Studying agile software estimation techniques: the design of an empirical study with students

Marko Poženel & Viljan Mahnič

University of Ljubljana
Ljubljana, Slovenia

ABSTRACT: Empirical studies with students (ESWS) are often viewed sceptically because of questionable external validity. However, if conducted in an adequate way, they can provide at least some preliminary evidence regarding new processes, methods and tools that are constantly proposed for possible use in software development. In order to maximise research and pedagogical value, the requirements that research and pedagogy place on valid ESWSs can be found in the scientific literature. In this article, the design of an ESWS is described that strictly follows a checklist for integrating ESWSs with research and teaching goals. The aim of the study is to compare two software effort estimation techniques: planning poker and the team estimation game. For each checklist item, there is a description of how it is considered in the study design. It is shown that the study can be incorporated into the software engineering capstone course seamlessly, without hindering the achievement of teaching goals. Since there is almost no evidence about the team estimation game in the scientific literature, it is expected that the study will help filling this gap and provide valuable results for researchers and industry.

Keywords: Software engineering education, empirical studies, capstone course, agile software development, Scrum, planning poker, team estimation game

INTRODUCTION

New processes, methods and tools are constantly proposed for possible use in software engineering practice. In order to take full advantage of their strengths and to minimise the risk that their introduction will fail, these new proposals should be carefully evaluated before they are actually used in an industrial environment. However, running empirical studies in industrial settings often requires a great deal of time, effort and resources; therefore, few software organisations can afford such an endeavour. This problem can be alleviated by conducting empirical studies with students as subjects (ESWS).

Although ESWSs are often viewed sceptically by researchers and practitioners because of questionable external validity, such studies - if conducted in an adequate way - can significantly contribute to empirical evaluation of new processes and tools. For example, ESWSs can provide at least some preliminary evidence that supports (or refutes) research hypotheses and serve as a basis for planning more rigorous subsequent studies. According to Carver et al, appropriately designed ESWSs not only benefit researchers and industry, but can also be beneficial for students and instructors [1].

By participating in ESWSs, students get hands-on experience with state-of-the-art topics, while instructors are encouraged to introduce less conventional ways of teaching based on project work and practical problem-solving. Additionally, empirical software engineering can be introduced as a part of the software engineering curriculum.

In order to maximise research and pedagogical value, Carver et al outlined the requirements that research and pedagogy place on valid empirical studies with students, and developed a checklist that provides guidance for researchers and educators when planning and conducting such studies in university courses (Table 1) [2]. The checklist consists of 10 items that are divided into four groups describing activities that must be performed during the study timeline, i.e. before the class begins, as soon as the class begins, when the study begins and when the study has been completed.

In this article, the authors describe how the aforementioned checklist was used to design an ESWS in the Faculty of Computer and Information Science at the University of Ljubljana, with the aim of evaluating two of the more widespread agile estimation techniques, i.e. planning poker [3] and the team estimation game [4]. At the time of this writing, the study is being conducted within the scope of the software engineering capstone course that requires students to work in teams in order to develop a project strictly following Scrum [5][6].

Table 1: ESWS checklist [2].

Before the class begins	
1	Ensure adequate integration of the study into the course topics.
2	Integrate the study timeline with the course schedule.
3	Reuse artefacts and tools as appropriate.
4	Write up a protocol and have it reviewed.
As soon as the class begins	
5	Obtain subjects' permission for their participation in the study.
6	Set subject expectations.
When the study begins	
7	Document information about the experimental context in detail.
8	Implement policies for controlling/monitoring the experimental variables.
When the study is completed	
9	Plan follow-up activities.
10	Build or update a laboratory package.

While planning poker has been investigated in several studies (e.g. [7][8]) and is well described in the literature [9], the authors were not able to find a systematic study on the team estimation game except strong recommendations from the part of some agile practitioners suggesting that the team estimation game is a better estimation technique (e.g. [4][10]). Therefore, it is expected that this study will contribute to existing body of knowledge by providing an empirical comparison of both.

The article is organised into four sections as follows. The two estimation techniques studied are described first, followed by a brief outline of the software engineering capstone course at the University of Ljubljana. Then, a detailed description of the corresponding ESWS is provided, clearly explaining how each checklist item is considered in its design. Finally, the more important conclusions are provided.

PLANNING POKER AND TEAM ESTIMATION GAME

Scrum assumes that each user requirement is formulated as a user story consisting of three parts: a written description (used for planning and as a reminder), conversations about the story (to flesh out the details) and acceptance tests (to determine when a story is *done*). In order to create a project plan, each user story must be estimated in story points, which represent its relative complexity in comparison with other user stories [9].

When using planning poker (PP), user stories are estimated at an estimating session by members of the development team responsible for implementation. They take user stories from the Product Backlog one by one. For each story, the Product Owner begins by explaining its requirements. In turn, the team discusses the work involved, posing questions to the Product Owner as needed. When the story has been fully discussed, each estimator privately estimates the required effort by writing the corresponding number of story points on a paper note card or choosing a card with the corresponding predefined value. All cards are revealed simultaneously in order to ensure the independence between different team members. If the estimates differ too much, the estimators discuss their estimates. The high and low estimators in particular should share their reasons. After discussion, the team re-votes on their estimates by revealing again all cards at the same time. The process is repeated until consensus is achieved.

Team estimation game (TEG) is a relatively new estimation technique that (to the best of our knowledge) has not yet been studied in detail. It starts from the premise that the estimation of a large Product Backlog using PP can be tedious and time-consuming task. Therefore, the goal of TEG is to balance the effort and time spent on user story estimation with the delivered value. Additionally, it is suggested that TEG better considers effort proportions among different user stories by providing the *whole picture*, while PP estimates each user story in isolation. TEG consists of two stages:

- Arranging user stories into columns relative to required effort to complete;
- Assigning each column the corresponding number of story points. A large enough playing surface for user story cards arrangement is needed.

Similar to PP, the team members take user stories from the Product Backlog one by one. The first team member selects the first card from the Product Backlog and places it on playing surface. Then, the next team member selects a card and places it on the surface relative to previously placed card. Less complex user stories are placed left to the first card, while more complex are placed right. If the user story complexity is the same, its card is placed in the same column as the first card. In general, each team member can either: a) select a new user story card from the Product Backlog and place it relative to other cards on the playing surface; b) move one of the user story cards already on the playing surface (if he/she thinks that its current relative position is incorrect); and c) pass (if he/she agrees with relative order of cards and there are no more user stories to estimate). Team members repeat these steps until there are no more user story cards in the Product Backlog and all team members select *pass* (no team member wishes to move a card).

In such a way, the team members can easily follow how columns of cards with the same complexity are formed on the playing surface. After all cards have been placed, values are assigned to each column representing the effort estimates of corresponding user stories.

THE CAPSTONE COURSE AT THE UNIVERSITY OF LJUBLJANA

The capstone course at the University of Ljubljana teaches students agile software development, in particular Scrum, through practical team-based project work. The course lasts 15 weeks and is taken by undergraduate computer science students in their last (sixth) semester. Students are required to work in groups in order to develop an (almost) real project on the basis of user requirements provided by a domain expert playing the role of the Product Owner.

The course design is based on the Scrum framework and 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. The rest of the course is divided into three regular Scrum Sprints (called Sprint 1, Sprint 2 and Sprint 3), each lasting four weeks.

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. These three weeks are also used to acquaint students with the initial Product Backlog, containing a set of prioritised 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 and prepares the release plan.

Sprints 1, 2 and 3 are regular Scrum Sprints that have the same structure. Each Sprint starts with a Sprint planning meeting at which student teams negotiate the contents of the next iteration with the Product Owner, and develop the initial version of the Sprint Backlog. During the Sprint, the teams have to meet regularly at the Daily Scrum meetings and maintain their Sprint Backlogs, while 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 students and instructors meet to review the development process in the previous Sprint, giving suggestions for improvements in the next. A more detailed description of the course can be found in earlier publications by the second author [5][11].

STUDY DESIGN

In order to study differences between PP and TEG, the ESGS is designed and incorporated into the capstone course in accordance with the checklist proposed by Carver et al [2]. The implications of each checklist item on the ESWS design are discussed in detail in the following subsections.

Ensure Adequate Integration of the Study into the Course Topics

In order to comply with requirements of this checklist item, it is important that a) the students who participate in an ESWS possess the necessary skills (which makes the subject population more similar to the target population and improves external validity); and b) that the research goals fit into the class materials (which means that the research goals do not jeopardise the teaching goals).

Considering the first of the aforementioned issues, the study is designed to be conducted with students of the software engineering module and take place in the last semester of their studies, when the students have already mastered traditional methods of software development, fundamentals of data bases and information systems, and basics of software project management in previous courses. Additionally, adequate training in Scrum and user story estimation is provided during Sprint 0, including a special estimating session during which the students practise PP and TEG.

With regard to the compatibility between research and teaching goals, special care should be taken to ensure that pursuing the research goal (i.e. comparison of accuracy of PP and TEG estimates) does not in any way hinder the achievement of teaching goals (i.e. teaching agile software development with Scrum in (as much as possible) a real environment. User story estimation and measuring the amount of work expended are regular activities in software development projects; therefore, the execution of these activities does not require any additional effort on the part of the students except careful recording of collected data. Moreover, it can be argued that devoting more attention to user story estimation contributes to the development of estimating skills among students, which several studies have shown to be often inadequate [6][12].

Integrate the Study into the Course Schedule

Given the fact that user story estimation and decomposition of user stories into pertaining tasks are regular parts of each Scrum project, integration of the study into the course schedule was not a problem. Scrum requires the user stories to be estimated at the end of Sprint 0 (in order to produce a rough release plan) and at the beginning of each regular Sprint (in order to develop the Sprint Backlog). Within the Sprint Backlog, each user story must be further decomposed into constituent tasks and the amount of time required for completing each task must be estimated.

Following the aforementioned Scrum rules, the points in the student projects' time-scale were defined, at which the required data had to be collected. Since the authors wanted to analyse how estimates improve when students obtain more practice and knowledge about the project domain, the ESWS design envisions collection of user story estimates at the end of Sprint 0 and at the Sprint planning meeting of each regular Sprint. In order to analyse how much the estimates in story points differ from the sum of estimates of pertaining tasks, the task estimates must be recorded when the Sprint Backlog is created during the second part of the Sprint planning meeting. Finally, in order to compute the time actually spent for each user story implementation, the work expended is recorded at each daily Scrum meeting.

Reuse Existing Artefacts and Tools as Appropriate

The ESWS design incorporates a great deal of authors' past experience, since the capstone course has already served as a basis for several similar studies [13]. A special computerised support tool that the authors have developed for managing students' projects is used to facilitate empirical data collection [11]. The tool fully automates PP and TEG, the decomposition of user stories into constituent tasks, as well as recording of work expended.

When playing planning poker, all members of a student team are presented with a screen containing the user story that is going to be estimated and a set of predefined values that are usually used for estimation. Each team member provides his/her estimate by simply clicking the corresponding predefined value (or *custom* if he/she wants to enter a value not in the list). When all estimates are submitted, the ScrumMaster ends the round and the results are displayed on the screens of all team members. This process is repeated until consensus is reached. For the purpose of the study, the tool automatically records individual estimates provided in each round, as well as the final consensus estimate.

In a similar way, the tool automates the team estimation game: the team members are presented with a drop-down list of stories that need to be estimated, and the playing surface on which the virtual story cards have to be arranged according to their relative size. The result of each move is displayed on the screens of all team members, so that they can easily follow how columns of cards with the same complexity are formed on the playing surface. In order to make it possible to reconstruct the game for research purposes, each move is recorded in the tool's database.

During the decomposition of user stories into constituent tasks, the estimate of effort required for each task completion is recorded. When a team member starts working on a task, he/she simply clicks the corresponding s*tart work* button on his/her task list and the tool measures the time elapsed until the *stop work* button is clicked or the team member starts working on another task. The time elapsed is automatically saved in the tool's database.

Write Up a Protocol and Have It Reviewed

A protocol describing a set of steps to follow was developed on the basis of the Scrum framework and reviewed by the teacher and his assistants who are involved in the course in both roles, i.e. as instructors and researchers.

According to the protocol, the following activities take place in Sprint 0: classical lectures are used to acquaint students with Scrum, user stories, PP and TEG. For the purpose of the project work, the students are asked to form teams of four. During laboratory classes, they are trained in managing their projects using the aforementioned computerised tool.

A special laboratory session is devoted to practising PP and TEG. After being acquainted with the Product Backlog of the project, they are going to develop, the students are asked for consent for their participation in the study. The teams that participate in the study are divided into two groups of equal size. The first group estimates user stories using PP, while the second group uses TEG.

At the Sprint planning meetings of Sprints 1, 2 and 3, the teams are required to re-estimate unfinished stories, decompose the stories into constituent tasks and estimate the effort required to complete each task. During each Sprint, the students must record the amount of work, and daily Scrum meetings are used to monitor whether all required data are stored in the tool's database.

At the end of the course, an anonymous survey among students, and statistical analysis of the data collected, are planned. It is envisioned that the survey results will be presented to the students during the last lecture. Afterwards, conversations with students will take place in order to obtain further understanding of their opinions and to corroborate the validity of the study results.

Obtain Subjects' Permission for Their Participation in the Study

According to the protocol, students' permission for their participation in the study is sought after they have obtained sufficient information about the required methodology (Scrum, user stories, PP, TEG) and the project they are going to develop (i.e. the Product Backlog). Their participation is completely voluntary. When asking for permission, the authors stress that there is no difference in the capstone course execution and the amount of work required between the teams that participate in the study and those that do not. The participating teams are only asked to execute all the required activities as conscientiously as possible and carefully record the amount of work expended.

Since Slovenian law does not prescribe a written agreement between the instructor/researcher and the participating students, and the study does not involve sensitive personal data, no special consent form was devised. Consequently, students' participation is based on an oral agreement between instructors and students.

Set Subject Expectations

As stated above, students' participation in the study is planned in such a way that it does not require any additional effort, but careful execution of activities are prescribed for all students attending the course. In order to increase motivation, each participating student is promised a grade incentive of up to 10 additional points, which means one grade more than he/she would have received otherwise. Students are informed that the grade incentive depends on how well they will follow the prescribed process, and the quality of data they will provide.

The study design envisions that at the beginning, the students are only roughly informed about the goals of the study. Therefore, their expectations are mostly focused on teaching goals, i.e. to learn agile software development, in particular Scrum, through practical project work that will require them to integrate previously learned material and apply their knowledge and skills in a realistic simulation of professional experience. Details of the study are not disclosed because they could bias the results. For example, the students could deliberately manipulate the results by reporting the amount of effort expended to be equal to the initial estimate or unconsciously try to please the researchers by trying to confirm the research hypotheses.

Document Detailed Information about the Experimental Context

The experimental context is documented in detail in a separate document that contains all elements prescribed by Carver et al [2] and can serve as a basis for possible study replication. A short summary is provided in Table 2.

Table 2: Summary of the study context.

Subjects	Students of the 3rd year of computer science with a major in software engineering.
Goals of the course	Teaching agile software development through practical project work on an industry relevant project; development of transferrable skills, particularly teamwork.
Topics covered	Scrum, user stories, software effort estimation, planning poker, team estimation game, balanced relative error and balanced relative error bias.
Teaching method used	Three weeks of formal lectures and preparatory activities (Sprint 0). Twelve weeks of practical project work providing students with a realistic simulation of professional experience.
Story point definition	A story point corresponds to one working day (6 hours of effective work).
Research questions	1. Does TEG take less time than PP? 2. Are TEG estimates less optimistic than PP estimates? 3. Are TEG estimates more accurate than PP estimates? 4. Does the estimation accuracy improve from Sprint to Sprint? 5. Are effort estimates after decomposition of user stories into tasks more accurate than the estimates obtained before decomposition?

Implement Policies for Controlling/Monitoring the Experimental Variables

In order to ensure accuracy and invasiveness of data collection, the study is incorporated within the project work seamlessly without hindering students' work on their projects. Effort estimation and decomposition of user stories into tasks take place during laboratory sessions under the supervision of teaching assistants and in the presence of the Product Owner. If necessary, the Product Owner provides details regarding each user story implementation. Regular Daily Scrum meetings serve for monitoring data collection of work expanded and allow instructors to immediately notice possible shortcomings. The aforementioned computerised support tool automates data collection; thus, allowing the students to focus on their projects instead of worrying how to record the required data.

Plan Follow-up Activities

A questionnaire was developed to obtain students' opinions at the end of the course, and the study results will be presented to students as described in the subsection about the study protocol. For research purposes, accuracy of estimates will be calculated using balanced measure of relative error. Possible differences between PP and TEG estimates will be analysed using well known statistical methods, such as independent samples *t*-test and Cohen delta.

Build or Update a Laboratory Package

Experience from running the study for the first time will serve as the basis for upgrading the study protocol into a laboratory package that will make it possible to replicate the study and could be readily reused in a professional environment.

CONCLUSIONS

The design of an ESWS was presented that strictly follows recommendations from the literature. For each checklist item proposed by Carver et al [2], a description was given of how the proposed ESWS would fulfil its requirements in order to provide valuable results that could benefit all parties involved: researchers, teachers, students and the industry. It was shown how the study fits into the course seamlessly without hindering students' work on their projects and jeopardising the teaching goals.

The proposed ESWS deals with the two software effort estimation techniques that are the most widely used in agile software development. While PP has been investigated in several studies, there is almost no evidence about TEG. Therefore, it is expected that the study described in this article will help filling this gap and provide at least preliminary evidence about possible TEG benefits. At the time of this writing, the study is being conducted within the scope of the Software Engineering capstone course at the University of Ljubljana.

REFERENCES

1. Carver, J.C., Jaccheri, L., Morasca, S. and Shull F., Issues in using students in empirical studies in software engineering education. *Proc. 9th Inter. Software Metrics Symp.*, Sydney, Australia, 239-249 (2003).
2. Carver, J.C., Jaccheri, L., Morasca, S. and Shull F., A checklist for integrating student empirical studies with research and teaching goals. *Empirical Software Engng.*, 15, 35-59 (2010).
3. Grenning, J., Planning Poker or how to Avoid Analysis Paralysis while Release Planning (2002), 16 May 2016, http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf
4. Johnson, H.L., How to Play the Team Estimation Game (2012), 16 May 2016, http://www.agilelearninglabs.com/2012/05/how-to-play-the-team-estimation-game/
5. Mahnic, V., A capstone course on agile software development using Scrum. *IEEE Trans. on Educ.*, 55, **1**, 99-106 (2012).
6. Mahnic, V., Teaching Scrum through team-project work: students' perceptions and teacher's observations. *Inter. J. of Engng. Educ.*, 26, **1**, 96-110 (2010).
7. Moløkken-Østvold, K., Haugen, N.C. and Benestad, H.C., Using planning poker for combining expert estimates in software projects. *J. of Systems and Software*, 81, 2106-2117 (2008).
8. Mahnic, V. and Hovelja, T., On using planning poker for estimating user stories. *J. of Systems and Software*, 85, **9**, 2086-2095 (2012).
9. Cohn, M., *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall (2006).
10. Björsne, T., Kostial, I. and Schmatz, K-D., Die Alternative zum Planning Poker: Das Team Estimation Game, *ProjektMagazin*, Ausgabe 02/2013, 16 May 2016, https://www.projektmagazin.de/artikel/die-alternative-zum-planning-poker-das-team-estimation-game_1077797#cut-off
11. Mahnic, V. and Casar, A., A computerized support tool for conducting a Scrum-based software engineering capstone course. *Inter. J. of Engng. Educ.*, 32, **1A**, 278-293 (2015).
12. Mahnic. V. and Hovelja, T., Teaching user stories within the scope of a software engineering capstone course: Analysis of students' opinions. *Inter. J. of Engng. Educ.*, 30, **4**, 901-915 (2014).
13. Mahnic, V., The capstone course as a means for teaching agile software development through project-based learning. *World Trans. on Engng. and Technol. Educ.*, 13, **3**, 225-230 (2015).

BIOGRAPHIES

Marko Poženel is a Teaching Assistant in the Faculty of Computer and Information Science at the University of Ljubljana, Ljubljana, Slovenia. His teaching and research interests include agile software development methods and empirical software engineering, as well as Web data mining and user behaviour analysis. He received his PhD in Computer Science from the University of Ljubljana in 2010.

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