Teaching Scrum to Software Engineering Students with Virtual Reality Support

Guillermo Rodríguez[1,2], Alvaro Soria[1], Marcelo Campo[1,2]
[1] ISISTAN, Research Institute - UNICEN- Argentina
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
{grodri, asoria, mcampo}@exa.unicen.edu.ar

Abstract. Scrum has received significant academic attention because of its widespread application in software development industries. Teaching agile software development can be illustrated by teaching with lecture-based classes. However, involving students in a real software environment is a trendy alternative which fosters their engagement. Software engineering graduates are not appropriately prepared for applying their skills in a real software project. Thus, we focus on teaching and integrating teamwork-oriented skills in a real software development environment based on Scrum. In this work, we present *Virtual Scrum (VS)*, a virtual reality environment that assists students with the running of a software project following the Scrum framework. *VS* supports artifacts needed for carrying out Scrum meetings and media-based tools to achieve permanent communication among Scrum members. A survey of students, who used the tool in a Software Engineering (SE) course, showed that *VS* is helpful to exercise Scrum practices.

Keywords: Virtual Reality, Software Engineering, Scrum

1 Introduction

The increasing use of Agile Methods (AM) for software development has reoriented the education of future computer and information science engineers. AMs emerge as a feasible alternative to teach software engineering practices in order to prepare students to face problems associated to current software development. Teaching AM offers students opportunities to put theoretical concepts into practice [2].

Out of the various agile software development approaches, Scrum and Extreme Programming are commonly selected to be taught in software engineering courses. Particularly, Scrum has gained wide acceptance because it concentrates on managing software projects and includes monitoring and feedback activities [3]. These features allow students to acquire skills beyond technical and scientific scenarios, such as teamwork-related abilities. Thus, these aspects are welcome for SE teachers because they enable students to get acquainted with agile methods and, at the same time, provide mechanisms for evaluating individual agile concepts. Several works [5, 6, 7] have aimed to teach a few practices of software engineering like programming, testing, estimating and planning techniques. However, they have not considered the implementation of a teaching environment. Other approaches have stated Scrum covers

a big number of good software engineering practices [8, 9]. For this reason, Scrum seems to be the most suitable alternative of AM to obtain a high level of coverage of software engineering practices. So far, there is not much evidence about teaching Scrum. Mahnic (2010) taught Scrum through a capstone project and described the course details, student perceptions and teacher observations after the course [2]. However, there is a lack of approaches to teach Scrum involving students in real software development. Scrum provides good software engineering practices.

To effectively learn how to work with Scrum, students have to setup a Scrum-based Team Room. This way, students are introduced to the use of tools and techniques for supporting different Scrum roles within a capstone project. However, setting up a Team Room for each group of students in a Software Engineering course is not always possible due to the cost of the required didactic material and, more important yet, the availability of university facilities.

In this context, we find virtual worlds highly suitable for helping students to feel familiar with development contexts by viewing software artifacts which are naturally intangible. These 3D environments provide a physical topology of the structure of information in a software project, which would allow accessing the project information [4]. For instance, one objective is to have the information readily available, visualize the traceability of requirements, and be aware of team progress. In this context, the artifacts are always shared in an integrated development environment. As a consequence, virtual worlds have to be as close to real life as possible by exploiting the 3D metaphor.

In this work, we present *Virtual Scrum*, a virtual world that provides students with a platform in which they can experience several aspects of the Scrum framework to prepare them for their future software development jobs and consequent exposures to the software process. To evaluate how much *Virtual Scrum* helps students in different aspects of the Scrum framework, we conducted a case study with students of a Software Engineering course at UNICEN University of Argentine. To record the perception of students of this study, we designed a questionnaire to collect students' opinions with respect to tool performance and usability. The results showed that there are some positive and some negative effects by introducing *Virtual Scrum*. The most notably positive effect is that students are clearly motivated to use the tool. However, there are some issues to improve which affect the tool performance, such as avatar movement controls, little friendly graphic user interface and tool deployment. Those issues were considered by students in the questionnaire because they affected tool usability.

The remainder of the paper is organized as follows. Section 2 reports background on collaborative work in software development. Section 3 gives an overview of *Virtual Scrum*. Section 4 describes experimental results. Finally, Section 5 concludes this research and identifies directions for future work.

2 Background

Scrum is a methodology that organizes projects into small, self-organized and cross functional teams, called Scrum Teams. Work is organized and prioritized according to

Product Backlogs. The backlog items are called *user stories*. These user stories are grouped into a serie of short iterations called Sprints. During the Sprint, a management representative, called Scrum Master, enforces Scrum practices and helps the team to make decisions or acquire resources as needed. A brief meeting held daily communicates to the team the work that has been accomplished, what work needs to be done, and asks for input concerning impediments that the team sees in reaching the goal. A planning session is held that defines backlog items to be grouped into a Sprint. Sprint Retrospectives in which team members discuss issues, accomplishments, and lessons learned about the previous Sprint are held [3]. Customer input is highly sought, and customers may become part of the team. The customers must be able to provide the time and energy required to produce a good product. Working, intermediate products are verified by customer or verified internally in increments; that is, as pieces or parts of the entire final product. These intermediate products are delivered in whole at the end of the entire development cycle. Each delivery builds more and more functionality into the product. It should be noted that this incremental development approach relies on brevity, i.e. brief meetings, brief planning sessions, and brief customer commitment time periods. It also relies on frequency, i.e. frequent meetings, frequent planning and re-planning, and frequent customer interaction [18].

In order to support Scrum practices, tools for planning, meetings, information visualization and requirement management are needed. There are some 2D tools for collaborative development in agile methods like CollabNet TeamForge[1] and ScrumWall[2] which allow developers to visualize and develop software artifacts in a distributed environment. However, those tools permit neither virtual interaction nor meetings among developers.

The advent of better Internet connections and more powerful computers have encouraged the creation of Virtual Worlds in which people get together and interact. Massive multiplayer online games (MMOG), like Second Life[3] and Wonderland[4], illustrate this trend.

In the light of the above, a 3D virtual environment allows developers to know about tasks performed by their peers, and even hold meetings regardless their physical location. Also, a 3D environment can be used to provide a physical topology of information structure in a software project, which allows a faster access to information [15]. Thus, these platforms are profitable to setup meetings and interact since people do not have to go one place to another.

In addition, several tools exploit collaborative integration features to provide group awareness and communication. For example, one aim is to have the information always available through a configuration management platform. In that context, artifacts are always shared in an integrated development environment [10]. In short, a 3D environment helps developers get familiar with the development context by visualizing software artifacts which are naturally intangible [16].

[1] http://www.collab.net
[2] http://www.scrumwall.com
[3] http://www.secondlife.com
[4] http://java.net/projects/wonderland

As a consequence, 3D virtual environments arise as viable solution to hold meetings. Nowadays, the best known virtual world is Second Life from Linden. Second Life is not a game, but a general purpose virtual world. It provides a Software Development Kit (SDK) in which people are able to create elements inside the world and design avatars to interact with those elements. Nevertheless, so far, this virtual world does not provide support for holding software meetings.

On the other hand, Wonderland, because of being Open Source, provides a toolkit that allows users to adapt the world to what they need. Wonderland project is totally developed in Java and freed under GPL license to create 3D collaborative virtual worlds. Once users connect to environment, they are capable of communicating through high fidelity audio, sharing desktop applications and documents, and holding real business.

In 2009, Wonderland presented a virtual world oriented to software development [14]. However, this tool is mostly focused on teaching agile methods rather than supporting software development and team member interaction.

Also, [12, 13] have worked on using virtual worlds to hold meetings but not software meetings. Thus, those environments were built without considering software artifacts related to a methodology. Furthermore, exploiting 3D metaphors for incorporating software artifacts to support meetings along the Scrum life cycle seems to be a limitation in existing virtual worlds.

3 Virtual Scrum

In order to introduce Scrum into a teaching context, teachers need to provide planning cards, whiteboards, and other artifacts so as to support the practices proposed by Scrum. So, we have developed *Virtual Scrum*, a groupware that provides support to the running of a software project following Scrum. Fig. 1 shows the artifacts of *Virtual Scrum* into the Scrum framework. The process starts with the load and prioritization of the Product Backlog, which is represented by *Virtual Product Backlog* artifact. This artifact allows students, who play the role of Scrum Team, to load user stories and also supports the prioritization process between the team and the Product Owner, who is the professor. At the beginning of each Sprint, students select the user stories to be developed during the iteration. These user stories are loaded into the Sprint Backlog represented by *Virtual Sprint Backlog* artifact which shows user stories of the current Sprint.

For the preparation of the Sprint, students estimate the user stories by using the *Virtual Planning Poker* artifact as shown in Fig. 2 (b). This artifact provides a mechanism for voting based on Planning Poker technique [1]. At the end of the activity, the team visualize the results through a shared board provided by the artifact.

During the Sprint, students use the *Virtual Spring Backlog* to set a color to each user story so as to keep track of its progress status. Red indicates a user story is not started (TO DO). Yellow denotes the user story is being performed (DOING). Green shows that the user story is completed (DONE).

At the different stages of the Scrum life cycle, the tool supports virtual meetings. The meetings play an important role in the communication among Scrum members in order to identify and plan corrective actions to possible issues or impediments to the development process. Also, the artifacts used in the meetings are very important to visualize and remember what has been discussed and analyzed. For example, it is helpful to use post-it notes on a board, sketches on flip charts, and writings on the whiteboard to support discussion and argumentation.

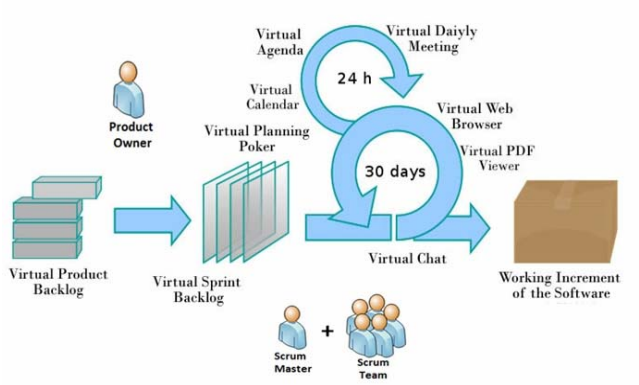

Fig. 1: *Virtual Scrum* in the Scrum life cycle

However, when a meeting is held among members physically distributed, those tools become obsolete. Thus, high fidelity tools able to support collaborative and distributed work are required. As a consequence, *Virtual Scrum* emerges as an alternative to maintain and visualize artifacts used during the meeting. This yields visible benefits not only for students, who no longer will need to attend the course, but also for Universities, which will save on resources to carry out their software engineering courses. Fig. 2 (a) shows the inside of *Virtual Scrum*, in which 4 members physically distributed are sharing a virtual room through an avatar representation.

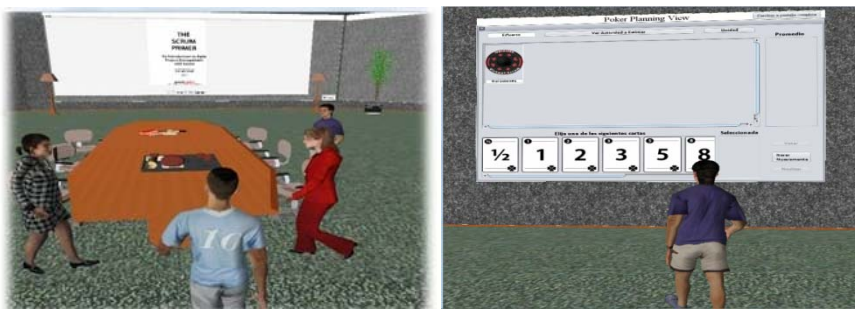

Fig. 2: Inside *Virtual Scrum:* (a) Meeting among Scrum members. (b) Virtual Poker Planning.

In order to hold a software meeting, several artifacts are mandatory to ensure that most software engineering practices are met. Thus, the *Virtual Daily Meeting* artifact is used by students to hold Daily Meetings. *Virtual Scrum* provides students with the *Virtual PDF Viewer* artifact to present reports or pictures, the *Virtual Browser* artifact to surf on the Web, and the *Virtual Calendar* to schedule events notified by the *Virtual Agenda*. In addition, a chat with videoconference support is provided through the *Virtual Chat* artifact.

During a daily meeting, the team reviews the estimations and considers the lessons learned so as to adjust the development velocity for the next Sprints. The Scrum Master, who is also a student, is responsible for starting the daily meetings taking into account the 15-minute time-boxing period. Within this period, the members of the Scrum Team answer the following questions: What have you done since yesterday? What are you planning to do today? And do you have any problems that would prevent you from accomplishing your goal? This information is used during the retrospective meeting at the end of each Sprint.

To sum up, by using *Virtual Scrum*, students will get a concrete picture of how to practically setup a suitable Team Room to implement the software practices of the Scrum framework and also to learn the different aspects related to the management of a software project in an agile methodology.

4 Experimental results

The experiment focused on evaluating how much *Virtual Scrum* helps students in different aspects of the Scrum framework. Thus, we carried out a case study with 8 students of last year of System Engineering at UNICEN University of Argentine.

Students were of both sexes and were between 21 and 25 years old. They were given some lectures of Scrum in the context of Software Engineering course of the System Engineering BSc program at the Faculty of Exact Sciences (Department of Computer Sciences-UNICEN). Each student had a personal computer in order to simulate a geographical distribution. Before the experiments, we held a training course for the participants so as to provide them with the skills to interact with each artifact of the virtual world and its relationship with the Scrum framework. The experiment consisted in building a web-based system to manage purchase and sale of products so as to perform all the Scrum activities such as building the Product Backlog, estimating and planning the user stories, holding Daily Meetings and observing task progress. Professors, who played the role of Product Owner, gave students a list of 9 user stories. To evaluate the perception of the tool, we used a questionnaire based on Likert's approach [17] so as to collect student opinions about *VS* artifacts. The questionnaire items were statements based on a 3-point scale in which the students could either totally agree, agree, or disagree.

4.1 Analysis of student performance in Scrum process

We observed students using *VS* artifacts. Particularly, we focused on estimating and disaggregating user stories. Most *VS* artifacts were suitable for discussing and

supporting argumentations in user story estimation. Table 1 summarizes the user stories loaded in the *Virtual Product Backlog* artifact, the estimations in days for each user story obtained from the *Virtual Poker Planning* artifact, and the *Earned Value* (ER) for each user story. ER refers to the ratio between remain work decrease and amount of work done. Ideally, the remain work decrease between days d_1 and d_2 once Sprint started should be grater or equal to the amount of work done during this period. Thus, the goal value of this metric is greater or equal to 1. However, values much greater than 1 mean a poor planning. Equation (1) shows how to calculate ER, where ER_{dj} is the ratio between the amount of work done for the task *j* during the day *i*, $WS_{i,j}$, with *i=1, 2, ..., d-1* and the total work $WS_{i,j} + WR_{d,j}$ (done and remain) [11].

$$ERd,j = \frac{\sum_{i=1}^{d-1} wsi,j}{\sum_{i=1}^{d-1} wsi,j + WR\,d,j} \quad (1)$$

Table 1. Estimation and earned values of user stories.

User Story	Execution time (in days)	Initial Estimation (story points)	ER_{tarea}
US 01	14	50	0,25
US 02	14	10	0,35
US 03	7	150	0,71
US 04	14	100	0,65
US 05	14	100	0,35
US 06	14	5500	0,35
US 07	14	100	0,43
US 08	14	100	0,35
US 09	7	100	0,55

For simplicity, user stories (USs) were comparable to each other, had the same level of granularity and began to be developed the same day. The USs were estimated to be ready within 1 to 2 weeks. The story point column of Table 1 shows the relevance of the US. For example, story points of *US 06* were high because students did not have enough knowledge about web front-end technology, which was a high priority requirement to the Product Owner.

When students reached the building phase, the execution days shown in Table 1 were more than they had estimated. Also, ER shows the percentage of completeness of each US in which the values for *US 01* and *US 02* show that students underestimated the importance of those features because they estimated few story points and they obtained low ER in that USs. During the retrospective session, students realized that all the estimations made with the Poker Planning technique did not reflect the complexity of the system. Then, they considered that the reason for underestimated and overestimated USs was the lack of experience in the technology and in the planning technique. To make matters worse, they did not acquire enough information from the Product Owner because they considered it unnecessary. In this light, we believe that the major obstacle the students faced was the lack of assistance in the decision making process, especially when they preformed the estimation of the user stories.

4.2 Questionnaire evaluation of *Virtual Scrum* perception

To better prepare the students to fill out the questionnaire, we included some general "warming up" questions placed at the beginning of the questionnaire, asking, for example, what Scrum is and how software projects actually benefit from it. Then, we added several query items designed to collect the students' opinions with respect to performance of *Virtual Scrum* according to decision making and coordination; usability; quality of tool features, and commitment.

According to Fig. 3, most students agreed that discussion processes were coordinated and efficient. Also, most students were satisfied with the meeting artifacts supported by the tool. Nevertheless, regarding decision making, the results showed that the amount of students who agreed to effectiveness of the decision making process was nearly the same as the amount of students who disagreed. So, it is necessary to work on decision making assistance related to user story disaggregation, estimating, prototyping, etc. Also, some issues related to usability caused problems in communication as it is shown in Fig.4.

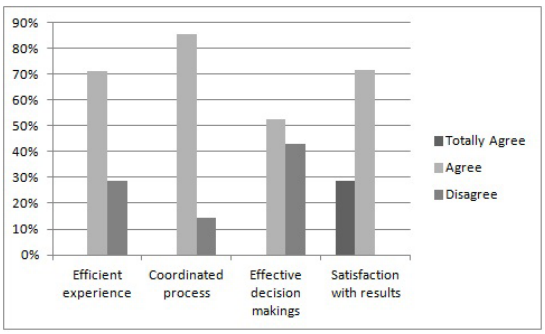

Fig. 3: Opinion about decision making and coordination supported by the tool

As regards tool usability, Fig. 4 shows there are some issues that affect the use of the tool like avatar movement controls and readability in the 3D scenario. On the contrary, most students agreed that face-to-face meeting can be replaced by *Virtual Scrum*. More than 80% of students will use the tool whenever it is not viable a face-to-face meeting.

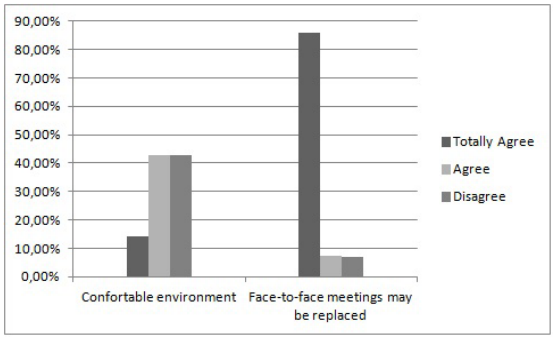

Fig. 4: Opinion about tool usability

Regarding tool features, Fig. 5 shows that more than 70% of students are satisfied with the quality of work supported by the tool. To measure the quality of work, we took into account the ease of students to interact with the tool. We concluded that problems in 3D interfaces do not affect considerably the Scrum process and member interaction. In addition, the figure shows high commitment to the solution obtained through the tool. Only less than 20% of students were little committed to the solution. Also, if usability improvements are implemented, there will be more commitment and responsible attitude of students in decision-making during meetings.

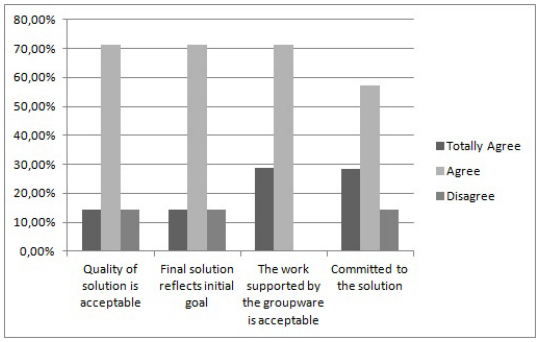

Fig. 5: Opinion about quality of tool features to support the process

5 Conclusion

In this work we presented a virtual world for teaching Scrum framework aiming at preparing students for future exposures to software engineering. *Virtual Scrum* represents a helpful tool for students to learn and exercise good software engineering practices. For instance, the Poker Planning technique is used to estimate efforts, the Product Backlog is used to manage user stories and Daily Meetings are used to perform peer reviews and progress checking.

The experiment carried out with students from last year of System Engineering at UNICEN showed that *Virtual Scrum* is also viable to support virtual meetings at different stages of Scrum life cycle during a real and controlled software development project. As a consequence, most students would use the tool again, and even replace a face-to-face meeting in case it cannot be held.

This finding provides an opportunity to use *VS* in a distributed software development context. Developers distributed all over the world make it difficult to apply Scrum due to the difficulty in coordinating a meeting and the lack of constant communication. Thus, *VS* emerged as an alternative to support virtual meetings in software environments using Scrum with members physically distributed.

However, there are some issues to be improved in order to enhance meeting performance like the interaction inside the tool and lack of traceability of real world software artifacts. On the other hand, there are a number of threats to validity in our evaluation of *VS*. The case-study was small to make generalization, even though it has shown positive feedback. In particular, students in the experiment had theoretical knowledge of Scrum but they had no real experience on it. Less than half of students were not involved in the software business. Regarding the size of the project, the case-study was performed by only one Scrum Team. Thus, we are interested in studying the coordination of multiple Scrum Teams in order to observe whether the tool is helpful to facilitate team coordination.

Finally, future work will be threefold: (1) to experiment with more students in larger software projects, (2) to integrate *VS* with software development tools like issue trackers and versioning systems, and (3) to apply profiling techniques to provide personalized assistance to students according to their observed problems such as decision making process at different stages of Scrum.

Overall, we concluded that *VS* helps students learn how to work on software development based on Scrum. In addition, *VS* provides effective communication among developers and coordination mechanisms among life cycle artifacts, in order to maintain the project synchronized. In this way, we confirmed the anecdotal evidence of Scrum as a framework used not only in the context of software development, but also in teaching software engineering.

References

1. Cohn, M.: Agile Estimating and Planning. Prentice-Hall (2006)
2. Mahnic, V.: Teaching Scrum through team-project work: students' perceptions and teacher's observations. International Journal of Engineering Education (2010)
3. Schwaber, K.: Agile Project Management with Scrum (2004)
4. Whitehead, J.: Collaboration in Software Engineering: A Roadmap. University of California, Santa Cruz (2007)
5. Kivi, J., et al.: Extreme Programming: A University team design experience. In Proceedings of 2000 Canadian Conference on Electrical and Computer Engineering, pp. 816--820 (2000)
6. Mahnic, V.: A case study on Agile Estimating and Planning using Scrum. Electronics and Electrical Engineering (2011)
7. Reichlmayr, T.: The agile approach in an undergraduate software engineering course project. In 33rd Frontiers in Education 2003, vol. 3, pp. 13--18 (2003)

8. Diaz, J., Garbajosa, J., Calvo-Manzano, J.: Mapping CMMI Level 2 to Scrum Practices: An experience report. Software Process Improvements, vol. 42, Springer Berlin Heidelberg, pp. 93--104 (2009)
9. Sutherland, J., Ruseng Jakobsen, C., Johnson, K.: Scrum and CMMI Level 5: The Magic Portion for Code Warriors. In Proceedings of the 41st Annual Hawaii International Conference on System Sciences (2008)
10. Herbsleb, J.: Global Software Engineering: The Future of Socio-technical Coordination. School of Computer Science. Carnegie Mellon University (2007)
11. Mahnic, V., Zabkar, N.: Introducing CMMI Measurement and Analysis Practices into Scrum-based Software Development Process. International Journal of Mathematics and Computers in Simulation (2007)
12. Nijholt, A., Zwiers, J., Peciva, J.: The Distributed Virtual Meeting Room Exercise. Proceedings ICMI 2005 Workshop on Multimodal multiparty meeting processing (2005)
13. Reidsma, D., et al.: Virtual Meeting Rooms: From Observation to Simulation. Journal AI & Society. London (2007)
14. Stockdale, R., Parsons, D.: Agile in Wonderland: Implementing a Virtual World Workshop Activity. 20th Australian Conference on Information Systems, Melbourne (2009)
15. Whitehead, J.: Collaboration in Software Engineering: A Roadmap. University of California, Santa Cruz (2007)
16. Teyseyre, A., Campo, M.: An Overview of 3D Software Visualization. IEEE Transactions on Visualization and Computer Graphics, vol. 15 (1), pp. 87--105 (2009)
17. Likert, R.: A technique for the measurement of attitudes, Arch Psychol 22 (1932)
18. Kulpa, M., Johnson, K.: Interpreting the CMMI. CRC Press (2008)