*International Journal of Engineering Education* Vol. 32, No. 5(A), pp. 2121–2133, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00


# The Influence of Diffusion of Innovation Theory Factors on Undergraduate Students' Adoption of Scrum*

VILJAN MAHNIČ and TOMAŽ HOVELJA
Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, 1000 Ljubljana, Slovenia.
E-mail: viljan.mahnic@fri.uni-lj.si; tomaz.hovelja@fri.uni-lj.si

Since Scrum is the most widespread agile software development method, teaching it is an important issue to prepare students for their professional careers. Scrum is often taught within the scope of a software engineering capstone course, which makes it possible for students to learn Scrum practices through practical project work. In this study, we use the Diffusion of Innovation theory (DOI) to analyze to what stage such a course enables students to assimilate the core Scrum practices and the factors that have the most impact on Scrum adoption. The study is based on the results of a survey that was conducted after each Sprint of the capstone course at the University of Ljubljana, Slovenia; the course has four Sprints and was attended by 88 undergraduates. It is shown that at the end of the course, all core Scrum practices reach either the acceptance or the routinization stage, and 11 most influential DOI factors are identified.

**Keywords:** Scrum; capstone course; diffusion of innovation; software engineering education; agile software development

## 1. Introduction

Although agile methods for software development [1, 2] were considered controversial at the beginning, their adoption in the industry has reached mainstream proportions. The Forrester's Global Developer Technographics Survey [3] found that agile development process (used by 35% of respondents) prevails over iterative and waterfall approaches (used by 21% and 13% of respondents, respectively), while 31% of respondents reported no use of a formal process methodology. According to the latest State of Agile Survey [4], the agile momentum continues and more bigger companies scale and embrace agile methods as part of the larger vision to deliver software faster, easier, and smarter.

Agile projects are also reported to be more successful; the results of the Standish Group's 2011 Chaos Report show that the success rate for agile projects (42%) is three times higher than for traditional waterfall projects (14%). Agile teams report significant improvements in productivity, quality, and stakeholder satisfaction, and reasonable improvements in cost [5]. Adopting agile methods also improves management of the development process and customer relationships [6], decreases the amount of overtime and increases customer satisfaction [7].

In order to fulfill industry needs and prepare students for their professional careers, teaching agile methods has become an important issue. Until a few years ago courses on agile methods were rather rare [8] and most attention was devoted to teaching Extreme Programming [9–11] and its practices, particularly pair programming [12] and test-driven development [13, 14]. However, given the fact that Scrum [15, 16] became the most widespread agile method (according to [4], Scrum and its variants are used by 72% of respondents) a significant shift towards teaching Scrum has been noticed recently.

An outline of the existing literature [17] revealed that the most widespread approach to teaching Scrum is through practical work on student projects, e.g., [18–24]. This approach takes into account the statement of Scrum creators that Scrum is simple to understand yet difficult to master [25]. Therefore, in order to learn Scrum, the students do not need extensive lectures but should try it in practice. Courses of this type often exploit the benefits of capstone projects that simulate professional working environment as much as possible.

Considerable effort has also been spent to find alternative ways of teaching that would enhance the learning of Scrum in an engaging way. Von Wangenheim et al. [26] introduced a manual paper and pencil educational game in addition to theoretical lectures, Rodriguez et al. [27] developed a tool that simulates a Scrum-based team room using different virtual elements, while Paasivaara et al. [28] described a LEGO-based game that simulates Scrum through building a product of LEGO blocks. Paasivara et al. [29] also used distributed Scrum to teach students global software engineering skills. More recently, the consideration of learning styles was suggested to enhance Scrum learning [30, 31] and the impact of agile coaching on students' performance was studied [32]. Scrum was also used in combination with a framework of design patterns to develop teaching/learning haptic simulators [33].

A well designed capstone course is a good place to teach Scrum since many Scrum practices seem to be controversial for beginners and can only be fully understood after trying them in practice. A typical

* Accepted 8 May 2016.

example are user stories, which only provide a short description of each required functionality written on a paper note card, while all the details must be clarified in conversations with the customer representative [34]. A study by Mahnic and Hovelja [35] revealed that the students are suspicious about this approach in the beginning, but their opinions significantly improve after they gain more experience. It was shown that the capstone course helped them to grasp the main concepts and understand the advantages and limitations of user stories, as well as the factors that affect their successful use in practice.

Therefore, it can be argued that teaching Scrum through practical work is crucial for a proper understanding of its concepts and practices. In order to make students operational on their jobs, it is important that these practices are imparted in a way that encourages their assimilation and proper execution at the workplace.

In this paper, a study is described that was conducted at the University of Ljubljana with the aim of identifying the level of assimilation of Scrum practices by students who attended the software engineering capstone course in the Summer term of the 2013–2014 academic year, as well as the factors that affect their willingness to accept Scrum as a software development methodology.

The study starts from the premise that Scrum is a new promising methodology that can in many cases replace traditional software development methods that the students have already mastered before attending the capstone course. Compared to traditional software processes that are predominantly technical, Scrum places greater emphasis on the social aspects of computing and stresses the role of self-organizing and self-managing teams that are collectively responsible for the development of the required functionality. The tasks are no more assigned to developers by the project manager, but on the basis of discussion among the team members. By allowing user requirements to evolve throughout the project, tight collaboration with the customer is required. Only the basic strategic aspects are defined in advance, while operational details are planned from iteration to iteration and Daily Scrum meetings are used for monitoring current progress.

From this point of view, the introduction of Scrum represents an innovation that significantly changes the working habits of developers, thus influencing their willingness to adopt the agile approach. Therefore, the success of Scrum should not only be measured in terms of project costs, productivity, code quality, customer satisfaction or delivery time (which is usually done by researchers in the area of software engineering and information systems development), but considerable attention should be devoted to factors affecting its adoption and long-term acceptance.

For the purpose of the study, students' opinions about Scrum and its practices were gathered by a comprehensive questionnaire that was answered after each Sprint (altogether 4 times) in order to analyze how students' perceptions change when they get more practice. In order to assess the level of assimilation of Scrum practices, the innovation diffusion model [36, 37] was used, while the Diffusion of Innovations (DOI) theory [38] served as a basis for studying factors affecting students' willingness to use Scrum.

The reminder of this paper is organized as follows: In Sections 2 and 3, respectively, research questions and study design are described. The results are presented in Section 4 and discussed in Section 5. Section 6 provides a conclusion.

## 2. Research questions

In the information technology (IT) literature, the best-known model describing the process in which an innovation (i.e., an IT solution) is assimilated by an organization, a group, or an individual, is the six-stage model defined by Kwon and Zmud [36] and Cooper and Zmud [37]. Each stage describes a certain level to which an innovation penetrates an adopting unit:

- Initiation: A match is found between an innovation and its application by the adopting unit.
- Adoption: A decision is reached to adopt the innovation.
- Adaptation: The innovation is adjusted and installed; members of the adopting unit are trained to use it. As a result, the innovation is available for use in organization.
- Acceptance: Members of the adopting unit are induced to commit to the innovation's usage.
- Routinization: Usage of the innovation is encouraged as a normal activity.
- Infusion: The innovation is used in a comprehensive and sophisticated manner so that the effectiveness of the adopting unit increases.

According to Senapathi and Srinivasan [39] the initial three stages relate to 'adoptive' behavior of an innovation, while the last three stages relate to an innovation's 'post-adoptive' use. In order to be effective, an innovation should reach one of the post-adoptive stages.

Considering the aforementioned innovation diffusion model, two research questions were defined:

- RQ1: How deeply (i.e., to which stage) do the students assimilate Scrum practices after completing our capstone course?

- RQ2: How does the stage of assimilation change from Sprint to Sprint?

The depth of assimilation depends on a variety of factors; therefore, it is also important to identify the factors that most significantly influence adoption of an innovation among its potential users.

Rogers's DOI theory [38] suggests that the adoption of an innovation depends on five sets of characteristics, called factors: (1) innovation factors, (2) individual factors, (3) task factors, (4) environmental factors, and (5) organizational factors. Each factor is further decomposed into multiple items (traits), so that Rogers's model comprises altogether 29 factors. Using this model as a basis, we wanted to answer the following research questions:

- RQ3: Which DOI factors importantly influence the adoption of Scrum methodology?
- RQ4: How do students' opinions about most important DOI factors change from Sprint to Sprint?

Since some authors (e.g., [39, 40]) claim that factors affecting successful adoption of an innovation differ between adoptive and post-adoptive stages, another research question was added:

- RQ5: How does the perceived importance of DOI factors change between adoptive and post-adoptive stages?

## 3. Study design

### *3.1 The capstone course*

The study was conducted within the scope of the software engineering capstone course at the University of Ljubljana, whose main aim is to teach Scrum through practical work on software projects from different real world problem domains [18, 20]. Additionally, by following recommendations for using student subjects in empirical studies [41, 42], the course also serves as a basis for different empirical studies [35, 43, 44].

The course lasts 15 weeks and is taken by undergraduate Computer Science students in their last semester. Before attending the course, the students must complete classes on data bases, software development processes, and Web programming. Therefore, it is assumed that they have already mastered traditional software development methodologies and have enough technical skills to start working on a more serious project.

The students are required to work in teams of four in order to develop a project on the basis of user requirements provided by a domain expert playing the role of the Product Owner. Each team must follow Scrum rules and perform all practices prescribed by the method.

The course consists of four Sprints. The first Sprint (also called Sprint 0) lasts three weeks and serves as a preparatory Sprint before the start of the project. During Sprint 0 formal lectures take place in order to teach students Scrum and how to apply user stories for requirements specification and project planning [34, 45]. 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. At the end of Sprint 0, each team estimates the effort required for implementation of each user story using planning poker [46] and prepares the release plan.

The rest of the course is divided into three regular Scrum Sprints, each lasting four weeks. Each Sprint starts with a Sprint planning meeting, at which the 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 the work spent and the work that remains. At the end of each Sprint, the Sprint review and Sprint retrospective meetings take place. At the review meeting, the students present their results to the instructors, while at the retrospective meeting both students and instructors meet to review the development process in the previous Sprint, giving suggestions for improvements in the next.

The Product Owner strictly enforces the concept of "done" and requires each user story to pass all acceptance tests. All stories that do not conform to user requirements are rejected and (if the shortcomings are not removed by the end of the Sprint) a new story is defined in the Product Backlog requiring completion of the missing features in one of the remaining Sprints. After three Sprints the first release should be complete and delivered to the customer.

Strictly following the Scrum principles, all roles are clearly defined. Students act as self-organizing and self-managing Scrum Teams that are collectively responsible for the implementation of the required functionality. The instructors do not interfere in the distribution of tasks among team members and the estimation of effort, but play the role of ScrumMasters of all teams responsible for the Scrum process and teaching Scrum to everyone involved in the project. They merely act as facilitators, ensuring that Scrum runs smoothly and everybody obeys its rules and practices.

The role of the Product Owner is played by a project domain expert representing the customer. The Product Owner can be one of the instructors (if the project is defined within the department) or a

representative of a company (if the project is developed for a partner from industry). The Product Owner must maintain the Product Backlog, prioritize user stories, and answer students' questions regarding user requirements. At the end of each Sprint, he or she is also responsible for accepting or rejecting the students' implementations of user stories while strictly enforcing the notion of "done", thus assuring that the implementation of each story is practically usable.

The study described in this paper was conducted in the Summer term of the 2013–2014 academic year when the course was attended by 88 students. Each team could choose between two different projects. The first project was defined within the department and consisted of the development of a Web-based Scrum project management tool. The second project was defined in partnership with a Slovenian IT company and required the development of a Web portal enabling different service providers to offer their services through the internet, thus giving their customers a possibility to find and book the services from their homes.

### *3.2 Questionnaire*

For the purpose of the study, a questionnaire was developed that consisted of two parts. The first part was intended to collect data required for answering research questions RQ1 and RQ2. It consisted of 13 questions referring to 13 core Scrum practices described in Table 1. For each question, 6 possible answers were offered, each of them corresponding to one of the 6 stages of assimilation defined by Cooper and Zmud [37].

**Table 1.** Core Scrum practices studied in the first part of the questionnaire

| Scrum practice | Description |
|---|---|
| Maintenance of the Product Backlog | The Product Backlog is a prioritized list of requirements for the system or product being developed. The Product Owner is responsible for its contents, prioritization, and availability. The Product Backlog is never complete, but evolves as the product and the environment in which it will be used evolves. |
| Using user stories for requirements specification | Each Product Backlog Item is represented as a user story consisting of a short description and a set of acceptance tests. A story description is formulated in the language of the customer and is intentionally short enough to be hand-written on a paper note card. |
| Cooperation with the Product Owner | All details regarding each user story implementation should be clarified in communication with the Product Owner. Therefore, good cooperation between the Product Owner and the Team is crucial for the success of a project. |
| Effort estimation | For the purpose of planning, user stories are estimated in story points using one of the techniques proposed by agile methods, i.e., planning poker or the team estimation game. |
| Release planning on the basis of estimated velocity | Considering the estimated velocity of the Team, user stories are allocated to Sprints to establish a rough completion date (or adapt the contents of the next release to the desired completion date). |
| Sprint planning | At the beginning of each Sprint, the Product Owner and the Team meet at the Sprint planning meeting to define the contents of the next Sprint. The amount of work agreed must not exceed the estimated velocity of the Team. |
| Maintenance of the Sprint Backlog | In the second part of the Sprint planning meeting, the Team compiles a list of tasks required for turning the selected subset of user stories into an increment of potentially shippable product functionality. The tasks are estimated and assigned to Team members. During the Sprint, the Sprint Backlog evolves by adding new tasks, if necessary, and decomposing roughly defined tasks into smaller ones. |
| Daily Scrum Meetings | Every day, the Team gets together for a 15-minute meeting at which each Team member answers three questions: What have you done since the previous meeting? What do you plan to do till the next meeting? Are there any impediments in your way? |
| Progress monitoring through burndown charts | Release and Sprint burndown charts show the amount of work remaining across time, thus visualizing the correlation between the amount of work remaining and the progress of the Team in reducing this work. |
| Strict enforcement of the concept of "done" | The Product Owner only accepts those user stories that conform to the definition of "done". This definition represents the project's quality statement for a user story ensuring that the story is fully developed and tested, that it works without errors, and that no more work is left to be done. |
| Sprint Review Meetings | At the end of each Sprint, the Team presents the results of the Sprint to the Product Owner and any other stakeholders who want to attend. |
| Sprint Retrospective meetings | Before the next Sprint, the ScrumMaster and the Team meet to review the development process in the previous Sprint, giving suggestions for improvements in the next. |
| Proper execution of the prescribed roles (the Product Owner, the ScrumMaster, the Team) | The Product Owner defines the vision of what is to be developed, maintains the Product Backlog, provides details regarding user requirements, and evaluates implementation of user stories. The ScrumMaster ensures that everybody follows Scrum rules and practices. The Team acts as a self-organizing, self-managing, and cross-functional development team that is collectively responsible for the success of the project. |

**Table 2.** Possible answers to questions concerning the level of assimilation

| Possible answers | | Corresponding level of assimilation |
|---|---|---|
| 1 | It is the first time that I have been acquainted with this practice. | Initiation |
| 2 | I have not performed this practice yet. | Adoption |
| 3 | I have enough knowledge to perform this practice. | Adaptation |
| 4 | I have positive experience with this practice. | Acceptance |
| 5 | I have performed this practice so many times that it has become a routine. | Routinization |
| 6 | I perform this practice effectively and efficiently, which contributes to better execution of other Scrum activities and improves the quality of the software developed. | Infusion |

The answers were formulated as shown in Table 2 so that the students were not asked directly about their level of assimilation, but had to choose the description that best matched their experience with the corresponding practice.

The second part of the questionnaire was intended to collect data for answering research questions RQ3, RQ4, and RQ5. It was compiled on the basis of previous research by Mustonen-Ollila and Lyytinen [47] who used the DOI theory in their longitudinal study on the adoption of information system process innovations. This part consisted of 29 assertions covering all sets of factors (i.e., Innovation, Task, Individual, Environment, and Organization) incorporated in the Rogers's DOI model. The assertions were chosen in such a way that, when used in combination with the phrase "I am willing to use Scrum since", they described factors that possibly influence the acceptance of Scrum from the part of students. For each assertion, the students were asked how important it is using a 7-point Likert scale (1—"Entirely unimportant", 2—"Unimportant", 3—"Somewhat unimportant", 4—"Neither important nor unimportant", 5—"Somewhat important", 6—"Important", 7—"Extremely important").

The structure of the second part of the questionnaire is shown in Table 3.

**Table 3.** Structure of the second part of the questionnaire

| DOI factors | | Assertion |
|---|---|---|
| Innovation | Relative advantage | Scrum is better than the traditional disciplined approach to software development. |
| | Ease of use | Scrum is easier to understand and use in practice. |
| | Compatibility | Scrum better matches my preferred way of working, my values and experience. |
| | Visibility | It is possible to observe how other teams use Scrum. |
| | Trialability | Scrum can be easily trialed in a working environment. |
| | Price | The use of Scrum does not involve additional costs and effort. |
| | Problem solver | Scrum solves many problems concerning software development. |
| | Standard | Scrum has become a standard method for software development. |
| | Technological edge | Scrum is more advanced than other methods for software development. |
| Task | Commercial advantage | Scrum significantly contributes to better customer satisfaction. |
| | User need recognition | Scrum matches programmers' needs in software development. |
| | User resistance | Scrum simplifies execution of difficult tasks. |
| Individual | Own testing | I can easily experiment with Scrum. |
| | Personal contact network | My friends and colleagues strongly recommend the use of Scrum. |
| | Own rules and control of work | Scrum can be easily adapted to my way of work. |
| | Learning by doing | I can learn Scrum through practical project work. |
| Environmental | Cultural values | We (the students) are well-disposed to changes required by the Scrum methodology. |
| | Technological infrastructure | We (the students) have the entire technological infrastructure required for successful use of Scrum. |
| | Community norms | We (the students) use Scrum rules and practices as requested by the teacher. |
| | Funding | We (the students) have all resources at our disposal (literature, time, instructors' support) for successful use of Scrum. |
| Organizational | Interpersonal networks | Most students from previous generations strongly recommend the use of Scrum. |
| | Peer networks | I develop a Scrum project in a team of my classmates with whom I associate most. |
| | Informal communication | Scrum encourages spontaneous and informal communication among team members. |
| | Technological experience | I have a lot of experience with software development methodologies. |
| | Working teams | Scrum allows us to plan the work and solve most of the problems faced during the development by ourselves. |
| | Opinion leaders and change agents | Most persons whose opinion I respect strongly recommend the use of Scrum. |
| | Interdependence from others | Each new Scrum user significantly increases the usefulness of Scrum. |
| | Adopter type | I usually start using new technologies earlier than my colleagues and friends. |
| | Management hierarchy | Teacher's demand is the main reason that I use Scrum. |

### 3.3 Statistical methods used

The questionnaire was answered four times during the capstone course, i.e., after Sprint 0 and after each of the three regular Scrum Sprints. After Sprint 0, when the students finished theoretical lessons on Scrum and started with practical work, nine distributions of answers significantly deviated from normal distribution (skewness greater than $\pm 1$, kurtosis greater than $\pm 3$). Afterwards, the number of deviations diminished to four at the end of Sprint 3. In light of the above, as well as considering the fact that ordinal scales were used to represent different levels of assimilation (in the first part of the questionnaire) and degrees of agreement with assertions (in the second part of the questionnaire), a decision to use non-parametric statistical methods was made.

In order to answer research question RQ1, the median of answers to each question in the first part of the questionnaire was computed indicating how deeply the students assimilated particular Scrum practices after each Sprint. Similarly, in order to answer research question RQ3, the median of answers to each assertion in the second part of the questionnaire was used to find DOI factors with greatest impact on Scrum adoption.

Research questions RQ2, RQ4, and RQ5 required the before-after analysis. Since the survey was anonymous and the number of respondents varied slightly from Sprint to Sprint between 68 and 84, the responses obtained after each Sprint were treated as independent samples. Therefore, the Mann-Whitney-Wilcoxon test [48] was used to identify possible significant changes in students' opinions during the course. A two tailed p-value of 0.05 or less had to be reached in order to demonstrate that the distributions of the two tested variables differed significantly from each other.

## 4. Results

### 4.1 Research questions RQ1 and RQ2

Results referring to research questions RQ1 and RQ2 (i.e., the first part of the questionnaire) are gathered in Table 4. For each Sprint, the medians of students' responses regarding their use of each Scrum practice are presented. Additionally, for Sprints 1, 2 and 3, the p-values of the Mann-Whitney-Wilcoxon test that compared each of these Sprints to its immediate antecedent (i.e., Sprint 1 to Sprint 0, Sprint 2 to Sprint 1, and Sprint 3 to Sprint 2) are shown.

The results clearly indicate that students quickly assimilated all of the 13 core Scrum practices. While immediately after Sprint 0 the great majority reported only their acquaintance with Scrum but no practical experience, their proficiency in using Scrum practices deepened from Sprint to Sprint. After Sprint 1, six out of 13 practices reached the stage 4 (Acceptance), 6 practices were at stage 3 (Adaptation), and only one practice remained on stage 2 (Adoption). The level of assimilation further increased in Sprints 2 and 3. After Sprint 2, four practices reached the stage 5 (Routinization), while the remaining 9 practices were at stage 4 (Acceptance). At the end of Sprint 3, the median of students' responses corresponded to the routine execution of 9 practices and acceptance of 4 practices.

Therefore, with regard to research question RQ1, it can be concluded that the assimilation of Scrum reached the Acceptance and Routinization stages corresponding to post-adoptive usage of an innovation according to classification by Senapathi and Srinivasan [39].

With regard to research question RQ2, it is evident that the level of assimilation increased very quickly. The p-values of the Mann-Whitney-

**Table 4.** Assimilation level of Scrum practices across Sprints

| Scrum activity/Practice | Sprint 0 | Sprint 1 | | Sprint 2 | | Sprint 3 | |
|---|---|---|---|---|---|---|---|
| | Median | Median | p-value | Median | p-value | Median | p-value |
| Maintenance of the Product Backlog | 2 | 3 | 0.000 | 4 | 0.000 | 5 | 0.003 |
| Using user stories for requirements specification | 2 | 4 | 0.000 | 5 | 0.000 | 5 | 0.015 |
| Cooperation with the Product Owner | 2 | 4 | 0.000 | 4 | 0.000 | 5 | 0.106 |
| Effort estimation | 2 | 3 | 0.000 | 4 | 0.000 | 5 | 0.137 |
| Release planning on the basis of estimated velocity | 2 | 3 | 0.000 | 4 | 0.000 | 4 | 0.119 |
| Sprint planning | 2 | 3 | 0.000 | 4 | 0.000 | 5 | 0.033 |
| Maintenance of the Sprint Backlog | 2 | 4 | 0.000 | 5 | 0.000 | 5 | 0.080 |
| Daily Scrum Meetings | 2 | 4 | 0.000 | 5 | 0.001 | 5 | 0.798 |
| Progress monitoring through burndown charts | 2 | 3 | 0.000 | 4 | 0.000 | 4 | 0.169 |
| Strict enforcement of the concept of "done" | 2 | 4 | 0.000 | 4 | 0.000 | 5 | 0.173 |
| Sprint Review Meetings | 2 | 3 | 0.000 | 4 | 0.000 | 4 | 0.039 |
| Sprint Retrospective meetings | 2 | 2 | 0.000 | 4 | 0.000 | 4 | 0.004 |
| Proper execution of prescribed roles (Product Owner, ScrumMaster, Team) | 2 | 4 | 0.000 | 5 | 0.000 | 5 | 0.305 |

**Table 5.** Students' perceptions of DOI factors across Sprints

| | | Sprint 0 | Sprint 1 | | Sprint 2 | | Sprint 3 | | Sprint 3 vs. Sprint 0 | Sprint 3 vs. Sprint 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| **DOI factors** | | **Median** | **Median** | **p-value** | **Median** | **p-value** | **Median** | **p-value** | **p-value** | **p-value** |
| Innovation | Relative advantage | 6 | 5 | 0.551 | 5 | 0.743 | 5 | 0.975 | 0.844 | 0.681 |
| | Ease of use | 5 | 6 | 0.011 | 5 | 0.803 | 5 | 0.471 | 0.085 | 0.289 |
| | Compatibility | 5 | 5 | 0.834 | 6 | 0.494 | 6 | 0.411 | 0.199 | 0.091 |
| | Visibility | 5 | 5 | 0.570 | 5 | 0.028 | 5 | 0.258 | 0.009 | 0.001 |
| | Trialability | 5 | 5 | 0.203 | 6 | 0.489 | 6 | 0.794 | 0.026 | 0.326 |
| | Price | 5 | 5 | 0.048 | 6 | 0.466 | 5 | 0.688 | 0.001 | 0.217 |
| | Problem solver | 5 | 5 | 0.971 | 5 | 0.483 | 6 | 0.024 | 0.005 | 0.004 |
| | Standard | 5 | 4 | 0.296 | 5 | 0.329 | 5 | 0.072 | 0.140 | 0.004 |
| | Technological edge | 5 | 5 | 0.796 | 5 | 0.305 | 5 | 0.409 | 0.120 | 0.052 |
| Task | Commercial advantage | 6 | 5.5 | 0.624 | 6 | 0.745 | 6 | 0.167 | 0.338 | 0.099 |
| | User need recognition | 5 | 5 | 0.923 | 5 | 0.536 | 6 | 0.420 | 0.143 | 0.103 |
| | User resistance | 5 | 5 | 0.653 | 5 | 0.236 | 5 | 0.538 | 0.781 | 0.486 |
| Individual | Own testing | 4 | 4 | 0.259 | 5 | 0.000 | 5 | 0.655 | 0.000 | 0.000 |
| | Personal contact network | 4 | 4 | 0.821 | 4 | 0.140 | 4 | 0.255 | 0.014 | 0.005 |
| | Own rules and control of work | 5 | 5 | 0.328 | 5 | 0.171 | 5 | 0.651 | 0.004 | 0.064 |
| | Learning by doing | 5 | 5 | 0.194 | 5 | 0.433 | 5 | 0.318 | 0.003 | 0.059 |
| Environmental | Cultural values | 5 | 5 | 0.382 | 6 | 0.289 | 6 | 0.115 | 0.000 | 0.017 |
| | Technological infrastructure | 6 | 6 | 0.402 | 6 | 0.771 | 6 | 0.417 | 0.044 | 0.237 |
| | Community norms | 4 | 5 | 0.000 | 5 | 0.355 | 5.5 | 0.875 | 0.000 | 0.261 |
| | Funding | 5 | 6 | 0.496 | 6 | 0.280 | 6 | 0.251 | 0.009 | 0.030 |
| Organizational | Interpersonal networks | 4 | 4 | 0.522 | 4 | 0.064 | 4 | 0.667 | 0.073 | 0.016 |
| | Peer networks | 5 | 5 | 0.594 | 6 | 0.697 | 6 | 0.587 | 0.199 | 0.365 |
| | Informal communication | 6 | 6 | 0.209 | 6 | 0.247 | 6 | 0.815 | 0.032 | 0.339 |
| | Technological experience | 3 | 4 | 0.002 | 5 | 0.000 | 5 | 0.402 | 0.000 | 0.000 |
| | Working teams | 5 | 5 | 0.077 | 6 | 0.506 | 6 | 0.449 | 0.002 | 0.121 |
| | Opinion leaders and change agents | 4 | 4 | 0.540 | 4 | 0.151 | 5 | 0.128 | 0.026 | 0.002 |
| | Interdependence from others | 4 | 5 | 0.008 | 5 | 0.478 | 5 | 0.201 | 0.000 | 0.028 |
| | Adopter type | 4 | 4 | 0.661 | 5 | 0.001 | 5 | 0.997 | 0.002 | 0.002 |
| | Management hierarchy | 5 | 6 | 0.201 | 6 | 0.980 | 5 | 0.116 | 0.951 | 0.110 |

Wilcoxon test indicate that the greatest growth was achieved in Sprints 1 and 2 when the level of assimilation statistically significantly differed from preceding Sprints 0 and 1 for all practices studied. In Sprint 3 the growth was somewhat slower, but still statistically significant with regard to 5 practices out of 13.

### *4.2 Research questions RQ3, RQ4 and RQ5*

Results referring to research questions RQ3, RQ4 and RQ5 (i.e., the second part of the questionnaire) are gathered in Table 5. For each Sprint, the median values for 29 factors that (according to the DOI theory) possibly influence the diffusion of an innovation are shown indicating how students rated their importance on a 7-point Likert scale. The "p-value" columns in Sprints 1, 2, and 3 represent the results of the Mann-Whitney-Wilcoxon test that we used to identify how significantly their opinions changed from preceding Sprint. The last two columns contain p-values of the Mann-Whitney-Wilcoxon test that compared students' opinions in initial Sprints (i.e., after Sprint 0 and Sprint 1) to their opinions at the end of the course (i.e., after Sprint 3).

The results indicate that the great majority of DOI factors were rated either "somewhat important" (the median value 5) or "important" (the median value 6). At the end of the course, there were only 2 factors considered "neither important nor unimportant" (the median value 4), and none of the factors was rated unimportant (the median value 3 or less).

The number of DOI factors that the students rated "somewhat important" was almost constant across Sprints, while the number of factors rated "important" increased from 4 after Sprint 0, to 11 at the end of the course. On the other hand, the number of DOI factors considered to be "neither important nor unimportant" decreased from 7 at the beginning, to 2 at the end of the course. There was only one factor rated "somewhat unimportant" (the median value 3) at the beginning but not later.

The p-values of the Mann-Whitney-Wilcoxon test indicate that students' opinions were relatively stable and did not change a lot between two consecutive Sprints. Most changes occurred between Sprint 0 and Sprint 1 when a statistically significant difference was identified with regard to 5 DOI

factors out of 29. Afterwards, the number of statistically significant changes diminished to 4 (between Sprints 1 and 2) and 1 (between Sprints 2 and 3). Therefore, it can be concluded that after Sprint 3 the students' opinions stabilized so that the results of the survey after Sprint 3 can be used to interpret the influence of DOI factors on their acceptance of Scrum.

With regard to research question RQ3, these results suggest that there are 11 DOI factors with the median value 6 that mostly influence students' willingness to accept Scrum as a new software development methodology. These factors belong to the following sets:

- Innovation factors: compatibility, trialability, problem solver.
- Task factors: commercial advantage, user need recognition.
- Environmental factors: cultural values, technological infrastructure, funding.
- Organizational factors: peer networks, informal communication, working teams.

It is also worth noting that 9 (out of 10) changes in students' opinions between consecutive Sprints that the Mann-Whitney-Wilcoxon test found to be significant did not affect the findings of RQ3 since all these changes referred to factors with lower impact, i.e., having median values less than 6 after Sprint 3. The only significant change in student opinions that affected the result of RQ3 happened between Sprints 2 and 3 when the survey revealed increased importance of the problem solver factor.

Therefore, with regard to research question RQ4, it cannot be concluded only that the student opinions were relatively stable and did not change a lot from Sprint to Sprint, but also that the changes in their opinions did not influence the identification of most important acceptance factors.

With regard to research question RQ5, the p-values in the last two columns of Table 5 indicate (at least at first sight) quite a lot of significant differences in students' opinions between Sprint 3 and initial Sprints 0 and 1. However, a more in-depth analysis has shown that:

- Most of the significant differences (11 out of 18 in comparison of Sprint 3 to Sprint 0 and 9 out of 12 in comparison of Sprint 3 to Sprint 1) referred to factors that were found not to be important for Scrum acceptance, thus not affecting the findings of research question RQ3.
- All significant differences concerning important acceptance factors were consistently in favor of these factors. The perceived importance of these factors grew during the course and was greater after Sprint 3 than after Sprints 0 and 1, thus indicating increasing awareness of importance of these factors.

Therefore, research question RQ5 did not identify such differences between adoptive and post-adoptive stages that would significantly affect the identification of most important acceptance factors.

## 5. Discussion

### *5.1 Research questions RQ1 and RQ2*

With regard to research questions RQ1 and RQ2, the results presented in Table 4 confirm the prevailing opinion that agile software development methods are best taught through practical project work [49].

Immediately after Sprint 0, all Scrum practices were rather new to the students, since most of them had no previous experience with Scrum, but only got acquainted with it during initial lectures. Therefore, it is normal that the great majority chose answer 2 (i.e., I have not performed this practice yet.), thus indicating their assimilation of Scrum to be at the adoption stage considering the 6-stage model of Cooper and Zmud [37]. Afterwards, the experience gained through practical project work helped them to master Scrum practices quickly and the assimilation level grew from Sprint to Sprint reaching the acceptance and routinization levels at the end of Sprint 3.

These results are in line with previous research that the authors performed within the scope of the software engineering course. A study on effort estimating and planning [43] found that the initial plans and effort estimates tend to be over-optimistic, but the abilities of estimating and planning improve from Sprint to Sprint so that most student teams are able to define almost accurate Sprint plans at the end of the course. A comprehensive analysis of students' opinions on user stories as a means for requirements specification [35] revealed that (in spite of initial doubts) the great majority of students successfully master this practice after three Sprints.

From the viewpoint of the capstone course execution, it is worth noting the differences among practices in terms of how quickly and how deeply their assimilation takes place. Table 4 clearly shows that the practices that Scrum requires to be performed on everyday basis (e.g., maintenance of the Sprint Backlog and Daily Scrum meetings) were assimilated more quickly and reached a higher assimilation stage than the practices that are performed periodically (e.g., release planning, Sprint review meetings, and Sprint retrospective meetings).

While this difference is understandable with regard to the aforementioned practices, the lower assimilation level of the progress monitoring

through burndown charts indicates a possible deficiency in the capstone course execution suggesting more attention to be devoted to this issue. Students should be encouraged to use Sprint burndown charts on everyday basis in order to have control over their projects and fulfillment of scope.

On the other hand, we were gratified that the students quickly grasped the use of user stories for requirements specification and proper execution of prescribed roles. With regard to user stories, it seems that our experience from previous years helped us to overcome initial doubts and establish appropriate expectations from the very beginning so that the students assimilated this new lightweight technique without problems. Successful use of user stories greatly depends on proper execution of prescribed roles, especially the role of Product Owner; therefore, it was beneficial for the course that these practices became a routine very soon.

As already mentioned, the practices that are performed only periodically (with exemption of Sprint planning and strict enforcement of the concept of "done") reached lower level of assimilation. This problem can be alleviated by using shorter Sprints and dividing the project into several releases. In such a way the students would have more possibilities to practice release and Sprint planning as well as Sprint review and Sprint retrospective meetings. The concept of "done" could be further enforced by evaluating user stories promptly, not only at the end of the Sprint.

### *5.2 Research questions RQ3, RQ4 and RQ5*

With regard to research question RQ3, 11 most influential DOI factors were identified that should be considered when teaching Scrum within the scope of a software engineering capstone course and preparing students to use it in their workplace.

Considering the innovation factors, the students are keener to accept Scrum if they find it compatible with their vision of how software should be developed, if they are able to try it in practice, and if they find it useful for managing their software development projects. With regard to task factors, they should believe that using Scrum significantly contributes to better customer satisfaction and matches the programmers' needs in software development.

The high rating of the trialabilty factor stressed once again the importance of practical work when teaching agile methods to students. With regard to other important innovation and task factors (i.e., compatibility, problem solver, commercial advantage, and user need recognition) it is crucial that students understand the problems that most frequently occur in software development and know how Scrum can be used to solve them. Ceschi et al. [6] cite the results of surveys of more than 8,000 projects showing that most project failures involve stakeholder problems causing that projects fail because of people and project management issues rather than technical issues. These problems are exactly the ones that Scrum is focused on. Therefore, during Sprint 0 Scrum should be presented to students as an appropriate tool for solving these problems and gaining commercial advantage. Additionally, the introduction of Scrum should be compatible with their preferred way of working as much as possible.

The acceptance of Scrum also greatly depends on environmental factors, which require its seamless incorporation in the study process, not disturbing the execution of other courses and complying with students' values regarding usefulness of the knowledge obtained. It is important that the students are provided with all required technological infrastructure and adequate support from the part of instructors playing the roles of ScrumMasters and Product Owners.

The role of the Product Owner is particularly important, since it is crucial for the success of a Scrum project [20]. He/she must communicate the vision of what is to be developed and define the criteria by which it will be judged. In order to ensure smooth running of the course, the Product Owner must provide timely answers to questions on details of user stories, and make quick evaluations of work completed strictly enforcing the concept of "done". A nonresponsive Product Owner can cause unproductive work periods, which make iteration planning more difficult or even impossible.

The required technological infrastructure should also comprise a Scrum project management tool that facilitates students the managing of their projects and helps the teaching staff with monitoring students' progress and reducing the burden of administrative work. An open source or a commercial tool of this kind can be used; however, an in-house developed tool can provide facilities that better match students' and teachers' needs [50].

With regard to organizational factors, the results indicate that the acceptance of Scrum greatly depends on social relationships among students. Teamwork plays an important role and teams composed of students that have been hobnobbing together since freshman year are expected to obtain better results. Therefore, the capstone course designers should consider the possibility to allow students to decide who they should work with when forming student teams.

The results also expose the importance of informal communication and Scrum's concepts of self-organizing and self-managing teams, allowing the students to plan their work and assign tasks by themselves. It seems that students like the freedom

that Scrum gives development teams and are ready to accept responsibility to find the best way how to accomplish the work committed.

On the other hand, the personal contact network and interpersonal network factors appeared to be least influential, being rated neither important nor unimportant. It is possible that the students evaluated these two factors in view of the fact that the use of Scrum was prescribed by the teacher, thus perceiving the exchange of experiences and evaluations of Scrum with their peers less important.

Findings of research questions RQ4 and RQ5 further corroborate the results of research question RQ3. Most changes in students' opinions that were identified during before-after analysis concerned adoption factors that appeared to be less important for Scrum acceptance. On the other hand, students' opinions regarding the 11 important factors found within the scope of research question RQ3 were consistent and indicated constant growth of their importance from Sprint to Sprint.

### *5.3 Relevance for software engineering and information systems research*

While the primary purpose of this study was to provide an in-depth analysis of students' acceptance of Scrum, its results can be used in software engineering research to pose initial hypotheses regarding Scrum adoption in industrial settings as suggested by Craver et al. [41]. Several studies have shown that in a project setting where the students have made a true commitment, students tend to act and think more like professionals [51] and provide answers that are in line with industrial practice [52]. Given the fact that the current body of knowledge lacks studies on Scrum [53–55], using student subjects in empirical studies can be helpful in filling this gap.

At the time of this writing, we were only able to find two studies dealing with acceptance of Scrum. Overhage et al. [56] developed a framework of drivers and inhibitors to developer acceptance of Scrum by upgrading the extended Technology Acceptance Model [57], while Overhage and Schlauderer [58] investigated long-term Scrum acceptance from the viewpoint of three DOI factors considering only a part of the innovation dimension, i.e., relative advantage, compatibility, and complexity. However, to the best of our knowledge, there has been no study dealing with Scrum acceptance considering the whole range of DOI factors.

On the other hand, the DOI theory served as the basis for a longitudinal study on why organizations adopt information system process innovations [47]. The authors analyzed factors that affected over 200 information system process innovation decisions over a period that spanned four decades. It was found that in all adoptions the innovation factor played the most important role, while other factors varied from one innovation type to another.

More research was also oriented towards extreme programming (XP), although XP is by far less used in industry than Scrum. Acceptance of XP in terms of the innovation diffusion cycle was studied by Mangalaraj et al. [40] who found individual, team, technological, task, and environmental factors to influence the acceptance of XP practices in an organization.

### *5.4 Threats to validity*

It is possible that the results concerning students' perceptions of Scrum were affected by the way how the capstone course was delivered. Therefore, the stage of assimilation Scrum practices and students' opinions on importance of DOI factors must be interpreted in terms of the context in which Scrum was introduced. It can be argued that the results are valid within the context of a software engineering capstone course and can be generalized with regard to teaching such a course in a similar way.

From the viewpoint 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 validity of findings, every effort was made to simulate an industrial environment as closely as possible. Students worked on projects dealing with real life problems (one of them was defined in cooperation with a software company) and followed the Scrum method as strictly as their other academic duties allowed.

It is also possible that students exaggerated in their opinions concerning the stage of assimilation of Scrum practices. However, the results of previous studies on students' abilities of effort estimation and planning [43] and their opinions on user stories [35] also confirm that students obtain considerable knowledge and experience to use Scrum practices effectively and efficiently at the end of the course.

Another possible threat to validity stems from the fact that students' opinions were gathered by the means of a questionnaire, which did not allow to explain their opinions in more detail. Additionally, it is possible that different students interpreted the questions in a different way. Therefore, augmenting the questionnaire with structured interviews with a representative subset of students would improve the validity.

## 6. Conclusions

The aim of our study was twofold: (1) to analyze the level of assimilation of core Scrum practices when Scrum is taught to undergraduates within the scope

of a software engineering capstone course; (2) to identify those DOI factors that most significantly influence Scrum acceptance among students.

With regard to the first aim, the results of the study confirm that the software engineering capstone course is an appropriate place for exposing students to Scrum. It is shown that practical work helps students to assimilate core Scrum practices so that they can use them in a way that corresponds to routinization (9 practices) and acceptance stages (4 practices) of the innovation diffusion model.

With regard to the second aim, 11 DOI factors were identified that most importantly influence students' acceptance of Scrum. These factors belong to innovation, task, environmental, and organizational dimensions and should be considered when designing the capstone course.

It is expected that the findings of this study will have implications on how Scrum is taught to undergraduates. While the results in general show a high level of assimilation of Scrum practices, the study also revealed several possibilities for improvement. Particularly those practices that Scrum requires to be performed periodically should deserve more attention since their assimilation was lower compared to practices that are performed on everyday basis. Therefore, it is suggested to use shorter Sprints and split the project into two (or more) releases in order to increase the number of release planning, Sprint planning, Sprint review, and Sprint retrospective meetings. More attention should also be devoted to the use of burndown charts as a means for monitoring project progress.

With regard to DOI factors influencing Scrum acceptance, the problem solver, commercial advantage, user need recognition, and compatibility factors should be adequately addressed through theoretical lessons during Sprint 0. The lessons should instill appropriate expectations regarding Scrum benefits so that the students could recognize its usefulness for solving software development problems and obtaining greater customer satisfaction. These perceptions should be further amplified through practical work on the project.

The environmental factors require the Scrum course to be incorporated in the study process seamlessly, provide students with all required technological infrastructure (including software for managing Scrum projects), and ensure proper execution of the Product Owner and ScrumMaster roles.

In order to satisfy the organizational factors, the course should encourage social relationships among students, allow them to choose their own teams, and let them organize their work themselves. These factors concur with Scrum concepts of face-to-face communication and self-managing teams, which additionally facilitates Scrum acceptance.

The results of the study can also be used for posing initial hypotheses regarding Scrum adoption in industrial settings. From this viewpoint, it is important that the study established a framework covering the whole range of DOI factors, which can be used for further research.

## References


1. P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, *Agile software development methods*, VTT Electronic, Espoo, Finland, 2002.
2. L. Williams, Agile software development methodologies and practices, *Advances in Computers*, **80**, 2010, pp. 1–44.
3. D. West and T. Grant, Agile Development: Mainstream adoption has changed agility, Trends in real-world adoption of agile methods, Forrester Research, January 20, 2010, https:// www.forrester.com/Agile+Development+Mainstream+Adoption+Has+Changed+Agility/fulltext/-/E-RES56100?objectid=RES56100, accessed 1 February 2016.
4. VersionOne, *9th Annual State of Agile Survey*, Atlanta, GA. http://info.versionone.com/state-of-agile-development-survey-ninth.html, accessed 1 February 2016.
5. S. W. Ambler, Has agile peaked? Let's look at the numbers. *Dr. Dobb's Journal*, May 07, 2008, http://www.ddj.com/architect/207600615?pgno=1, accessed 1 February 2016.
6. 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.
7. 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)*, Denver, CO, 24–29 July 2005, pp. 70–79.
8. D. F. Rico and H. H. Sayani, Use of agile methods in software engineering education, *Proceedings of the Agile 2009 Conference*, Chicago, IL, August 24–28, 2009, pp. 174–179.
9. 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)*, Madrid, Spain, March 20–22, 2003, pp. 176–184.
10. 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)*, Herzlia, Israel, 4–5 November 2003, pp. 53–59.
11. 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)*, Covington, KY, 25–27 February 2002, pp. 184–191.
12. L. Williams, Lessons learned from seven years of pair programming at North Carolina State University, *ACM SIGCSE Bulletin*, **39**(4), 2007, pp. 79–83.
13. P. J. Schroeder and D. Rothe, Teaching unit testing using test-driven development, *4th Annual Workshop on Teaching Software Testing (WTST 4)*, Florida Institute of Technology, Melbourne, FL, February 4–6, 2005. http://www.testingeducation.org/conference/wtst4/pjs_wtst4.pdf, accessed 1 February 2016.
14. 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)*, Honolulu, HI, 10–12 July 2006, pp. 500–505.
15. K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, Redmond, 2004.
16. K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, 2002.

17. V. Mahnic, Scrum in software engineering courses: an outline of the literature, *Global Journal of Engineering Education*, **17**(2), 2015, pp. 77–83.
18. V. Mahnic, Teaching Scrum through team-project work: students' perceptions and teacher's observations, *International Journal of Engineering Education*, **26**(1), 2010, pp. 96–110.
19. T. Reichlmayr, Working towards the student Scrum—developing agile Android applications, *Proceedings of the 118th ASEE Annual Conference and Exposition*, Vancouver, Canada, June 26–29, 2011, p. 12.
20. V. Mahnic, A capstone course on agile software development using Scrum, *IEEE Transactions on Education*, **55**(1), 2012, pp. 99–106.
21. L. Werner, D. Arcamone and B. Ross, Using Scrum in a quarter-length undergraduate software engineering course. *Journal of Computing Sciences in Colleges*, **27**(4), 2012, pp. 140–150.
22. M. Kropp and A. Meier, Teaching agile software development at university level: Values, management, and craftsmanship, *Proceedings of the 26th Conference on Software Engineering Education and Training*, San Francisco, CA, May 19–21, 2013, pp. 179–188.
23. A. Scharf and A. Koch, Scrum in a software engineering course: an in-depth praxis report, *Proceedings of the 26th Conference on Software Engineering Education and Training (CSEE&T 2013)*, San Francisco, CA, May 19–21, 2013, pp. 159–168.
24. S. D. Zorzo, L. De Ponte and D. Lucredio, Using Scrum to teach software engineering: a case study, *Proceedings of the Frontiers in Education Conference*, Oklahoma City, OK, October 23–26, 2013, pp. 455–461.
25. K. Schwaber and J. Sutherland, *The Scrum Guide*, 2013, http://www.scrumguides.org/, accessed 1 February 2016.
26. C. G. Von Wangenheim, R. Savi and A. F. Borgatto, SCRUMIA–An educational game for teaching SCRUM in computing courses, *Journal of Systems and Software*, **86**(10), 2013, pp. 2675–2687.
27. G. Rodriguez, A. Soria and M. Campo, Virtual Scrum: a teaching aid to introduce undergraduate software engineering students to scrum, *Computer Applications in Engineering Education*, **23**(1), 2015, pp. 147–156.
28. M. Paasivaara, V. Heikkilä, C. Lassenius and T. Toivola, Teaching students Scrum using LEGO blocks, *Companion Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India, May 31–June 7, 2014, pp. 382–391.
29. M. Paasivaara, C. Lassenius, D. Damian, P. Raty and A. Schroter, Teaching students global software engineering skills using distributed Scrum, *Proceedings of the 35th International Conference on Software Engineering*, San Francisco, CA, May 18–26, 2013, pp. 1128–1137.
30. E. Scott, G. Rodriguez, A. Soria and M. Campo, Are learning styles useful indicators to discover how students use Scrum for the first time?, *Computers in Human Behavior*, **36**, 2014, pp. 56–64.
31. E. Scott, G. Rodriguez, A. Soria and M. Campo, Towards better Scrum learning using learning styles, *Journal of Systems and Software*, **111**, 2016, pp. 242–253.
32. G. Rodriguez, A. Soria and M. Campo, Measuring the impact of agile coaching on students' performance, *IEEE Transactions on Education*, **59**(3), 2016, pp. 202–209.
33. C. Fernandez, G. Esteban, F. J. Rodriguez-Lera, F. Rodriguez-Sedano and D. Diez, Design patterns combination for agile development of teaching/learning haptic simulators, *International Journal of Engineering Education*, **32**(2B), 2016, pp. 1036–1052.
34. M. Cohn, *User Stories Applied for Agile Software Development*, Addison-Wesley, Boston, 2004.
35. V. Mahnic and T. Hovelja, Teaching user stories within the scope of a software engineering capstone course: analysis of students' opinions, *International Journal of Engineering Education*, **30**(4), 2014, pp. 901–915.
36. T. H. Kwon and R. W. Zmud, Unifying the fragmented models of information systems implementation, in R. J. Boland and R. A. Hirschheim (eds.): *Critical Issues in Information Systems Research*, Wiley & Sons, New York, 1987, pp. 227–251.
37. R. B. Cooper and R. W. Zmud, Information technology implementation research: A technological diffusion approach, *Management Science*, **36**, 1990, pp.123–139.
38. E. M. Rogers, *Diffusion of Innovations*, 5th edition, Free Press, New York, 2003.
39. M. Senapathi and A. Srinivasan, Understanding post-adoptive agile usage: An exploratory cross-case analysis, *Journal of Systems and Software*, **85**(6), 2012, pp. 1255–1268.
40. G. Mangalaraj, R. Mahapatra and S. Nerur, Acceptance of software process innovations—the case of extreme programming, *European Journal of Information Systems*, **18**, 2009, pp. 344–354.
41. J. Carver, L. Jaccheri, S. Morasca and F. Shull, Issues in using students in empirical studies in software engineering education, *Proceedings of the 9th International Software Metrics Symposium*, Sydney, Australia, September 3–5, 2003, pp. 239–249.
42. J. C. Carver, L. Jaccheri, S. Morasca and F. Shull, A checklist for integrating student empirical studies with research and teaching goals, *Empirical Software Engineering*, **15**(1), 2010, pp. 35–59.
43. V. Mahnic, A case study on agile estimating and planning using Scrum, *Elektronika ir Elektrotechnika* (*Electronics and Electrical Engineering*), **111**(5), 2011, pp. 123–128.
44. V. Mahnic and T. Hovelja, On using planning poker for estimating user stories, *Journal of Systems and Software*, **85**(9), 2012, pp. 2086–2095.
45. M. Cohn, *Agile Estimating and Planning*, Prentice Hall, Upper Saddle River, 2006.
46. J. Grenning, Planning poker or how to avoid analysis paralysis while release planning, 2002, http://www.renaissance software.net/files/articles/PlanningPoker-v1.1.pdf, accessed 1 February 2016.
47. E. Mustonen-Ollila and K. Lyytinen, Why organizations adopt information system process innovations: a longitudinal study using Diffusion of Innovation theory, *Information Systems Journal*, **13**(3), 2003, pp. 275–297.
48. H. B. Mann and D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, *Annals of Mathematical Statistics*, **18**(1), 1947, pp. 50–60.
49. V. Devedžić and S. R. Milenković, Teaching agile software development: A case study, *IEEE Transactions on Education*, **54**(2), 2011, pp. 273–278.
50. V. Mahnic and A. Casar, A computerized support tool for conducting a Scrum-based software engineering capstone course, *International Journal of Engineering Education*, **32**(1), 2016, pp. 278–293.
51. P. Berander, Using students as subjects in requirements prioritization, *Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE'04)*, 19–20 August 2004, Redondo Beach, CA, pp. 167–176.
52. 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)*, Kaiserslautern, Germany, October 9–10, 2008, pp. 288–290.
53. T. Dybå and T. Dingsøyr, What do we know about agile software development?, *IEEE Software*, **26**(5), 2009, pp. 6–9.
54. P. Abrahamsson, K. Conboy and X. Wang, 'Lots done, more to do': the current state of agile systems development research. *European Journal of Information Systems*, **18**(4), 2009, pp. 281–284.
55. T. Dingsøyr, S. Nerur, V. Balijepally and N. B. Moe, A decade of agile methodologies: Towards explaining agile software development, *Journal of Systems and Software*, **85**(6), 2012, pp. 1213–1221.
56. S. Overhage, S. Schlauderer, D. Birkmeier and J. Miller, What makes IT personnel adopt Scrum? A framework of drivers and inhibitors to developer acceptance, *Proceedings of the 44th Hawaii International Conference on System Sciences*, Koloa, Kauai, HI, January 4–7, 2011, pp. 1–10.
57. V. Venkatesh and F. D. Davis, A theoretical extension of the technology acceptance model: four longitudinal field studies. *Management Science*, **46**(2), 2000, pp. 186–204.

58. S. Overhage and S. Schlauderer, Investigating the long-term acceptance of agile methodologies: an empirical study of developer perceptions in Scrum projects, *Proceedings of the 45th Hawaii International Conference on System Sciences*, Wailea, Maui, HI, January 4–7, 2012, pp. 5452–5461.

**Viljan Mahnič** is a Full 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.

**Tomaž Hovelja** is an Assistant Professor at the Faculty of Computer and Information Science of the University of Ljubljana, Slovenia. His research areas are social, economic and organizational factors of IT deployment in enterprises and IT projects success criteria. He received his Ph. D. in Economics from the University of Ljubljana in 2006.