	8	9	10
1	Requirements	1.00	0.30	0.44*	0.18	0.29	0.10	–0.11	0.29	0.59**	0.06
2	Effort estimation		1.00	0.26	0.10	0.24	0.06	0.31	–0.09	0.18	0.06
3	Sprint Backlog			1.00	0.15	0.49**	0.29	–0.14	0.24	0.41*	0.33
4	Administration				1.00	0.24	0.22	0.32	–0.16	0.50**	0.24
5	ScrumMaster					1.00	0.78**	0.32	–0.02	0.63**	–0.09
6	Product Owner						1.00	0.26	–0.09	0.49**	0.05
7	Teamwork							1.00	–0.58**	0.11	–0.14
8	Workload								1.00	0.11	0.00
9	Satisfaction									1.00	0.17
10	Scrum Method										1.00

* Correlation is significant at the 0.05 level (2-tailed).
** Correlation is significant at the 0.01 level (2-tailed).

Table 3. Results of the final survey at the end of the course (Part I)

1	**Do you support the decision to run the course as a capstone project providing students with necessary knowledge through solving a practical problem in a semi-industrial environment?**		
	a) yes	30	100.0%
	b) no	0	0.0%
2	**How useful is the course?**		
	a) the course is useful and interesting	21	70.0%
	b) the course is useful	5	16.7%
	c) the course is not useful	3	10.0%
	d) the course is not useful and uninteresting	1	3.3%
3	**How do you rate the course in comparison to other courses in the final year of your study?**		
	a) better	14	46.7%
	b) approximately the same	13	43.3%
	c) worse	1	3.3%
	d) no answer	2	6.7%
4	**How do you rate the objectivity of examination?**		
	a) very bad	1	3.3%
	b) bad	0	0.0%
	c) average	2	6.7%
	d) very good	8	26.7%
	e) excellent	19	63.3%
5	**Does the course contribute to your employability and professional career?**		
	a) yes	27	90.0%
	b) no	3	10.0%

Unfortunately, we could not find a significant correlation between question 10 and other questions. It seems that students' satisfaction with Scrum depends mostly on the ease of the Sprint Backlog maintenance ($r = 0.33$) and the amount of additional administrative workload ($r = 0.24$).

4.2 Final Survey at the End of the Course

The final survey at the end of the course was also anonymous and was answered by all students except one ($N = 30$). It consisted of two parts. Results of the first part concerning the general evaluation of the course are presented in Table 3, while the results of the second part dealing with students' opinions regarding the benefits of Scrum are gathered in Table 4.

4.2.1 General evaluation of the course

General evaluation of the course clearly confirmed our hypothesis that students prefer learning Scrum through practical project work than formal lectures. The answers to question 1 show a unanimous support for the decision to run the course as a capstone project enabling them to learn by solving an almost-real problem. Computer Science curricula at universities in Slovenia usually lack such courses, therefore a comment of one of the students that this was the only real problem that he had worked on during his studies was not a surprise. Accordingly, the great majority of students (86.67%) found the course useful or useful and interesting (question 2). The answers to question 3 also show that students were satisfied with the course. Almost half of them felt that the course was better than other courses in the final year of their study and only one of them rated the course as worse.

There was no formal exam, and students' grades were obtained taking into consideration the amount of Product Backlog accomplished, the quality of software and documentation developed as well as the instructors' judgment on how well the team worked together, maintained the Sprint Backlog and kept to schedule during both Sprints. Answers to question 4 show that the great majority considered the grades that they obtained to be fair

Table 4. Results of the final survey at the end of the course (Part II)

		Mean	Std. dev.	One-sample t-test (t-value)
1	The product becomes a series of manageable chunks.	4.33	0.88	7.90**
2	Progress is made, even when requirements are not stable.	3.19	0.88	1.10
3	Everything is visible to everyone.	4.19	0.74	8.37**
4	Team communication improves.	4.04	0.90	6.00**
5	The Team shares successes along the way and at the end.	4.59	0.57	14.46**
6	Customers see on-time delivery of increments.	4.07	0.78	7.15**
7	Customers obtain frequent feedback on how the product actually works.	3.93	1.00	4.83**
8	A relationship with the customer develops, trust builds, and knowledge grows.	4.19	0.79	7.83**
9	A culture is created where everyone expects the project to succeed.	4.15	0.86	6.91**

** $p < 0.01$.

and correct. Most of students also found the course beneficial to their employability and professional career (question 5).

4.2.2 Students' opinions regarding Scrum benefits

The second part of survey consisted of nine assertions regarding Scrum benefits as reported in [6]. For each assertion students were asked to specify how much they agree (or disagree) using a 5-point Likert scale (1 'Strongly Disagree' to 5 'Strongly Agree'). Three questionnaires that were not answered completely were removed from analysis. The consistency of students' responses was checked by computing the average measure ICC using the absolute agreement two-way random effects model. The ICC value of 0.829 indicates that the survey data were reliable enough to substantiate appropriate conclusions.

The one-sample t-test was used again to verify the null hypothesis that student opinions are neutral regarding the cited benefits. Results in Table 4 show a statistically significant positive level of agreement with all assertions except assertion 2, thus rejecting 8 null hypotheses out of 9.

Seven assertions obtained an average grade greater than 4. Students strongly agreed that Scrum enables teams to share successes throughout the project (assertion 5), helps to split the product into a series of manageable chunks (assertion 1), makes everything visible to everyone (assertion 3), improves customer relationships (assertion 8), creates a culture where everyone expects the project to succeed (assertion 9), and enables customers to see on-time delivery of increments (assertion 6).

There was also an evident agreement with assertions 4 and 7 concerning improved team communication and customer feedback on how the product really works.

It seems that the only assertion students were not sure about was the progress in the case of unstable requirements, although this grade is slightly positive, too. We think that this is a consequence of the fact that the course consisted of only two Sprints and (due to limited course duration) we could not simulate a quickly changing environment to provide students with the necessary experience.

4.3 Sprint Retrospective Meetings

In addition to surveys, the Sprint retrospective meetings were an excellent opportunity to discuss students' opinions and gather their suggestions for improvement. We organized a formal Sprint retrospective meeting at the end of each Sprint (and before the beginning of the next one) as prescribed by the Scrum method. At the meeting each Team was asked to answer two questions: (1) What went well during the last Sprint; and (2) What could be improved in the next Sprint. Potential improvements were prioritized and discussed in order to make the development process more effective and enjoyable for the next Sprint.

At the first retrospective meeting students reported a positive experience with Scrum, encouraging them to begin work on the project early and to work consistently rather than procrastinate. Additionally, they found Scrum helpful in establishing good communication within their teams. On the other hand they felt the need to improve the definition of tasks in the Sprint Backlog and their allocation to individual team members in a way that would enable uniform distribution of the workload. Experience from the first Sprint showed that some teams did not pay enough attention to testing and integration; therefore, all teams were strongly advised to include the corresponding tasks into their Sprint Backlogs. The next improvement referred to better communication between the teams and both Product Owners in order to provide students with timely answers to their questions regarding Product Backlog requirements.

At the second retrospective meeting, students reported that the improved knowledge of Scrum (as well as of the technology they used) helped them to organize their work better and progress more quickly. They paid more attention to testing and integration, but mostly planned these tasks to take place at the end of the Sprint. Therefore, an improvement was suggested that the students test and integrate throughout the Sprint as often as possible. A collocated customer representative should also be available to execute acceptance tests instead of having to wait for customers' remarks until the Sprint review meeting.

The issue of overhead was discussed at both meetings since the question of administrative workload caused by the maintenance of the Sprint Backlog and recording work spent was rated worse than others (although still positive). We believe that students see this work as unproductive, but think it is much lower than traditional heavy methods of software development. Representatives of the participating company that also attended the meetings helped us to convince the students that most companies request employees to report on the amount of work actually spent on different tasks, and that successful project management is impossible without a clear definition of the tasks and their distribution between team members. Therefore, some additional administrative work is inevitably present in all real projects.

5. TEACHER'S OBSERVATIONS

It had already become evident during the course that the majority of students are enthusiastic about the practical approach used in the course and the use of Scrum. They provided the data required for monitoring performance on time and—except for two freeloaders—tried to do their best to contribute to the success of their teams. Nevertheless, the course also presented some dilemmas regarding the amount of coaching needed in order to improve the final result of the students' projects.

5.1 Monitoring Performance

According to the third aim of the course a great deal of the teacher's attention was devoted to performance measurement. Although the results of the survey (see question 4 in Table 1) did not confirm our hypothesis that the administrative work required by the Scrum method does not represent a significant additional workload, collecting data on the hours spent and hours remaining for each task in the Sprint Backlog proved to be useful for the computation of different performance indicators. In addition to the burn-down chart proposed by Scrum and representing the total amount of work remaining on each day of the Sprint, the computation of the earned value method (EVM) [54] schedule and cost performance indexes (SPI and CPI) was introduced in order to obtain a complete insight into project performance. While other studies dealing with the use of EVM within Scrum (e.g., [55, 56]) describe the computation of earned value at the release level, we introduced the computation of EVM indexes at the Sprint level as proposed in [46, 47]. This approach provided the values of SPI and CPI on a daily basis, thus enabling an immediate response in the case of deviation from the plan. An analysis of the burn-down, SPI and CPI charts of different teams gave the course an additional research component that rendered it more interesting.

Figures 3, 4 and 5 represent an example of performance monitoring in Sprint 2 for one of

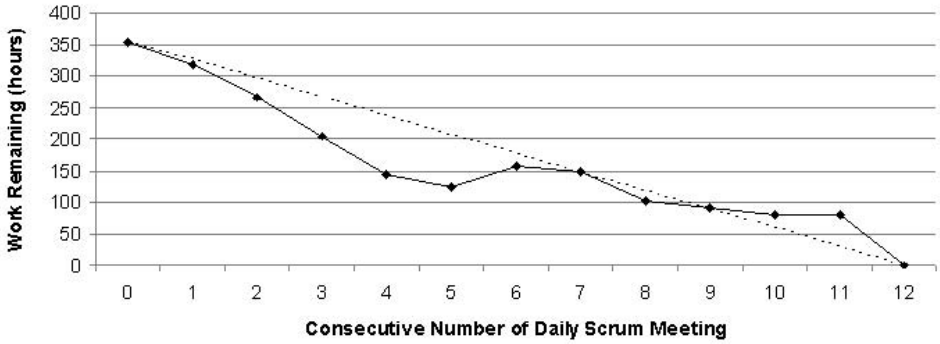

Fig. 3. Sprint Burn-down Chart (Team #3, Sprint #2) representing the total amount of work remaining (in hours).

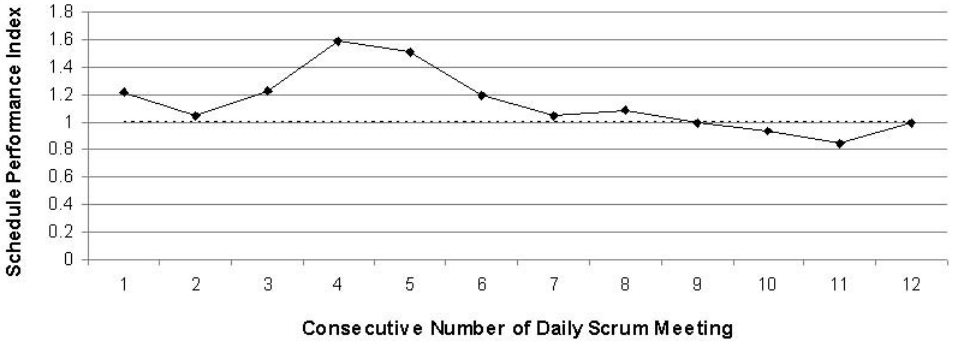

Fig. 4. Schedule Performance Index (Team 3, Sprint 2).

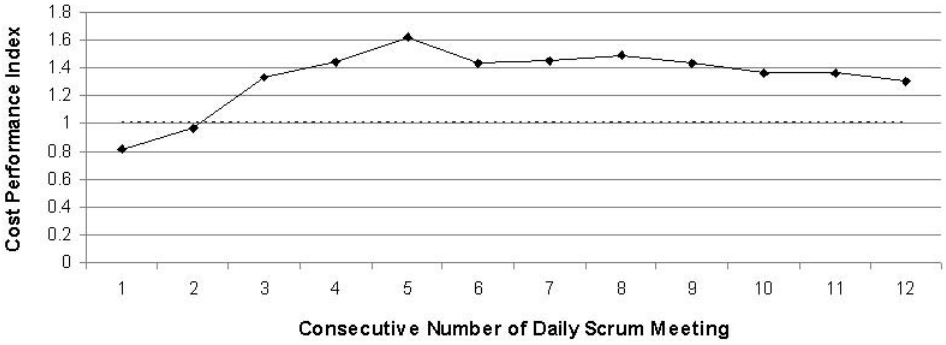

Fig. 5. Cost Performance Index (Team 3, Sprint 2).

the student teams. The total amount of work remaining, shown in Figure 3, and the earned value SPI and CPI indexes, shown in Figures 4 and 5, were computed after each Daily Scrum meeting, which took place twice a week as described in Section 2. Therefore, the horizontal axis represents the consecutive numbers of the Daily Scrum meetings and the value 0 on the horizontal axis of Figure 3 corresponds to the initial state after the Sprint planning meeting. In a real Scrum project the aforementioned values should be computed every day and the horizontal axis should represent the number of days that had elapsed since the beginning of the Sprint.

The computation of the SPI and CPI proved to be simple and required only one additional metric: the number of hours spent on each task recorded in the Sprint Backlog after each Daily Scrum meeting. In our calculations we assumed the hourly rate to be equal for all students. However, in practice, the cost of an engineering hour should be maintained for each developer (or each task type) separately.

Although the burn-down chart was a valuable tool for monitoring performance it did not provide information about the effort spent and work effectiveness. For this reason the use of additional indicators was beneficial if we wanted to have full insight into the project performance. While the SPI provided similar information to the burn-down chart, the CPI enabled the actual labour cost to be compared with the plan, considering the values of hours spent and hours remaining maintained in the Sprint Backlog.

These graphs helped us to monitor the progress of the student teams more closely and to discover potential problems early. For example, the Sprint Burn-down Chart in Figure 3 clearly shows that at the beginning of the Sprint the amount of work remaining (as reported by the team) diminished faster than planned, a situation quite opposite to the late project experiences usually reported in the literature. The same effect was also seen from the SPI graph in Figure 4, showing unusually high values of SPI. Examining this situation we found out that the team members declared some tasks to be done without thorough testing and without assuring that the functionality developed met the end user needs. In order to solve this problem the team had to allocate additional time for reworking and testing, causing an increase in the amount of work remaining and a decrease of the SPI after the Daily Scrum #5. In this way, the early discovery of a problem helped the team to make corrective actions in time and to complete the Sprint successfully.

On the other hand the CPI remained high (see Figure 5) even after the allocation of additional time, indicating that the team needed less money to complete the tasks than planned at the beginning of the Sprint. This could be a sign of the high productivity of the team (this team was one of the best), but also a sign of poor planning. In fact, the course revealed that estimating the amount of time needed for individual tasks is not easy and needs more experience than students have. In some cases the high value of CPI was also a consequence of the 'releasing before done' effect described in the next section.

5.2 Balancing Coaching and Self-Organization

In spite of the fact that student surveys clearly indicated that the course was successful, the evaluation of the students' projects presented some questions regarding the quality of their work. Only two teams out of seven produced code that was really shippable and could be (with some minor improvements) used in practice. Three teams developed solutions that formally offered the required functionality, but lacked robustness, reliability and professional polish. The other two teams developed individual functions, but did not succeed in integrating them completely into a workable solution.

The quality of the students' projects could be improved by stricter coaching, but in this case we would sacrifice the principle of self-organization advocated by Scrum. Therefore, an important issue in designing such a course is the maintenance of balance between strict coaching and self-organization. In our case the first aim was to follow the Scrum method strictly since the course was also used to study the students' perceptions about Scrum and the computation of EVM indexes. For this reason the principle of self-organization was given priority. It was up to student teams to decide about the roles of each team member and the methods and tools used for the implementation of the requested functionality. Consequently, the roles within the teams were not always adequately assigned and the requirement specifications were seldom elaborated further in discussions with the Product Owner (who also played the role of the end customer). Additionally, the first Sprint review meeting revealed that some teams did not fully understand the meaning of task being 'done', but sometimes declared the tasks to be completed without thorough testing and checking compliance with end user needs. This effect of 'releasing before done' meant that the amount of work remaining as reported by some teams did not reflect reality, but was decreasing too quickly, thus giving a false impression of the project being ahead of schedule and the team being more efficient than expected.

This experience should be considered by those designing similar courses in the future. The teacher should adapt the balance between coaching and self-organization to the aims of their course. While more coaching enables a higher quality of final product, self-organization provides experiential learning through discovery and participation rather than following instructions. We tried to resolve these dilemmas by a means provided by Scrum, i.e., through Sprint retrospective meetings. At these meetings the ScrumMaster and the Product Owner played a more active role than prescribed by Scrum, providing (in addition to

students' opinions) their views of what was good and what went wrong in the previous Sprint and suggesting improvements for the next one. Teams that succeeded in adopting the suggested improvements obtained better results.

Our plan is to increase the amount of coaching in the future. In spite of good results of the survey regarding the clarity of the Product Backlog, we plan to pay more attention to requirements specifications. We expect that the more rigorous introduction of user stories (especially acceptance criteria) and story point estimates will enhance the understanding of requirements, improve planning, and cut the number of changes requested by end users at the Sprint review meetings. Additionally, students will be requested to clearly assign team roles in order to achieve cross-functionality and not underestimate the roles of tester and collocating representative of end user.

6. SUMMARY OF FINDINGS, POSSIBLE GENERALIZATIONS AND LIMITATIONS OF THE STUDY

Survey results have confirmed our initial hypothesis that students prefer learning through practical project work to that of formal lectures. Detailed statistical analysis after Sprint 2 has also shown a significant positive attitude regarding all aspects of our course except administrative workload. It is important that the students' opinions improved as they gained more knowledge about Scrum. Additionally, a high level of agreement with anecdotal evidence about the benefits of Scrum reported in the literature has been found, confirming 8 assertions out of 9.

Cross-correlation of survey items has revealed that the satisfaction with the work on a Scrum project is strongly correlated with the quality of requirements specified in the Product Backlog, clear rules for the maintenance of the Sprint Backlog, appropriate administrative workload, and good co-operation with the ScrumMaster and the Product Owner. Collecting data about work spent has proved beneficial in monitoring progress not only through the amount of work remaining, but also by measuring earned value in terms of schedule and cost. However, there has not been statistically significant agreement with our hypothesis that collecting data on work spent does not hinder the agility of the method.

These results can be generalized in two ways: from the standpoint of teaching a similar course and from that of using Scrum in industry. From the standpoint of teaching Scrum it is evident that students prefer the practical approach that enables learning through project work. Therefore, the results concerning the students' perceptions of Scrum are valid under the assumption that the course is delivered in a similar way and the instructors pay enough attention to the aforementioned items that affect satisfaction with the work on a Scrum project. From our point of view, the roles of the ScrumMaster and the Product Owner are crucial to the success of such a course. Playing these roles requires much effort on the part of instructors as they must not only monitor the progress, but on many occasions also collaborate with team members contributing to the success of the project. There is no guarantee that the same results will be obtained if the course is delivered as formal lectures.

From the standpoint of using Scrum in industry the main limitation of our study is that it was conducted with students in an academic environment. However, in order to increase the degree of validity we made every effort to simulate an industrial environment as closely as possible. Students worked on almost-real projects (one of them was defined in co-operation with a software company) and followed the Scrum method as strictly as their other academic duties allowed. The study relied on senior students enrolled in their last semester, many of them having some professional experience, thus blurring the line between these students and novice professionals. Berander [57] identified that in a project setting where the students have made a true commitment, students tend to act and think more like professionals. A recent study by Svahnberg *et al.* [58] has also shown that it may be possible to influence students to provide answers that are in line with industrial practice. For all these reasons we argue that the results of our study are applicable to situations (currently very frequent) when companies start introducing Scrum into their development process, but do not have enough developers with adequate knowledge of Scrum.

7. CONCLUSIONS

In the Spring semester of the Academic Year 2008/09 the final Software Engineering course at the University of Ljubljana was redesigned in order to expose students to agile methods, particularly Scrum. Scrum was chosen because it is one of the most widespread agile methods, but lacks more detailed evaluation and empirical evidence about its applicability. For this reason, in addition to offering a new up-to-date content the course also had a strong research component, serving as a case study for evaluation of students' perceptions and introducing measurement of earned value at Sprint level.

In order to provide students with an almost-real environment the course was designed as a capstone project with only a small amount of formal lectures needed to introduce agile concepts and Scrum. The majority of the course was dedicated to project work during which student teams developed real projects strictly following the Scrum method. By choosing relevant projects from real life the course not only provided students with a knowledge of Scrum, but also with professional skills needed in

an industry environment. One of the projects was defined in co-operation with a software company that provided Product Backlog with requirement specifications.

Experience after teaching the course for the first time has shown that the course achieved all three objectives.

1. Students enjoyed learning Scrum in a close to real world environment.
2. Their perceptions about Scrum were positive, and the empirical evaluation based on surveys conducted after each Sprint and at the end of the course confirmed the anecdotal evidence about the strengths of Scrum reported in the literature.
3. Data from Sprint Backlogs collected during the course enabled the computation of earned value schedule and cost performance indexes as proposed in some of our previous works. Computing SPI and CPI on a daily basis complemented the Sprint burn-down charts and enabled an immediate reaction in case of deviations from the plan.

Experience has also shown that a balance between coaching and self-organization is important when teaching Scrum to beginners. While self-organization is useful from the pedagogical point of view, providing experiential learning through discovery and participation, strict coaching is important to achieve a quality end product. Since our course closely followed Scrum rules we gave priority to self-organization trying to improve the development process through discussions at Sprint retrospective meetings. However, in the future we also plan to increase the amount of coaching, paying more attention to the precise specification of requirements and the assignment of appropriate roles within student teams.

REFERENCES

1. P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, *Agile Software Development Methods,* VTT Electronic, Espoo, 2002.
2. D. Cohen, M. Lindvall and P. Costa, An introduction to agile methods, *Advances in Computers*, **62**, 2004, 2–67.
3. CMMI, CMMI® for Development, Version 1.2. CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, 2006.
4. Manifesto for Agile Software Development, http://www.agilemanifesto.org/, 2001 (viewed 11.8.2009).
5. K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2000.
6. L. Rising and N. S. Janoff, The Scrum software development process for small teams, *IEEE Software*, **17**(4), 2000, pp. 26–32.
7. K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, 2002.
8. K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, Redmond, 2004.
9. J. A. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publisher, New York, 2000.
10. A. Cockburn, *Agile Software Development*, Addison-Wesley, Boston, 2002.
11. S. R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*, Prentice-Hall, Upper Saddle River, 2002.
12. J. Stapleton, DSDM, *Dynamic Systems Development Method: the Method in Practice*, Addison-Wesley, Reading, 1997.
13. M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*, Addison-Wesley, Boston, 2003.
14. M. Ceschi, A. Sillitti, G. Succi and S. De Panfilis, Project management in plan-based and agile companies, *IEEE Software*, **22**(3), 2005, pp. 21–27.
15. C. Mann and F. Maurer, A case study on the impact of Scrum on overtime and customer satisfaction, *Proceedings of the Agile Development Conference (ADC'05)*, 2005, pp. 70–79.
16. B. Schatz and I. Abdelshafi, Primavera gets agile: A successful transition to agile development, *IEEE Software*, **22**(3), 2005, pp. 36–42.
17. V. Mahnic and S. Drnovscek, Introducing agile methods in the development of university information systems, *Proceedings of the 12th International Conference of European University Information Systems EUNIS 2006*, Tartu, Estonia, 2006, pp. 61–68.
18. J. Fecarotta, MyBoeingFleet and agile software development, *Proceedings of the Agile 2008 Conference*, 2008, pp. 135–139.
19. K. Scotland and A. Boutin, Integrating scrum with the process framework at Yahoo! Europe, *Proceedings of the Agile 2008 Conference*, 2008, pp. 191–195.
20. J. Scott, R. Johnson and M. McCullough, Executing agile in a structured organization: government, *Proceedings of the Agile 2008 Conference*, 2008, pp. 166–170.
21. S. W. Ambler, Has agile peaked? Let's look at the numbers, *Dr. Dobb's Journal*, http://www.ddj.com/architect/207600615?pgno=1, 2008 (viewed 11.8.2009).
22. VersionOne, 3rd Annual Survey: 2008, The state of agile development, Full Data Report, http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf, 2008 (viewed 11.8.2009).
23. M. M. Müller and W. F. Tichy, Case study: extreme programming in a university environment, *Proceedings of the 23rd International Conference on Software Engineering (ICSE'01)*, 2001.
24. A. Shukla and L. Williams, Adapting extreme programming for a core software engineering course, *Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02)*, 2002.

25. O. Hazzan and Y. Dubinsky, Teaching a software development methodology: the case of extreme programming, *Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03)*, 2003.
26. Y. Dubinsky and O. Hazzan, eXtreme programming as a framework for student-project coaching in computer science capstone courses, *Proceedings of the IEEE International Conference on Software-Science, Technology & Engineering (SwSTE'03)*, 2003.
27. H. Srikanth, L. Williams, E. Wiebe, C. Miller and S. Balik, On pair rotation in computer science course, *Proceedings of the 17th Conference on Software Engineering Education and Training (CSEET'04)*, 2004.
28. L. Williams, L. Layman, J. Osborne and N. Katira, Examining the compatibility of student pair programmers, *Proceedings of the Agile 2006 Conference*, 2006, pp. 411–420.
29. L. Williams, Lessons learned from seven years of pair programming at North Carolina State University, *ACM SIGCSE Bulletin*, **39**(4), 2007, pp. 79–83.
30. P. J. Schroeder and D. Rothe, Teaching unit testing using test-driven development, *Workshop on Teaching Software Testing 2005*, http://www.testingeducation.org/conference/wtst4/pjs_wtst4.pdf, 2005 (viewed 11.8.2009).
31. A. Tinkham and C. Kaner, Experiences teaching a course in programmer testing, *Proceedings of the Agile Development Conference (ACD'05)*, 2005, pp. 298–305.
32. D. S. Janzen and H. Saiedian, Test-driven learning: intrinsic integration of testing into the CS/SE curriculum, *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'06)*, 2006, pp. 254–258.
33. J. Bowyer and J Hughes, Assessing undergraduate experience of continuous integration and test-driven development, *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, 2006, pp. 691–694.
34. B. Carlson, An agile classroom experience: teaching TDD and refactoring, *Proceedings of the Agile 2008 Conference*, 2008, pp. 465–469.
35. S. Xu and V. Rajlich, Empirical validation of test-driven pair programming in game development, *Proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering*, Software Architecture and Reuse (ICIS-COMSAR'06), 2006, pp. 500–505.
36. P. Reed, An Agile classroom experience, *Proceedings of the Agile 2008 Conference*, 2008, pp. 478–483.
37. R. P. van Til, M. W. Tracey, S. Sengupta and G. Fliedner, Teaching lean with an interdisciplinary problem-solving learning approach, *International Journal of Engineering Education*, **25**(1), 173–180, 2009.
38. L. Layman, L. Williams, K. Slaten, S. Berenson and M. Vouk, Addressing diverse needs through a balance of agile and plan-driven software development methodologies in the core software engineering course, *International Journal of Engineering Education*, **24**(4), 2008, pp. 659–670.
39. T. Dingsøyr, T. Dybå and P. Abrahamsson, A preliminary roadmap for empirical research on agile software development, *Proceedings of the Agile 2008 Conference*, 2008, pp. 83–94.
40. A. C. Edmondson and S.E. McManus, Methodological fit in management field research, *Academy of Management Review*, **32**(4), 2007, pp. 1155–1179.
41. G. Melnik and F. Maurer, A cross-program investigation of students' perceptions of agile methods, *Proceedings of the 27th International Conference on Software Engineering (ICSE'05)*, 2005, pp. 481–487.
42. F. Macias, M. Holcombe and M. Gheorghe, A formal experiment comparing extreme programming with traditional software construction, *Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)*, 2003.
43. P. N. Robillard and M. Dulipovici, Teaching agile versus disciplined processes, *International Journal of Engineering Education*, **24**(4), 2008, pp. 671–680.
44. T. Dybå and T. Dingsøyr, Empirical studies of agile software development: A systematic review, *Information and Software Technology*, **50**, 2008, pp. 833–859.
45. VersionOne, *2nd Annual Survey 'The State of Agile Development', Survey Highlights & Full Data Report*, http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf, 2007 (viewed 16.11.2009).
46. V. Mahnic and I. Vrana, Using stakeholder driven process performance measurement for monitoring the performance of a Scrum based software development process, *Electrotechnical Review*, Ljubljana, **74**(5), 2007, pp. 241–247.
47. V. Mahnic and N. Zabkar, Introducing CMMI measurement and analysis practices into scrum-based software development process, *International Journal of Mathematics and Computers in Simulation*, **1**(1), 2007, pp. 65–72.
48. S. Smith, M. Mannion and C. Hastie, Encouraging the development of transferable skills through effective group project work, *Software Engineering in Higher Education II*, edited by J-L. Uso, P. Mitic and L. J. Sucharov, Computational Mechanics Publications, Southampton, UK, 1996.
49. L. Neal, The development of the technical, professional and personal competencies of software engineering students through work based learning, Software *Engineering in Higher Education II*, edited by J-L. Uso, P. Mitic and L. J. Sucharov, Computational Mechanics Publications, Southampton, UK, 1996.
50. J. D. Tedford, R. H. A. Seidel and M. A. Islam, Teamwork and its influence on learning in industry based projects, *Proceedings of the 10th UICEE Annual Conference on Engineering Education*, Bangkok, March 2007, pp. 203–206.
51. A. Drobnic Vidic, Development of transferable skills within an engineering science context using problem-based learning, *International Journal of Engineering Education*, **24**(6), 2008, pp. 1071–1077.
52. P. E. Shrout and J. L. Fleiss, Intraclass correlations: Uses in assessing rater reliability, *Psychological Bulletin*, **86**(2), 1979, pp. 420–428.

53. D. Garson, Reliability Analysis, Statnotes from North Carolina State Univesity, http://faculty.chass.ncsu.edu/garson/PA765/reliab.htm (viewed 25.10.2009).
54. F. T. Anbari, Earned value project management method and extensions, *Project Management Journal*, **34**(4), 2003, pp. 12–23.
55. T. Sulaiman, B. Barton and T. Blackburn, Agile EVM—Earned value management in Scrum projects, *Proceedings of the Agile 2006 Conference*, 2006, pp. 7–16.
56. A. Cabri and M. Griffiths, Earned value and agile reporting, *Proceedings of the Agile 2006 Conference*, 2006, pp. 17–22.
57. P. Berander, Using students as subjects in requirements prioritization, *Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE'04)*, 2004, pp. 167–176.
58. M. Svahnberg, A. Aurum and C. Wohlin, Using students as subjects—An empirical evaluation, *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement (ESEM 2008),* October 9–10, 2008, Kaiserslautern, Germany, 2008, pp. 288–290.

Viljan Mahnic is an Associate Professor and the Head of the Software Engineering Laboratory at the Faculty of Computer and Information Science of the University of Ljubljana, Slovenia. His teaching and research interests include agile software development methods, software process improvement, empirical software engineering and software measurement. He received his Ph.D. in Computer Science from the University of Ljubljana in 1990.