A Case Study of Using Scrum in Teaching Software Process

Minjie Hu
School of Business & ICT, UCOL
Palmerston North, New Zealand
m.hu@ucol.ac.nz

Sandra Cleland
School of Business & ICT, UCOL
Palmerston North, New Zealand
s.cleland@ucol.ac.nz

Aaron Steele
School of Business & ICT, UCOL
Palmerston North, New Zealand
a.r.steele@ucol.ac.nz

ABSTRACT

Much research has been published regarding the use of Scrum to teach the software development process since Scrum has emerged as a dominant approach among agile development methods. However, there is no common consensus on the teaching process, specifically how to deal with changes to the project as well as how to assess individual learning in a team-based project. This case study reports on the design of a teaching process that used Scrum in a second-year software process course for a web development task where requirement changes were going to be introduced during the project development. Although Scrum emphasises collaboration, the assessments included both individual and group assessments in order to let student to learn all the aspects in Scrum development process. The case study describes the student's perceptions to this course and concludes with lessons learned in this study.

Keywords: Scrum, collaboration, assessment

1. INTRODUCTION

Agile methods have been highly adopted by the software development industry (Al-Zewairi, Biltawi, Etaiwi, & Shaout, 2017). According to a recent State of Agile Survey, "Scrum and Scrum/XP Hybrid (68%) continue to be the most common agile methods used by respondents' organizations" (VersionOne, 2017). Much research has been published about using Scrum to teach software process since Scrum emerged as a dominant approach among agile development methods (Mahnic, 2015). However, there is no common consensus on the teaching process, specifically how to deal with changes to the project as well as how to assess individual learning in a Scrum-oriented project-based course.

This case study focuses on the design of a teaching process that used Scrum in a second-year software process course for a web development task in order to meet requirements changes during the project development. Although Scrum emphasises collaboration, the assessments include both individual and group projects allowing a student to learn all the aspects in the Scrum development process. Accordingly, our research questions are:

RQ1: *What do we need to teach students so they can quickly and effectively deal with requirement changes during the project development?*

RQ2: *How can we assess individual performance when teaching a Scrum-oriented project-based course?*

In this report, we firstly explore the literature about teaching Scrum in Section 2. In Section 3, we describe our course design in order to solve the problems in our research questions. Next, we discuss our findings from our teaching in Section 4. Finally, we conclude with lessons learned in this case study.

2. LITERATURE STUDY

In order to change conventional documentation driven and heavyweight software development processes, Agile methods were formally introduced in 2001 through the Agile manifesto, emphasising individual and interactions; working software; customer collaboration; and responding to changes (Beck, Beedle, Van Bennekum, Cockburn, 2001). Being highly adopted by software development industry, three commonly used agile methods from 2000 to 2015 were surveyed in a literature study (Al-Zewairi et al., 20017). These methods were extreme programming (XP), Scrum, and feature-driven development (FDD). The study concluded that all the agile methods pay attention to the improvement of requirements and testing stages rather than the coding stage. As the most commonly used agile method in industry Scrum has also been widely introduced into software engineering courses (Mahnic, 2015). In the following literature study, we focus on how Scrum has been taught and what needs to be improved in teaching Scrum.

2.1 How Scrum Was Taught

In order to adopt Scrum in a classroom setting in education, Delhij, van Solingen & Wijnands (2015) released a framework called eduScrum. Like Scrum, eduScrum has a similar definition, but emphasises that it's "difficult to master (because student teams have to do it themselves)" (Delhij, et al. 2015, p5). The framework added a new event document of personal reflection. It suggested that the teacher takes the role of Product Owner and a student fills the role of Scrum Master. A student team size of four is also recommended.

Nowadays, two approaches are commonly used in teaching Scrum. One is through student projects, another is through educational games. Teaching Scrum through student projects was revealed as the most widespread approach (Mahnic, 2015). Mahnic (2015) advocated that "*in order to learn Scrum the students do not need extensive lectures, but should try it in practice. Courses of this type mainly exploit the benefits of capstone projects that simulate professional working environment as much as possible*" (p78).

One of the early studies reported an experiment of teaching Scrum for 31 students through team-project work (Mahnic, 2000). The students were grouped into seven teams. Four teams worked on Project A for an internal course, developing a management tool for monitoring the process of Scrum-based software development. Three teams worked on Project B for an external software company, developing a simplified hospital information system. The teacher played the role of both customer and product owner for Project A, while people from

This quality assured paper appeared at the 9th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2018) and the 31st Annual Conference of the National Advisory Committee on Computing Qualifications, Wellington, NZ, July 11-13, 2018 as part of ITx 2018.

industry played the same roles for Project B. Moreover, the teacher also played the role of Scrum Master for all seven teams. Both projects had two development sprints. Each sprint had 30 days. Scrum meetings were held twice a week. Surveys were taken at the end of each sprint and at the end of the project. The results indicated that students preferred learning Scrum through practical projects instead of formal lectures. A similar method was applied to teach 52 students in 13 teams working on a web-based student records system the next year ((Mahnic, 2012). The project development was divided into three sprints after three weeks of lectures on Scrum. Each sprint took four weeks. The survey at the end of course showed better results than those in previous years. The overall research suggested that it is best to teach Scrum through projects and practical work.

Moreover, considering similarities between teaching Scrum and problem-based learning (PBL), such as teacher-based learning and self-directed learning, Scrum had been successfully implemented into PBL for product design and development for the improvement of communication and team efficiency (Ovesen, 2013). Scrum was successfully implemented in a year-long engineering design capstone project by providing students a well-defined rationale and flexibility (Sarang-Sieminski & Christianson, 2016). Furthermore, in order to encourage trust, engagement, and accountability among students, Scrum had been extended for use in teaching English, Chemistry, and Digital Media courses (Krehbiel, Salzarulo, Cosmah, Forren, et al. 2017).

Another common way to teach Scrum is using educational games. Scrum simulation with LEGO[1] was used to teach Scrum and promote collaboration and communication by developing a LEGO city (Paasivaara, Vanhanen, Heikkilä, Lassenius, et. al., 2017; Kropp & Meier, 2016). Paasivaara, et. al. (2017) introduced a four-hour LEGO game before starting a capstone 18-week development project with six sprints. Through a survey of 96 students and interviews with16 teams, they concluded that the high-performing teams paid much more attention to the Scrum process, e.g. daily Scrum and retrospective meetings, than the low-performing teams. Inspired by use of the LEGO game, Lee (2017) developed a simulation game Scrum-X using MS Excel and VBA. The simulation started from role playing as a Scrum Master, then setting up a team for the rest of Scrum process. The pilot test indicated that it has potential to support students learning Scrum.

Likewise, a Virtual Scrum simulation was developed to teach large student classes (Rodriguez, Soria & Campo, 2015). It allowed students to have chances to play different roles in Scrum, such as: Product Owner, Scrum Master, and Team member. An evaluation was done by 45 students who used it as a tool to support learning Scrum and developing a capstone project. Based on survey results, Rodriguez et al. (2015) concluded that using Virtual Scrum was helpful in teaching and developing software with Scrum.

2.2 What Issues Are in Teaching Scrum

One of the important values of teaching Scrum is to engage students in collaborative learning. However, "*students are often resistant to collaboration with their peers, for a variety of reasons, including lack of interest in the project, lack of trust in their peers…*" (Pope-Ruark, Eichel, Talbott, & Thornton, 2011, p2). Through teaching Scrum with professional communication and real collaboration in group projects, Pope-Ruark et al. (2011) concluded that Scrum had changed the way that students "*see collaboration: careful attention to process and product rather than dissection and allocation of individual tasks*" (p14). Still, there were issues when adding new project tasks and introducing new team members. Santana, Santos, Silva, and Villar, et al. (2017) reported their experience of one team of nine students developing in eleven sprints. Four students started the project in the first three sprints. It was not very productive in the first sprint because of inefficient communication. After improving communications, the productivity increased. However, after adding three new projects and five new students to the team, students did not feel they adapted to the Scrum process in an appropriate way due to a lack of training. They concluded that team integration and communication were essential to the development.

Since Scrum development focuses on team collaboration and communication, many researchers teaching Scrum focus on evaluating teamwork rather than individual student's performance. Some researchers masked and ignored individual assessment behind teamwork assessment (Kropp & Meier, 2016; Mahnic, 2015). Others considered that it is hard to evaluate individual student performance in the Scrum project (Kuhl, 2014; Gamble & Hale, 2013).

Kropp & Meier (2016) taught 45 students in five Scrum teams using a three-hour LEGO game and a ten-week educational project in five sprints. They advocated that it is important for students to learn collaborative practice and argued that "Assessing individual team members is a contradiction to the agile values" (p1010). Therefore, they awarded the same grade to all the members of a Scrum team. They found that unit testing was also identified as one of important factors in the development.

Furthermore, Igaki, Fukuyasu, Saiki, & Matsumoto, et. al. (2014) advocated that teachers should consider the inequality in the learning opportunities for each member in Scrum practice. They proposed a ticket driven development method to measure quality, task assignment, and delivery of students' projects. In order to reduce the inequalities, students took turns as Scrum Master during development. Although a set of metrics was developed for the measurement of teamwork, individual assessment was still not included in this method.

Conversely, Kuhl (2014) explicitly added evaluation of individual performance (40%) alongside team performance (60%). The individual performance consisted of peer evaluation, individual documentation, and group active evaluation. Moreover, Gamble and Hale (2013) defined four performance metrics to evaluate individual member's attributes in terms of contribution, influence, impact, and impression within a Scrum team project. Based on assessing 16 students across four teams in a software project, they found that individual performance metrics data had a strong relationship to final product outputs.

In summary, Scrum has been widely used in teaching collaborative team project development. Various Scrum simulation games were also introduced to support student learning. Besides training, there is a gap of what is needed to teach students to quickly and effectively deal with requirement changes during project development. Since each student has different learning opportunities in Scrum practice, by taking different roles and facing various difficulties in project parts, the above individual evaluation methods are more or less based on these inequalities in teamwork. Therefore, the proposed teaching design aims to provide a model-driven rapid development method for software project development in order to effectively meet changes in the project. Regarding the individual assessment, besides the metrics of evaluating individual performance in teamwork, we add an individual project, in

[1] https://www.lego4scrum.com/

which student simulates the Scrum process, taking various roles and develops the entire project.

3. COURSE DESIGN

At Universal College of Learning (UCOL) in New Zealand, the course design of D202 Software Process considers both dealing with changes and assessing individual learning in Scrum development. Scrum was introduced as the development approach in this course in 2002 (Cleland, 2003). In 2017, Based on the experience of using model-driven development (MDD) (Hu & Steele, 2017), we choose ASP.NET for our web development project. In the previous semester, students learned to generate object-oriented class code in C# from their class design model. In this course, we proposed that students continue MDD to generate a prototype of an entire web project in order meet the changes of project requirements. Therefore, ASP.NET was introduced in four weeks (three hours per week) before the Scrum project started (see Table 1). There are three assessments in this course. An individual practical test (30%) to ensure students are ready for project development. A group project (40%) is the core part of this course where students learn the Scrum software development process. In this project the practical development only accounts for 30 out of 100 marks. The last assessment is an individual project (30%) for students to simulate Scrum process under their management, in which practical development also accounts for 30 out of 100 marks.

Table 1: D201 Software Process Course Outline

Week	Topics
1	Introduction to Agile
2	Development from class code
3	Development from database
4	Unit test and web management
5	Individual Practical Test (30%)
6	Scrum in detail / Planning Poker, Group Project setup (40%)
7	Group Project - Sprint One
8	Group Project
9	Group Project - Sprint Two
10	Group Project
11	Group Project - Sprint Three
12	Group Project
13	Group Project – Sprint Four
14	Group Project - Final Sprint
15	Individual Project Work (30%)
16	Individual Project Presentation

Although we encourage students to take on various roles in Scrum, we still have concerns around the inequality of learning due to the different tasks worked on during the teamwork (Igaki et al., 2014). The aforementioned individual assessment methods (Kuhl, 2014; Gamble & Hale, 2013) were based on teamwork and contribution rather than the learning and experience of the Scrum process for the entire development. On the other hand, regarding the Scrum simulation games, although students can play various roles, the simulation does not provide real experience of software development. Therefore, besides the group project that is assessed using metrics, we add an individual project so that every student can have a chance to manage the entire project in a Scrum process. The weight of each assessment is shown in Table 1.

It is plausible to teach students the Agile Scrum framework using a conventional diagram[2]. However, it is hard for novices to master it. The notation of a conventional diagram does not rigorously match all the components and their relationships in the Scrum process. We propose that DFD's provides for better comprehension than other formative descriptions for our second-year students as they learn DFD's for analysis and design in software development during their first-year study. Based on both the Scrum Guide (Schwaber & Sutherland, 2016) and eduScrum Guide (Delhij et al., 2015), we developed a model for teaching the Scrum development process using Data Flow Diagram's (DFD) (see Figure 1). In the DFD three roles of a Scrum Team are presented as External Entities. All the Scrum events are presented as Processes, while the sequence number of each Process indicates the order of Scrum events. The user stories and Scrum artefacts are presented as Data Stores. Process 1 & 2 in the DFD represent events before sprints. The sprint cycle is shown by Process 3, 4, 5 and 6. The changes to the project are represented as New User Stories in the model.

In the group project of our course, there were 25 students in their first or second semester of second-year study. Since students were from different levels of programming courses, they were randomly divided into five groups by using a Moodle auto group feature. We wanted students to work in a new environment rather than in a familiar group. We noted that lecturers took the role of Product Owner and even Scrum Master in the aforementioned Scrum practices. We decided that our students should experience most of these roles. Initially the Product Owner was played by the lecturer who checked User Stories and then observed the sprints of all the Scrum teams. Therefore, in each group, one student took the role as Scrum Master and also played the role Product Owner; the other four students worked in pairs as two Development Teams while the Scrum Master was working in both teams.

All the groups undertook the same project, developing a web project to aid degree students in planning their study options. The project was scheduled for four sprints. Each sprint took two weeks. After two sprints, extra features were added to the current group project in order to see the benefits of using the Scrum process in the software development. At the end of each sprint, all the groups submitted documentation in the form of a Sprint Review and Sprint Retrospective. A Scrum board (see Figure 2) was used for each group in the software lab where students had both lectures and practical. The Scrum board has five columns: Story, To Do, In Process, To Verify, and Done.

Figure 2: The Scrum board in software lab

[2] https://www.neonrain.com/agile-scrum-web-development/

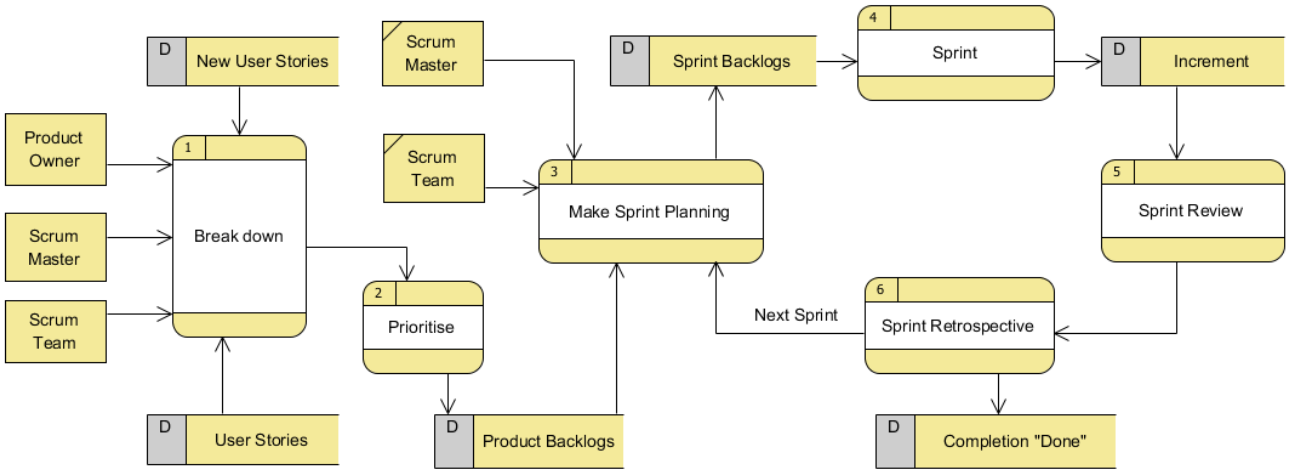

Figure 1 Scrum process in our teaching

4. DATA GATHERING

Considering the aforementioned difficulties in assessing individual performance in group projects (Kuhl, 2014; Gamble & Hale, 2013), we use a similar metric as Gamble and Hale proposed (2013) (see Table 2). Each criterion is assessed by the scale of Always, Mostly, Usually, Hardly Ever, and Never scored from 5 to 1. Each student evaluates other members and himself/herself in all the criteria and provides comments. The grand average from average scores of all the criteria is used as an index for the portion of the group grade allocated to individuals (see Table 3). The sample data in Table 2 shows a student's evaluation details and final result of Grand Average scored 4. Based on the Table 3, this student receives 90% of teamwork mark. For example, if the group mark is 90, his/her individual mark will be 81.

Table 2: Sample Metrics of Individual Performance in Scrum

By	Communi-cation	Fulfilled Role	Contribution	Reliable
A	4	4	4	5
B	5	4	5	4
C	3	3	4	3
D	3	3	3	3
E	5	5	5	5
Average:	4	3.8	4.2	4
Grand Average:		4		

Table 3: Individual Portions from Group Mark

Grand Average	Percent from Group Mark	Grand Average	Percent from Group Mark
≥ 4.5	100%	≥ 2.0	40%
≥ 4.0	90%	≥ 1.5	30%
≥ 3.5	75%	> 1.0	15%
≥ 3.0	60%	≤ 1.0	0%
≥ 2.5	50%		

In the first practical test, students were required to choose their own scenario to develop a web site using ASP.NET. After two sprints in the group project, students were also given an individual project to add additional functions to his/her prototype implemented in the practical test in order to simulate the Scrum development process. Finally, each student had a presentation for the simulated Scrum software development.

5. RESULTS AND DISSCUSIONS

After four weeks of training in ASP.NET, most students (22, 88%) had successfully developed a prototype of a web site in the practical test. However, ten students (40%) failed to complete a unit test. We realised that nearly half of students had not taken the advanced programming course where they would have learned this technique. We felt that these students would learn the creation of unit tests over the next two assessments in Scrum. However, only one of the three students who failed the test passed the course at the end of semester.

In the Scrum group project, we worked closely with every group through participation and observation. At the beginning, some students were not able to transfer project requirements into user stories. Others were quickly putting their cards on the Scrum board. Using the Scrum board for transparent development, students learned from other groups and set up priorities for Product backlogs. Before sprint starts, the lecturer checked User Stories for all groups. Although each group can change their Scrum Master, no groups changed their selected Scrum Master. We observed that students were supporting their teamwork using various tools for communication such as Google Docs, Trello, and GitHub. In each sprint, every group updated their Scrum board and burndown chart. Based on the above individual assessment metrics, two students failed in the group project. One who had already failed in the first test failed the course eventually. Another passed the course based on his good work in the individual project.

In the final individual assessment, five students gave up. Among them two had already passed the course (50%) based on first test (30%) and group project (40%). The other two who had already failed in the first test failed the course. Although the last one did not fail in the previous two assessments, failed the course overall. In other words, twenty-two out of twenty-five students successfully passed this course. One student failed the course because he failed all three assessments. Another student failed both the individual assessments and the course overall although he passed in group assessment. The last one failed the course because he failed the individual project although he passed the other two assessments.

After the course, a survey was collected from twelve students (see Table 4). We used a Likert scale to evaluate each survey question as Lethbridge et al. (2011) suggest, ranking from 1 to 5 in accordance to "Strongly disagree", "Disagree", "Undecided/neutral", "Agree", and "Strongly agree". In the Table 4, the results of "Agree" and "Strong Agree" were merged to "Agreed" as well as those of "Disagree" and "Strong

Disagreed" to "Disagreed". The results show that there are more students who agreed than those who disagreed in our use of model-driven development to generate code for changes and managing the changes from Scrum. Although students agreed that they learned from the group project for their individual project, they did not feel engaged or aided in their learning through the group project.

Table 4: Survey Results

Question	Agreed	Neutral	Disagreed
1. I believe model-driven development of generating code fits well to dynamic changes.	50%	33%	17%
2. Through Scrum, I understand software process of facing dynamic changes.	50%	17%	33%
3. I applied what I learned from group project to my individual project.	50%	8%	42%
4. The group project using Scrum process engages my learning	25%	33%	42%
5. I believe that group project helps my learning in this course.	33%	8%	58%

In the comments, it reveals that some students felt that they learnt most from the individual assignment although many students agreed that they learned from both individual and group assignments. Many students prefer to learn the technical skills, i.e. code development rather than management skills in the software development process. Some of them expected to learn coding experience as in Advanced Programming course before studying this one. That is to say, students expected to learn technical skills rather than project management method from this course. In this course, they did learn the Scrum method from the group project and used it to manage the process of their individual project. However, they also learned more technical development skills from their individual projects than from the group project.

The limitations of our study are that we only have a small class size. The survey data size is even smaller. Our metrics data was only collected after teamwork, which may not fully reflect individual performance in the group project. In exceptional cases, the scale setting may not be appropriate for converting the assessment of an individual's grand average into a percentage of group marks. More testing is needed for the data conversion. The bias of student comments to other members may affect the validity of the metrics data. We noticed that many students gave full marks to themselves. Our students were from different levels of programming courses, which could constrain their motivation, contribution, and collaboration by some students.

Recently, similar research had been done by using Scrum for Android project at university in New Zealand (Zolduoarrati, Mohammadi, Lotter, Alessi, et. al., 2017). Zolduoarrati, et al. (2017) reported their study of using Scrum for student software project management to develop an Android app of managing receipts by six teams in two three-week sprints. The students also as authors of their research had learnt many lessons in every event of Scrum, specifically in project requirements, priority, schedule, collaboration, and risk monitoring by burndown charts, However, their experiment of changes were limited to short-term scheduling rather than functional changes in the project. It may not be adaptable to rapid development in their Android app development. Also, they only had one assessment based on teamwork, which may lead to inequality of learning as aforementioned in the literature study.

6. CONCLUSIONS

From this study, we conclude that although teaching Scrum focuses on collaboration, both individual and group projects are necessary in this course. Student perceptions of this course were to learn more technical development skills rather than project management methods. Therefore, we need to ensure that students understand Scrum is a software project management method rather than a software development technical skill. Students mainly learnt the management method in the group project and study more details of the method and most technical skills through the individual project.

Another important value of teaching Scrum is to deal with changes during software development. Student's handling of changes was not only satisfied at the project management level, e.g. scheduled in a sprint, but also at a development technical level, such as model-driven development and rapid development.

We also see the limitation of our methods from student feedback and failure in the group project using Scrum. We conclude that we need to enhance student engagement during the Scrum process. For example, our metrics for assessing individual performance in teamwork are only utilised when the group project was completed. This loosens student's response and obligation to contribute to each sprint. We would then adjust and utilise the assessment metrics more often than we had, e.g. once per sprint. We propose that based on the feedback during the Scrum process we could provide effective support to meet individual student learning requirements.

The contribution of this study is that we use a widely recognised and rigorous technical notation to describe the Scrum teaching process. We introduced model-driven development for students to effectively deal with requirement changes in the project development. We emphasise the need to have both individual and group assessments in order to let students learn all the aspects of the Scrum development process. We also developed evaluation metrics for assessing individual performance in teamwork.

In future research, we will update our software tool for model-driven development. We will also develop more evaluation metrics to be used at the end of each sprint.

7. REFERENCES

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile software development methodologies: survey of surveys. *Journal of Computer and Communications*, 5(05), 74.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development. Retrieved from http://www.agilemanifesto.org/

Cleland, S. (2003). Agility in the classroom: Using Agile development methods to foster teamwork and adaptability amongst undergraduate programmers. In Mann, S. & Williamson (Eds) Proceedings of the *16th Annual NACCQ Conference*, Palmerston North, New Zealand, July, 2003

Delhij, A., van Solingen, R., & Wijnands, W. (2015). *The eduScrum guide*. Retrieved from: http://eduscrum.nl/en/file/CKFiles/ The_eduScrum_Guide_EN_1.2(1).pdf

Gamble, R. F., & Hale, M. L. (2013, October). Assessing individual performance in Agile undergraduate software engineering teams. In Proceedings of *Frontiers in Education Conference, 2013 IEEE* (pp. 1678-1684). IEEE.

Hu, M., & Steele, A. (2017). Reducing the Gap between OOP Design and Development. In Erturk, E. (Ed) Proceedings of the 8th Annual Conference of Computing and Information Technology Research and Education New Zealand (CITRENZ), 70-75. Retrieved from http://www.citrenz.ac.nz/conferences/2017/pdf/2017CITRENZ_1_Hu_Development.pdf

Igaki, H., Fukuyasu, N., Saiki, S., Matsumoto, S., & Kusumoto, S. (2014, May). Quantitative assessment with using ticket driven development for teaching scrum framework. In Jalote, P., Briand, L. & van der Hoek A. (Eds) Companion Proceedings of the *36th International Conference on Software Engineering* (pp. 372-381). ACM.

Krehbiel, T. C., Salzarulo, P. A., Cosmah, M. L., Forren, J., Gannod, G., Havelka, D., ... & Merhout, J. (2017). Agile Manifesto for Teaching and Learning. *Journal of Effective Teaching*, *17*(2).

Kropp, M., & Meier, A. (2016). Collaboration and human factors in software development: Teaching agile methodologies based on industrial insight. In Proceedings of the *Global Engineering Education Conference (EDUCON)*, IEEE (pp. 1003-1011).

Kuhl, J. G. (2014). Incorporation of Agile Development Methodology into a Capstone Software Engineering Project Course. In Proceedings of the *2014 ASEE North Midwest Section Conference* (pp. 1-8).

Lee, W. L. (2016). SCRUM-X: An interactive and experiential learning platform for teaching scrum. Retrieved from: http://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=4379&context=sis_research

Lethbridge, T. C., Mussbacher, G., Forward, A., & Badreddin, O. (2011). Teaching UML using umple: Applying model oriented programming in the classroom. In Proceedings of *Software Engineering Education and Training (CSEE&T) 2011, 24th IEEE-CS Conference* (pp. 421-428). IEEE.

Mahnic, V. (2015). Scrum in software engineering courses: an outline of the literature. *Global Journal of Engineering Education*, *17*(2), 77-83.

Mahnic, V. (2012). A capstone course on agile software development using Scrum. *IEEE Transactions on Education*, *55*(1), 99-106.

Mahnic, V. (2010). Teaching Scrum through team-project work: Students' perceptions and teacher's observations. *International Journal of Engineering Education*, *26*(1), 96.

Ovesen, N. (2013). Facilitating problem-based learning in teams with Scrum. In Proceedings of the *15th International Conference on Engineering and Product Design Education*, Dublin, Ireland.

Paasivaara, M., Vanhanen, J., Heikkilä, V. T., Lassenius, C., Itkonen, J., & Laukkanen, E. (2017, May). Do high and low performing student teams use Scrum differently in capstone projects?. In Werner, C. & Whittle, J. (Eds) Proceedings of the *39th International Conference on Software Engineering: Software Engineering and Education Track* (pp. 146-149). IEEE Press.

Pope-Ruark, R., Eichel, M., Talbott, S., & Thornton, K. (2011). Let's Scrum: How Scrum methodology encourages students to view themselves as collaborators. *Teaching and Learning Together in Higher Education*, *1*(3), 5.

Rodriguez, G., Soria, Á., & Campo, M. (2015). Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to Scrum. *Computer Applications in Engineering Education*, *23*(1), 147-156.

Santana, L. F., Santos, L. F. C. D., Silva, T. S. C., Villar, V. B., Rocha, F. G., & Gonçalves, V. (2017). Scrum as a platform to manage students in projects of technological development and scientific initiation: a study case realized at UNIT/SE. *Journal of Information Systems Engineering & Management*, *2*(2).

Sarang-Sieminski, A., & Christianson, R. (2016). Agile/Scrum for Capstone Project Management. *Capstone Design Conference*, Columbus, Ohio.

Schwaber, K. Sutherland, J (2016). *The Definitive Guide to Scrum: The Rules of the Game*. July 2016. Retrieved from https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf

VersionOne, *The 11th Annual State of Agile Report* (2017). Retrived from http://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf

Zolduoarrati, E., Mohammadi, R., Lotter, A., Alessi, N., Michael, K., & Licorish, S. A. (2017). Reflections on the use of Agile practices and associated tools in university settings for an Android project. In Erturk, E. (Ed) Proceedings of the *8th Annual Conference of Computing and Information Technology Research and Education New Zealand (CITRENZ)*, Retrieved from http://www.citrenz.ac.nz/conferences/2017/pdf/2017CITRENZ_1_Licorish_Agile.pdf