Informatics in Education, 2022, Vol. 21, No. 4, 605–634

DOI: 10.15388/infedu.2022.25

Adapting Scrum for Software Capstone Courses

Hung-Fu CHANG, Mohammad SHOKROLAH SHIRAZI
R.B. Annis School of Engineering, University of Indianapolis
Indianapolis, United States
e-mail: hchang@uindy.edu, shirazim@uindy.edu

Received: April 2022

Abstract. Scrum is a widely-used framework in industry, so many schools apply it to their software engineering courses, particularly capstone courses. Due to the differences between students and industrial professionals, changing Scrum is necessary to fit capstone projects. In this paper, we suggest a decision-making process to assist instructors in developing a strategy to adapt Scrum for their course. This framework considers critical differences, such as student's workloads and course schedules, and keeps the Agile principles and Scrum events. To evaluate the adapted Scrum, we investigated student's learning experiences, satisfaction, and performance by quantitatively analyzing user story points and source codes and qualitatively studying instructor's evaluations, student's feedback, and Sprint Retrospective notes. Our two case studies about adapted Scrum showed that having daily stand-up meetings in every class was not helpful, student's satisfaction positively correlated to the difficulty of the task they tackled, and the project provided good learning experiences.

Keywords: Scrum, software engineering, Agile software development, software engineering education.

1. Introduction

University software engineering courses mainly focus on building up student's theoretical foundation and technical knowledge. Students learn theories and techniques through assignments within a limited problem scope. Although these assignments could be created by dividing a project into smaller pieces where students finally can see the entire project outcome, it is difficult for students to acquire the real-world project experiences. These courses emphasize teaching hard skills rather than developing student's soft skills. When students enter the industry, soft skills in project development and management, which the industry needs, are lacking (Begel *et al.*, 2008).

To respond to this, universities embrace capstone projects in software engineering education (Mahnic *et al.*, 2008). These projects usually involve real-world domain experts as clients who bring practical needs with actual project timelines to students. Such courses address an opportunity to offer students industry-like software development ex-

periences. Students can deepen their theoretical understanding of software engineering practices through the process of solving the problems given by the industrial experts and develop their teamwork, communication, and management skills during the project execution (Coppit *et al.*, 2008; Paasivaara *et al.*, 2013; Kamthan *et al.* 2016).

With the widespread usage of the Agile process in many organizations, Scrum, which is presented by Ken Schwaber (1997), has become a trendy framework in Agile software development approaches. When Scrum is incorporated in the software capstone project in the classroom, how to employ Scrum to ensure the quality of assessments on student's theoretical understandings of the software development practices and their applications in solving the real-world problem and maintaining student's learning experiences becomes critical. Hence, modifying Scrum to fit a classroom is usually required because literature emphasizes the importance of understanding the differences between student projects in the classroom setting compared to industry, for example, student versus professional expertise, academic calendar, and student's limited project working time versus a 40-hour work per week (Hoskey *et al.*, 2016; Mahnic, 2012). Research indicated the need for an adaptive approach tailored to the limited time and resources available in an academic setting (Wagh, 2012; Baird *et al.*, 2012). At the same time, the change should not compromise its core belief and principles when adaption happens.

Since Agile development processes encourage finding solutions according to the given circumstances instead of blindly adhering to a prescribed process (Beck *et al.*, 2001), we can develop a method to adapt Scrum to a software capstone course. One obvious challenge is student's learning assessment. In comparing the capstone course with regular teaching, assignment or exam problems are designed and controlled in a way that can facilitate student's understanding of assessments. It is not easy to monitor the quality of activities and student's learning experiences in a software project. Among various methods, such as only assessing student's learning at the end of the project or treating students as laboratory members to examine progress and solutions in a laboratory setting, the ultimate strategy requires teaching staff to accompany students while they are working throughout the entire project to ensure that what students learn is correctly applied (Matthies *et al.* 2016). Rather than applying these approaches, we mimic how companies evaluate project development when Scrum is used. We assess student's learning at Scrum events, but we alter the frequency of the Scrum events to fit our course. More importantly, at these points, we observe the performance of the students, provide feedback after each event, and guide students if they do not properly apply what they learn. At the same time, we want to reduce interference when we observe these Scrum events. In other words, we aim to maintain this freedom while creating comparable insights into the process.

To understand student's learning experiences on our adapted Scrum, we conduct quantitative analysis on source codes and user story data along with a qualitative study on evaluation notes and surveys. The quantitative and qualitative analyses of the two selected teams can cross-validate our findings. Our main contribution is that these study results and lessons learned will benefit those who carry out Scrum in capstone projects in software engineering courses.

In this paper, we discuss related work in section 2. In section 3, two student teams, course settings, and capstone projects are presented. Then, we discuss how we change Scrum to use it in the team's capstone projects. Section 4 introduces our Scrum adaption decision process and discusses the research method, data collection, and measurements. Then, we show our case study results and discuss our findings in section 5. Finally, we conclude our work in section 6.

2. Literature Review

Our research focuses on applying Scrum in the classroom and how to assess its implementation in courses. We would like to discuss related works in the following three categories and conclude the need for investigating Scrum adaption in the engineering courses.

2.1. *Evaluations and Feedback on Agile and Scrum Implementation*

Many previous studies investigated using Agile or Scrum in the software engineering context. Umphress *et al.* (2002) studied 49 capstone projects and concluded that Agile approaches assist student team in meeting customer expectations in the final software product. More recent work described student feedback from Agile or Scrum implementations in various courses, including capstone projects and web programming courses (Mahnic, 2012; Baird *et al.*, 2012; Weber *et al.*, 2016). Mahnic (2012) reported overwhelmingly positive perceptions from students about introducing Scrum in an undergraduate capstone course in software engineering and similar results were also found in other research like (Scharf *et al.*, 2013; Magana *et al.*, 2018; Christov *et al.*, 2019; Kharitonova *et al.*, 2019; Ju *et al.*, 2020). These positive experiences showed that using Scrum in the classroom offered students to develop better skills (e.g., social and technical skills) in preparation for the industry although intensive efforts are required from students.

Many studies analyzed evaluations, student interviews, or teachers' reflections at the end of a course for their main purpose – improving the course design (Christov *et al.*, 2019; Kharitonova *et al.*, 2019; Ju *et al.*, 2020). However, these studies also suggested that deepening our understanding beyond effective implementation and positive perception were needed. Research should move towards understanding what happens inside Scrum teams, observing the dynamics between team members, and investigating what students learn. Studies can be done from student perspectives, specifically on how teams manage software projects and how they reflect on the development process.

2.2. *Scrum Adaptions in Classrooms*

Modifications on Scrum to fit the classroom are usually required because literature emphasizes the importance of understanding the differences between student projects in

the classroom setting compared to industry, for example, student versus professional expertise, academic calendar, and student's limited project working time versus a 40-hour work week (Mahnic, 2012; Hoskey *et al.*, 2016). When adaption happens, the change should not compromise its core belief and principles. Baird *et al.* (2012) integrated plan-driven and Scrum approaches by excluding Sprint Reviews and Sprint Retrospectives. Wagh (2012) mentioned the need for an adaptive approach that can be tailored to the limited time and resources available in an academic setting. Baham (2019) discussed using Scrum to fit software development life cycle principles to provide students with an example of how all the roles and ceremonies work. Masood *et al.* (2018) highlighted the effectiveness of specific adaptations to agile practices in a university context and recommended conducting face-to-face events, supporting teams with experienced tutors and training, and using online tools for team communication.

Some recent studies discussed how Scrum was implemented in various courses or emphasized techniques for teamwork under the Agile theme. Jiménez *et al.* (2016) discussed their Scrum implementation experiences in software engineering, computer game, and human-computer interface design courses. Kropp *et al.* (2016) focused on collaboration skills in the Agile development process. May *et al.* (2016) used a ball game to let students experience the effect of a self-organizing team – an expectation to a Scrum team. Jurado-Navas *et al.* (2017) discussed that students had positive Scrum team experiences in the English classroom. Hence, these related Scrum application or modification works followed the Scrum philosophy and mainly focused their analysis on a particular aspect. However, they do not strictly follow Scrum event execution principles as what the industry does.

2.3. *Course Project Assessment*

Another primary reason for adapting Scrum is how and when instructors conduct their project performance assessments. One work studied a laboratory-like setting. It allowed students to almost manage the project fully themselves and guidance was provided when students asked. This way potentially reduced the learning experience for students (Devedzic *et al.*, 2011). Matthies *et al.* (2016) used surveys and tutors along with Scrum events to understand how students implemented Scrum in this course and which elements require further teaching in later iterations. A common practice from Kropp and Meier (2013) is to perform post-hoc oral or written exams to judge the project's success and student's learning outcomes.

Unlike industrial assessment practice that uses Scrum burndown chart to gain insight of the project performance, measuring student's performance in the classroom also needs to be changed. Mahnic (2010) applied a technique called the Earned Value method (EVM) and its associated performance indexes to project management. In the same study, he also used a series of surveys to determine if his course learning targets were met. Igaki *et al.* (2014) used an approach named Ticket Driven Development (TiDD) that gained substantial insights of Scrum teams. However, the ticket creation of the TiDD added overhead and reduced the freedom in adapting Scrum to the team's needs.

2.4. *Summary*

Even though past research discussed Scrum adaptation and approaches for project assessments, many studies encouraged deepening our understanding beyond effective implementation. Therefore, research should move towards investigating what happens inside Scrum teams, observing the dynamics between team members, and understanding what students learn. Studies can be done from student perspectives, especially on how teams manage software projects and reflect on the development process. In addition, previous studies have applied Scrum in the course, but most executed Scrum events without strictly following the suggested frequency. Moreover, no Scrum implementations simultaneously considered student's working hours and class schedules – two critical factors about how to handle course projects as professional ones. As a result, we present a new approach to adapt Scrum for classroom usage, study its influences on the undergraduate course, and examine the detailed team dynamics to know how it impacts student's competencies.

3. Research Context

3.1. *Scrum Framework Explained*

The Scrum workflow contains a sequence of iterations named Sprints. Its duration is decided by the team and should be from one to four weeks. The team in Scrum includes the following roles: Product Owner, developers, and a Scrum Master to manage the project. When the product starts, the team specifies the functionalities that system should have and puts them into a list, called product backlog item list. In other words, a product backlog item is a system functionality that can contain vague or very specific description. Each Sprint is a development cycle, and the to-be-developed tasks will move from the product backlog item list to the Sprint backlog item list. So, at each Sprint, the team performs requirement analysis, design, task planning, and implementation and testing according to the Sprint backlog item and is required to meet in four timeboxed events, which are daily stand-ups, Sprint plannings, Sprint reviews, and Sprint retrospectives. We describe Scrum roles and events in details in the following.

Accountabilities in a Scrum Team

- Scrum Master (SM):
  This role is mandatory and not concurrent with other roles to ensure the job is carried with full attention. The primary responsibility of SM is to facilitate events to remove the possible blocks of the progress.
- Product Owner (PO):
  PO acts like an advocator for customer's voices and is responsible for creating and maintaining Product Backlog priorities.
- Developers:

Responsible for development tasks in each Sprint.

Scrum Events

- Sprint Planning:
  The entire Scrum team must attend the Sprint Planning. During the meeting, the Scrum team plan for which product backlog items will be fulfilled in the following Spring. The outcome of the planning meeting is a Sprint backlog item list that serves as an agreement among team members.
- Daily Stand-up Meeting:
  This is a time-boxed event that will be held every day. The Scrum team uses this meeting to synchronize individual status and discusses what has been done and what needs to be done. Due to the time limit, technical discussions are suggested to be avoided in this meeting.
- Sprint Review:
  This meeting is conducted at the end of the Spring. The attendees of this meeting are the Scrum team members and associated stakeholders. All the stakeholders in the meeting provide feedback for completed Sprint backlog items.
- Sprint Retrospective:
  The Scrum team assesses the performance of the Sprint. It is a mandatory event for the entire team, and the Sprint is closed with a Sprint Retrospective.

A popular solution to describe the product backlog item is the user story. User stories are designed based on software requirements. Every user story contains one or more tasks that need to be completed by the Scrum team in a Sprint. In practice, when a user story cannot provide sufficient details, this type of user story is called epic. In real-world Scrum practices, depending on the project, the team can decide whether a separate meeting, called Scrum grooming, is needed to conduct each Sprint to let the team fine-tune the product backlog. Some Scrum teams can also use the partial Scrum planning meeting time to refine backlog. Grooming is a critical activity because it makes engineers dig into the analysis and design and enables a smooth Sprint planning meeting. In our study, we combined the grooming activities with Sprint planning because it is challenging to distribute too many separated meetings in the classroom.

3.2. *Course*

SWEN400 Software Project Management at the University of Indianapolis is a one-semester course that teaches general software project management techniques and practices Scrum in a capstone project. The course was taught during a 15-week semester and required team-based work, three individual assignments, one mid-semester exam and a final exam. These exams tested student's understanding of software project management, like various development processes, measurement, and planning. The teaching materials covered traditional plan-driven and Agile software development approaches and techniques.

Before the semester started, software projects were collected from industrial partners. After course projects were assigned to student teams, several client interviews were held for teams to understand the project scope and customer needs. During this period, high-level software architecture, requirements, and work breakdown were also given as assignments and then discussed during the class.

Students received Agile and Scrum training in the first three weeks of the course to ensure that students had sufficient knowledge to execute their projects before they started development. Students learned how to write user stories through several assignments and in-class discussions. Students practiced user story point estimation by playing Scrum (Agile) Poker to reach a team consensus; that is, all team members must agree upon the story points for a user story during the Sprint planning meeting. The user story points followed the Fibonacci sequence numbers. Team members understood that they estimated the complexity of the user story rather than the time spent on the task completion.

After the project started, teaching and team project discussions went parallelly. In other words, some lecture time is reserved for student's project. Student's performance during the Scrum events will be evaluated by an instructor in the class. After each Scrum event, the meeting notes must be submitted for evaluation. In the middle of the semester, there were a midterm project review and a course survey. At the end of the semester, the final project review and survey were executed.

3.3. *Teams and Project Background*

Among fourteen registered senior software engineering students, which formed four student teams, two student teams joined our case study. We call these two teams, A and B, and students in both teams are considered to have sufficient training in some software development skills. Prior to taking this course, students were required to complete the prerequisite courses like programming language, database, operating systems, introduction to software engineering, software testing, and software architecture courses. Part of the students gained team development experiences from their internship or other engineering project courses. Since the user interface design and web development courses are selective, only a few students took them (see Table 1).

Team A and B worked on two different types of software systems. Team A built a brand new online real-time chatting system. Team B took over an incomplete online classroom system, so customers expected new functionalities and improvements from the team's development. Therefore, the common skills for both systems are web technology, database, software architecture, and user interface (see Table 2). Both teams need different project specific techniques in addition to the understanding of the domain business (see Table 3). For team A, students need to understand real-time systems. Team B students must know Google Cloud Services. Because both teams had poor understandings about project specific technologies, around the beginning of the project, students were asked to research those technologies and discuss with instructors about their understandings.

Table 1
Team formation and project

	Team A	Team B
Software System	Online real-time chatting system	Online classroom
System Type	Brand new system	Existing system
Team Members	Four senior software engineering students	Four senior software engineering students
Project Experiences	• Three team members have leadership experiences in other engineering courses. • Two team members have Agile development experiences.	• Two team members have leadership experiences in other engineering courses. • Only one has Agile development experience.

Table 2
Evaluation of the team A and B common skill set

Skills	Team	Evaluations	Comments
Web Technology	A	Good	Three team members know to write a web application in the Python Flask framework.
	B	Excellent	All team members know how to write JavaScript programs and can use ReactJS framework. They also know about Python Flask web framework.
Database	A	Relational: Average NoSQL: Poor	All team members know about relational databases but have limited understanding about NoSQL databases.
	B	Relational: Good NoSQL: Average	All team members understand the relational database well and have limited programming experience in NoSQL databases.
Software Architecture	A	Poor	All team members have limited understanding about software modeling and design.
	B	Average	Two members know how to design a system and analyze the architecture.
User Interface (UI) Design	A	Average	Three team members have experiences in wireframe and UI design.
	B	Average	All team members have the experiences in wireframe and UI design.
Domain Knowledge	A	Average	All team members know the workflow of the website message and interaction between two users on the chatting system.
	B	Good	One team member worked on a similar online lesson system before. The rest of the team members understand the features an online learning system has.

Table 3
Evaluation of the team A and B skill sets about specific technologies

Skills	Team	Evaluations	Comments
Real-time System	A	Very poor	No team members equip the knowledge of a real-time system.
Google Cloud Technology (Firebase)	B	Poor	All team members have limited understanding about Google Cloud Services.

Fig. 1. User Interfaces of the ClickUp.

3.4. *Tools*

ClickUp (see Fig. 1) was used to manage the project. The tool provides a columnized Scrum board, which allows users to define various states of a user story (e.g., open, in-progress, and close). Its user interfaces and operations are similar to most industrial tools.

Other software development tools are also introduced to students for managing the software project. Both teams saved their codes in Gitlab repositories and used Miro (`https://miro.com/`) to create their user interface wireframe. To help the system deployment, Amazon Web Service (AWS) and Google Cloud Platform are clearly explained to team A and B, respectively.

One primary purpose of using these tools, particularly ClickUp, to student learning experiences is to let them understand how the industry handles Scrum in product development. Also, saving notes and source codes in Gitlab lets us easily collect data for obtaining insightful information.

4. Method

Our proposed adapted Scrum decision process can assist course instructors to change their Scrum event schedules according to their settings. To investigate the proposed adapted Scrum decision process, two case study teams followed our adapted Scrum and then we analyzed team dynamics and performance. The following sections discuss our framework, research questions, data collection, and analysis procedure.

4.1. *Scrum Adaptation Decision Process*

Our decision process considers the differences between students and industrial workers. It listed the following aspects to help instructors specify how Scrum should be adapted for classroom usage. At the same time, we also believe that our adapted Scrum decision process not only follows the principles stated in the Agile Manifesto (Beck *et al.*, 2001) but also closely aligns with what Scrum framework requests. In Fig. 2, we show how Scrum event execution recommendations are associated with considerations of practitioners' differences in the decision process.

Decide Number of Teams

We suggest that evaluators monitor all the Scrum meetings and one evaluator (i.e., instructor or teaching assistant) helps at most two teams. This number is also influenced by how many projects the course has.

Team Size

Team size is related to the team number and the Scrum best practice. In the industrial best practices, the range of team size is from six to eight people. We think having a large team may discourage team communication and participation and further impacts student's learning. We also suggest that the minimum number of team members is three

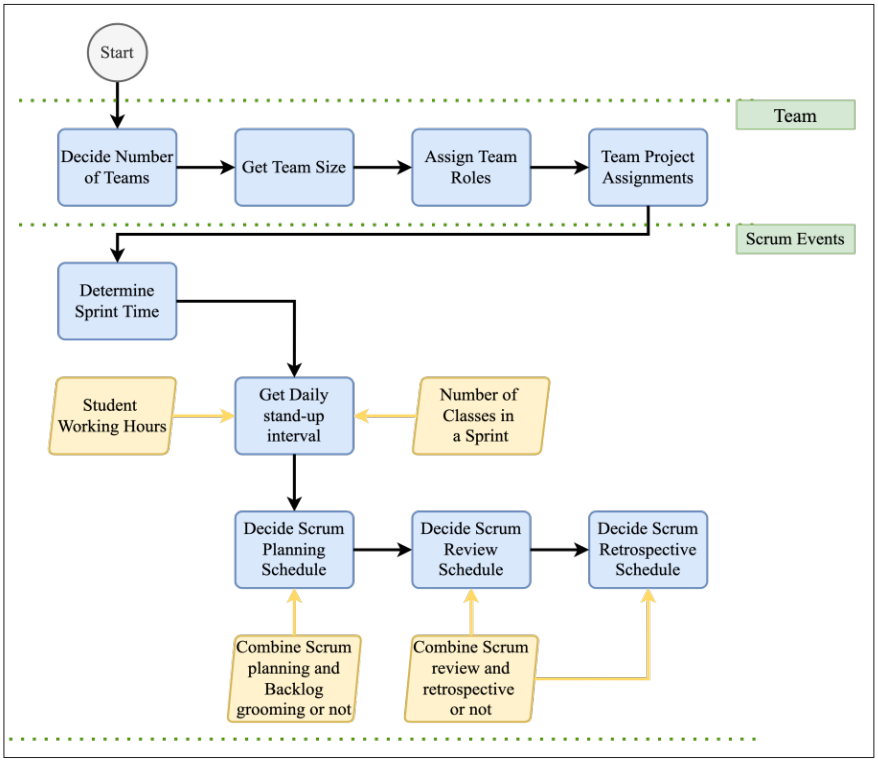

Fig. 2. Scrum Adaptation Decision Process.

because one team member has his/her own role and Scrum roles include product owner, developers, and Scrum master.

Assigning Team Roles

We can fix the role for individual members throughout the entire course or let members rotate some roles. This decision is made according to whether we want to enhance certain role's expertise and experiences for all students. For example, if we would like to encourage all the students to exercise Scrum master, the instructor can consider rotating Scrum master role-play among students.

Team Project Assignment

The project assignment to a team depends on team member's learning goals and skills. The team can express their strong desire to learn some technologies when they do not have any background. This risk must be evaluated and communicated between the instructor and students.

Determine Sprint Time

One or two weeks for a Sprint is suggested in practice. More than two weeks should not be considered because the development team wants quick customer feedback and then improve the progress or adjust the implementation when the situation changes. However, one week for the class setting is too short because students do not have enough time to make sufficient progress on the project. To note that, when we consider classes, we include both lectures and labs (exercises). We also need to consult with our customers about their schedules because Scrum review requires customer participation.

Get Daily Stand-up Interval

This interval considers two factors: student working hours and the number of classes in a Sprint. We do not encourage students to hold stand-up meetings without the teaching staff's participation. The teaching staff can provide feedback to students after the stand-up meeting so that students can learn how to handle the stand-up meeting efficiently.

- **Student Working Hours**
  Industrial professionals work eight hours per day. In other words, the interval between two daily stand-up events for eight working hours maps to 24 hours. If students can work on the project for two hours per day. Daily stand-up should be changed to at least every four days (i.e., 8/2 = 4).
- **Number of Classes in a Sprint**
  Holding daily stand-up meetings in class time is a straightforward way because we want teaching staff to accompany student teams. If the daily stand-up meeting interval is four days and the weekly schedule of the class time is Monday, Wednesday, and Friday, the daily stand-up time should be every Friday or two consecutive classes closest to the computed interval. However, one disadvantage of picking class time for daily stand-up meeting is that the regular lecture time decreases. If the teaching staff can reserve another time for the stand-up meeting, this factor is not considered.

Determine Scrum Planning Schedule

With the same consideration regarding the teaching staff's participation, we suggested the Scrum planning meeting should be held in the class. Because product backlog item

grooming meeting is very critical for task estimations, it must be before the planning. If the time permits, grooming and planning can be combined. However, if the teaching staff can use other time for planning, using class time is not necessary.

Determine Scrum Review and Retrospective Schedules

The review schedule needs to take the client's schedule into account. It occurs at the end of the Sprint and the team expects the client's feedback in the meeting. We insist on teaching staff's participation so using class time is suggested. The same suggestion also applies to retrospectives. One can combine both review and retrospective at the end of the Sprint but keep in mind that the client cannot join the Scrum retrospective event. Again, if other time can be reserved for review or retrospective events, using class time is not required.

Table 4 justifies why our adapted Scrum decision process preserves the spirit of Agile and the key elements of Scrum, except for one major difference regarding stand-up meeting frequency. We argue that violating daily stand-up principle is inevitable because students cannot meet daily to synchronize their individual status. To understand the impact of this change, we will investigate it in our case study.

4.2. *Role Play in The Team*

There are two reasons for us to determine the role play in the team. First, undergraduate students did not have sufficient experience in requirement specification and software architecture to handle complex design problems. To help students learn about them and

Table 4
Adapted Scrum justifications

Sprints	Duration	Adapted Scrum 1 to 2 weeks	Scrum 1 to 4 weeks
Events	Daily stand-up	**Frequency –** not daily. Adjusted frequency based on working hours (Note: see potential issue)	**Frequency –** Daily
	Retrospective	Same as Scrum	**Duration –** Timeboxed **Handling –** Three questions: (1) What did you do well in the Sprint? (2) What didn't you do well in the Sprint? (3) What will you do to improve it in the next Sprint?
	Review	Same as Scrum	Demo to stakeholders (including customers). Review the finished work.
	Planning	Same as Scrum	Assign user stories and tasks.
Roles	Responsibilities	Same as Scrum	Scrum master, product owner, and developers have their responsibilities described in the Scrum guide (Schwaber *et al.*, 2020)

Note: potential violation on the Agile principle – **Business people and developers must work together daily throughout the project.**

make the project handle smoothly, the instructor provided the least technical support to students for playing the product owner role but did not directly guide how to write user stories, work breakdowns, and task assignments. In other words, the instructor played the part of the product owner role. Second, one learning objective is to let students be familiar with the responsibilities of the Scrum master. In both Team A and B, each student plays the Scrum master role each Sprint.

4.3. *Research Questions*

Rather than measuring our adapted Scrum classroom application by examining grades, customer satisfaction, or assignment achievement, we particularly want to understand what happened inside the team. By studying team dynamics, we can understand whether our adapted Scrum can bring similar effects or benefits as original Scrum. If our adapted Scrum is against the original Scrum principles or cannot be as effective as the original one, such as increasing communication to promote collaboration or offering flexibility to address changes, team collaboration and performance will be impacted. Moreover, we want to understand if the adapted Scrum promotes student learning and team experiences. Hence, we ask questions to examine member satisfaction and team improvement. Therefore, we decide on the following research questions by extending the proposed criteria from past research (Song *et al.*, 2015).

- **Scrum Learning:**

  RQ1: How well do students learn about Scrum from the adapted Scrum?
  RQ2: How do the adapted Sprint events impact the student team?

- **Team Performance:**

  RQ3: What makes the student team increase or decrease product development performance?

- **Member Satisfaction:**

  RQ4: What makes team members feel satisfied?

- **Team Improvement:**

  RQ5: What parts of the team experiences enhance each member's capability to work and learn together?

RQ1 investigates if our adapted Scrum can enable the understanding of Scrum. It is also used for understanding the impact of our adaptation. Since we know the violation of the Agile principle occurs at the stand-up meeting interval, in our case study (see Table 5), we asked team A to perform a stand-up meeting every class and team B to follow our adjusted frequency calculated based on their working hours. Therefore, in RQ2, we can examine the team activities to know how much impact the stand-up meeting frequency brings to the project.

RQ4 analyzes what contributes to their satisfaction. Because we think improvements in individuals or teams and project achievement should also be a factor of satisfaction, RQ4 looks at what RQ5 analyzes from another angle.

Table 5
Compare daily stand-up events and roles

		Adapted Scrum in the class	Scrum
Daily Scrum	Frequency	**Team A:** Course days (i.e., Monday, Wednesday, and Friday) **Team B:** Every Wednesday	Everyday
	Duration	15 minutes	15 minutes
	Format	The same as the original Scrum	Three questions to synchronize the project progress
	When	The beginning of every class	The beginning of every working day
Role	Scrum master	Each student will practice Scrum master role in one Sprint.	The Scrum team has a designated Scrum master
	Development team	Based on the task assignments and student's skill sets, developer and tester roles could be changed between Sprints	The developer and tester roles are fixed according to the employee's job description
	Product Owner	Student (with instructor's supports)	Designated product owner

Offering realistic project experiences is one primary purpose of the capstone project course. It is crucial to understand how much influence the industrial partner provides. Therefore, in addition to investigating the research questions above, we also want to examine the following three things: (1) how students view the support from the industrial partner, (2) how the industrial partners assess the student team's performance, and (3) what specific help is provided by the industrial partner.

4.4. *Procedure and Data Collection*

Case study is our research method that is used to investigate the above research questions. It is a methodology conducted by giving special attention to completeness in observation, reconstruction, and analysis of the cases under study (Zonabend, F., 1992) and is recommended for exploratory, explanatory, and descriptive studies (Yin, 2012; Yin, 2009; Stake, 1995). With the consideration on exploring, observing, and analyzing the team under the adapted Scrum, we believe the case study method will offer us the significant details that can be used effectively to enhance learning.

Table 6 displays how we prepare for the case studies, how the adapted Scrum is handled in the class, and how data is collected. At each Sprint, in addition to entering the user stories in ClickUp, the Scrum master also needs to write a report to summarize each finished Scrum event. Since the instructor was with the team at every event, student's activities were recorded and evaluated. Then, the feedback of the overall Sprint execution was shared with student teams after Sprint was finished. The data collection does not include server course outcomes. Development-related materials such as client interview notes, requirement analysis reports, and UI design wireframes are not col-

Table 6
Data collections at each stage

Stages	Activities	Data Collection
Pre-Sprint	User story trainings, client interviews, requirement analysis and design discussions	No data collection at this stage
Scrum Execution	• Iterative development from Sprint 1 to 5 • Mid-term project review	(1) Scrum master planning report (2) Scrum master review report (3) Scrum master retrospective report (4) Scrum master stand-up meeting report (5) Instructor's evaluation (and feedback) note
Final	Final project review	(1) Two surveys: - Self and peer reviews - Learning reflection (2) All source codes (3) All the user stories (4) Instructor's evaluation (and feedback) note

lected for case studies. However, they served as references for data analysis, mostly used for instructor's evaluations. The client evaluation is not included in the data collection because it is mostly about their satisfaction with the product, which has no observations on teams.

4.5. *Measurement*

The qualitative analysis is applied to all the Scrum reports, instructor's evaluation, and some survey questions. We used quantitative analysis on source codes and the result is for validating our observations. We will calculate the number of line changes by combining added and deleted lines in each Sprint and normalizing the changes to help us validate our observations of the team's activities (see Eq. (1)). We also look at commit history in a Sprint to know student's work patterns.

Nikolaus *et al.* (2009) stated that measuring changing lines of code is a useful metric for software evolution. The evolution includes structure changes, bug fixes, and new feature implementation. To know what students learned results in causing the code modification, counting line changes is used. If we pair the results with our qualitative analysis, it should increase the confidence level of our conclusion and the findings on our qualitative analysis.

$$Normalized\ Change = \frac{(Added\ Lines + Modified\ Lines + Deleted\ Lines)}{Max\ (Added\ Lines + Modified\ Lines + Deleted\ Lines)} \quad (1)$$

Other quantitative measurements follow the burndown chart within a Sprint, which is often suggested in practice. We examine the completeness rate of each Sprint because finished tasks or user stories is often updated during or after daily stand-up meetings in practice and our stand-up meetings do not happen every day.

Table 7
Instructor's assessments on competencies

Category	Instructor's (Expert's) Assessment on Competencies
Problem-Solving Skills	Using knowledge Researching, Investigating, Problem-Solving Developing solutions
Contribution and Attitude	Taking initiative and responsibility Their planning and response to schedule pressures
Ability to Communicate	Communicating with team members Interacting and presenting with the client
Fulfillment	Understanding of main duties and responsibilities and fulfill them
Accountability and Commitment	Their most important achievements of the phase of development

We learned from the software engineer competencies suggested by Turley and Beiman (1994, 1995) and developed our assessment matrix (see Table 7). This assessment matrix uses a scale from 0 to 10 to evaluate each competency for each student. The value from 10 to 8 means exceptional, from 8 to 6 means good, from 6 to 4 indicates development, and below 4 means under-development. As experts, instructors can assess individual problem-solving abilities from every Scrum review event and all the code review activities.

5. Results and Discussions

In this section, our discussions include the investigations on our research questions, the evaluations on the industrial partner's involvement, the scalability of our proposed approach, and additional findings along with our case study.

5.1. *Research Question Investigation*

The research questions are listed under four different aspects. In the following sections, we will discuss each aspect our case study investigates.

5.1.1. *Scrum Learning*

We selected some questions from the student's final survey (see Table 8) to analyze the student's learning about Scrum. From the answers to question 1, 2, and 5, we discovered student feedback regarding Scrum events. First, students learned most from the review and demo. We believed that working with industrial partners indeed brought different perspectives from their teaching staff. Students did not know their misinterpretation on the business needs until they released some wireframes or prototypes to the client in the demonstration. Industrial partners often provided end-user viewpoints on student's user

interface implementations. Students were often encouraged by the client when they had done something right. These were situations that we noticed from the demo and review events. Second, students claimed to learn a lot from planning and retrospective. The planning event forced team members to have internal discussions on whom to work what and examine everyone's responsibilities and the retrospective event led the team to have reflections on how to improve the process or development. These events also promote team collaboration through communication.

Table 8

Feedback about Scrum events and learning from Team A and B

Question 1: *What Scrum events did you learn most from the course? (Please select at most two)*		
Team A	Most	Retrospective Planning
	Second most	Review and demo Retrospective
Team B	Most	Daily stand-up Planning Grooming Retrospective
	Second most	Review and demo Retrospective
Question 2: *What Scrum events did you think most useful for conducting your project in the course? (Please select at most two)*		
Team A	Most	Daily stand-up Planning Grooming Review and demo
	Second most	Daily stand-up Planning
Team B	Most	Daily stand-up Planning Grooming Review and demo
	Second most	Daily stand-up Planning Review and demo
Question 3: *Do you also want to learn more about SCRUM?*		
Team A	100%	Yes. Definitely
Team B	75%	Yes. Definitely
	25%	Maybe. I might be since I am not very interested in.
Question 4: *Maintaining a Product Backlog List is critical to me for the project?*		
Team A	50%	Yes
	50%	I consider to be a team job
Team B	75%	Yes
	25%	I consider to be a team job

Continued on next page

Table 8 – continued from previous page

Question 5: *I reflect and think again about my project and work every time during and after the Retrospective event*		
Team A	50%	Yes, always
	25%	Sometimes
	25%	Occasionally
Team B	25%	Yes, always
	50%	Sometimes
	25%	Depends on the issues or discussions
Question 6: *Do you think the entire team user story point estimation is getting accurate after a couple of Sprints?*		
Team A	75%	Maybe yes (I feel like that)
	25%	Yes
Team B	100%	Maybe yes (I feel like that)
Question 7: *Do you think that your user story point estimation skill improves (i.e., providing more accurate points) after a couple of Sprint?*		
Team A	100%	Maybe yes (I feel like that)
Team B	50%	Yes
	50%	Maybe yes (I feel like that)
Question 8: *Do you think it is good to have two weeks for the Sprint?*		
Team A	100%	Yes
Team B	100%	Yes
Question 9: *How effective was the Daily Scrum event?*		
Team A	50%	It was good. That reminds me my job to be done.
	50%	It was bad. Heavy loading.
Team B	50%	It was okay. We are more like reporting, sometimes, we synchronize.
	25%	It was good. That reminds me my job to be done.
	25%	It was good. We synchronized with each other.

When we investigate the feedback on the product backlog item list, the user story, and Sprint duration, we could justify that our adapted Scrum did not decrease the effectiveness of executing the Sprint, the product backlog maintenance, and the user story point estimation. We believe that keeping the same recording format and Sprint practices from the original Scrum in our adaption is the main reason. Repeating several times of exercises made students improve these skills as well.

From question 3, students expressed their strong interest in learning Scrum. We could also discover some details when students mentioned the three things they learned most from the Scrum. These answers again enhanced our statements on student's learning about Scrum. Students applied every element of adapted Scrum and learned from them very well. Overall, our adapted Scrum did help with the project development and did encourage student's learning on Scrum.

The key difference between our adapted Scrum and Scrum is the stand-up meeting frequency and this changed frequency is also inevitable against the Agile principle. Under the limitation caused by the course setting and instructor's in-person participation,

Table 9
Selected feedback about Scrum learning through adapted Scrum

Question: *List at least three things you learned most from the Scrum*	
Answers	1. Keeping up to date with teammates, 2. Staying organized on what I need to do and what teammates are doing, 3. Reviewing and reevaluating process standards
	Be flexible, communication is critical, don't be afraid to discard work
	Epics can be broken down and completed faster, team communicates more, user story points are agreed upon among the team
	Story points estimation, self-organization, Commitment to completing project, increased productivity
Question: *Overall, what do you think about the Scrum learning experience?*	
Answers	I enjoy it and think it is a very useful software development framework.
	It was very helpful as I know now how teams work under Scrum and will use it in the future.
	Scrum was basically something new we learnt in thus course, although I have used SCRUM in my internship, learning about it helped to understand why it is being done and I got to learn more about it than you will learn when using it. I think the scrum learning should continue in future sections of this course as to me, it seems to work very well.

we can only make team A have a stand-up meeting every course day and team B have every Wednesday as the meeting frequency.

From our observation, having a stand-up meeting every day is ineffective. Students could not spend time working on the project between two consecutive meetings. In their meeting notes, you can often see the following reasons or responses:

> *"**What did you do yesterday?** Worked on the database stuff some more. What will you do today? Nothing. Cannot spend time on the task."*

> *"**What did you do yesterday?** hasn't worked on the project yet this week, too much other stuff to do"*

Team A's result tells us that students have their schedule to distribute their working time for all their assignments, tests, readings, or work. It might not be reasonable to require them to work on the project almost every other day. In contrast, we do not find those kinds of answers in their logs. We found that team B could reasonably distribute their personal time to work on the entire course works or tests so that students formed their project working and collaboration pattern between two consecutive stand-up meetings.

After the team's commit history is viewed, the consequences caused by the difference between student's and industrial professional's working patterns can be revealed. Agile development claims a weekly 40-hour work; this is, professionals can work on the development every weekday. From the commit histories of both teams, most of the commits were on Friday, Saturday, and Sunday. That means that student teams mostly do their system development around weekends. This implies that the weekend could be their best time to work together because team members find available free time. Their stand-up meeting and review notes also show the same pattern regarding their working time.

Table 10

Selected student's feedback about Adapted Scrum practices and learning

Question: *Given the course limitations (ex: time or course arrangement), do you think SCRUM is executed properly in the course? (Please explain why whether YES or NO)*	
Answers	Yes, it was fairly effective even though we weren't able to daily standups. We were still able to practice the 5 scrum events listed above at least once per sprint.
	I think we could have spent a little more time doing stand ups, but given the time restrictions I understand why we didn't. I think more time on this would've helped us stay on the same page.
	I think it is, since we were working on a website already created and planned out, it was easier for us to do planning and reviews. Also the time worked well although it wasn't enough because of other classes and commitments. But the arrangement in general seemed to work for the class, also with our class size, we were able to conduct meetings, reviews and planning in a specified class time.
	Yes, everyone worked in every role.
Question: *What would be your suggestions on learning SCRUM?*	
Answers	Maybe do a stand-up form on days that we can't do it in class.
	I will say, SCRUM is best learnt by doing it. My original exposure to SCRUM was confusing for the first week because I did not understand why it was used even after explanations, but as I continued to do it, I learnt it well and also the knowledge of why we use it.

However, from Table 10 where we gather student's answers about the overall changes to the original Scrum, regarding stand-up meeting frequency, the selected answers show a controversial response. Students would like to have more stand-up meetings in a Sprint to synchronize team member's statuses, to remind individual's responsibilities (see question 9 in Table 8), and to learn more about Scrum, but they also understood the limitation. This reminds us of a possible or flexible adjustment on stand-up meeting frequency if students expect more face-to-face short meetings. To conclude the overall experiences on the adapted Scrum, we think it did not impact project execution, and students were satisfied with the overall outcomes.

Providing Scrum practices to students with industrial customers improved their understanding of the Scrum framework. Executing these events under the adapted Scrum framework did not decrease understanding of Scrum. Table 11 shows that team A and B improved their grades by 10.4% and 40.9%, respectively. The normalized grades were scaled from 0 to 1 according to the results of students' actual scores from all the Scrum-related exam questions divided by the questions' full scores.

The improvement can be validated on the Scrum-related exam questions. We particularly estimate how much Scrum hands-on experience is accumulated at the time when questions are asked. Students took the midterm exam around the end of Sprint 2 and took the final exam after five Sprints. The timing indicates that practicing the adapted Scrum multiple times aids in understanding Scrum better. Students knew the Scrum event durations were modified in the classroom because both teams received high scores in the Scrum execution questions.

We also want to know what competencies improve during or after the Scrum events in addition to Scrum learning. Instructors also assessed student's learning according to our measured competencies. From the measurement results, instructors also know that students improved competencies in the "ability to communicate" and "fulfillment" cat-

Table 11
Student Grade Improvement on Scrum Related Questions

Team A	**Midterm Questions**	Continuous Improvement and Scrum Principles
	Normalized Grades	**Average:** 0.7785 **Standard Deviation:** 0.1075
	Final Questions	Scrum Planning and Execution Questions
	Normalized Grades	**Average:** 0.8705 **Standard Deviation:** 0.1645
	Improvement Rate*	10.4%
Team B	**Midterm Questions**	Scrum Basics, Continuous Improvement, and Scrum Pros and Cons
	Normalized Grades	**Average:** 0.63 **Standard Deviation:** 0.2732
	Final Questions	Scrum Planning and Execution Questions
	Normalized Grades	**Average:** 0.8875 **Standard Deviation:** 0.075
	Improvement Rate	40.9%

Note* $\text{Improvement Rate} = \frac{(\text{Average Final Grades} - \text{Average Midterm Grades})}{\text{Average Midterm Grades}}$

egories. We believe the improvements are earned from the lessons learned in the review and retrospective in the first two or three Sprints because students realize that they need to increase communication quality to reduce the overlapping work or misinterpretations.

Regarding "problem-solving" competency, we also found that students needed guidance to lay out the software architecture so that they could divide the system into modules for an individual to work on. This deficiency was evident to us when we observed student glooming activities. When students encounter a real-world project, how to design the architecture becomes very challenging. Once students worked on a module, developed a piece of the system, or saw the skeleton codes for the system, they could actively propose a proper solution and know what to research. These behaviors can be easily seen during the planning and retrospective events.

5.1.2. *Team Project Performance*

Story point completion analysis (Fig. 3 and Fig. 4) provides insights into the team performance. Both teams showed that all user stories were fulfilled in the first Sprint because teams started with user stories that they thought "they could finish" or were more about research about the system or technologies. However, the first Sprint user stories and task assignments are conservative. In Sprint 2, both teams had substantial unfinished user stories. We think it is not a coincident phenomenon for both teams even though various factors were causing this gap. Students tended to underestimate their task challenges after a successful Sprint 1 and still learned how to estimate user stories. Then, those unfinished tasks were carried over to Sprint 3 and the later Sprints. Team A finally finished all the required functions at the end of the last Sprint. Team B's unfinished user stories became more in Sprint 4. According to our observations and discussions in the Sprint

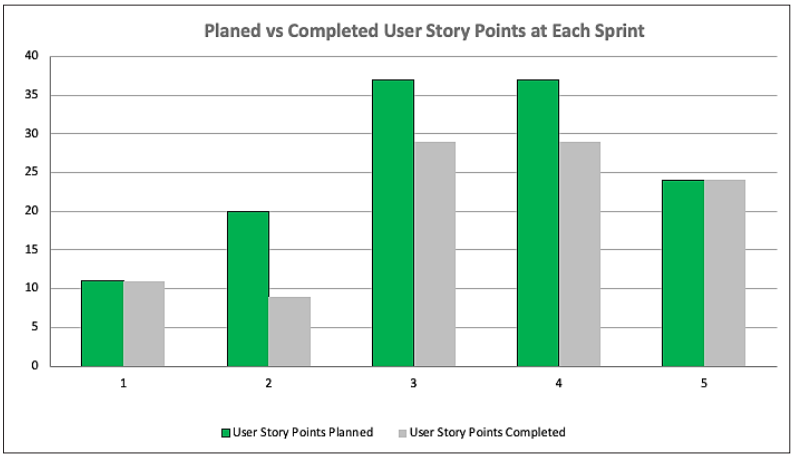

Fig. 3. Team A Story Point Analysis.

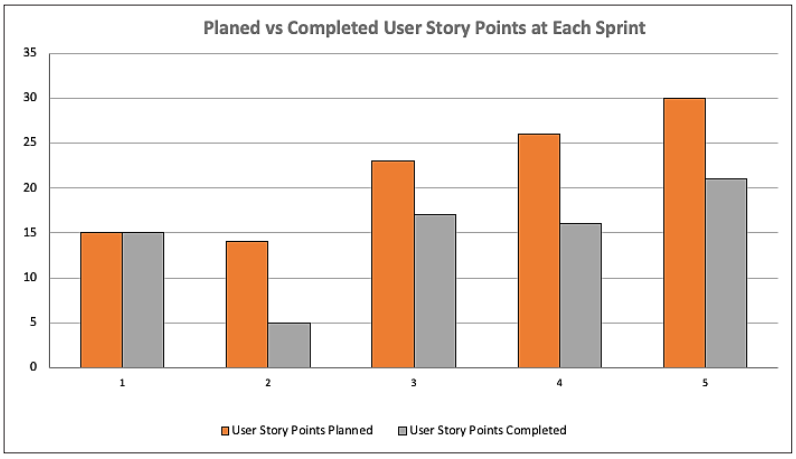

Fig. 4. Team B Story Point Analysis.

Retrospective, we knew one team member could not finish his job due to a personal issue. As a result, the user story point gap could not be filled.

Moreover, according to the tasks specified in the first two Sprints, one important factor that impacted the team performance is individual technical skill. Many user stories could not be finished due to working on the research on the required technologies. However, learning new technologies also turned out to be the parts students like. We can discover this impact in the following quotes from student's reflections.

"I like it"

"Useful"

"Interesting"

"Getting better after a couple of practices or task implementations."

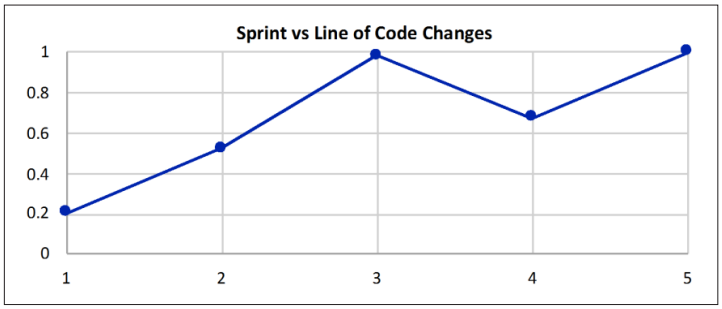

Fig. 5. Team A Code Analysis.

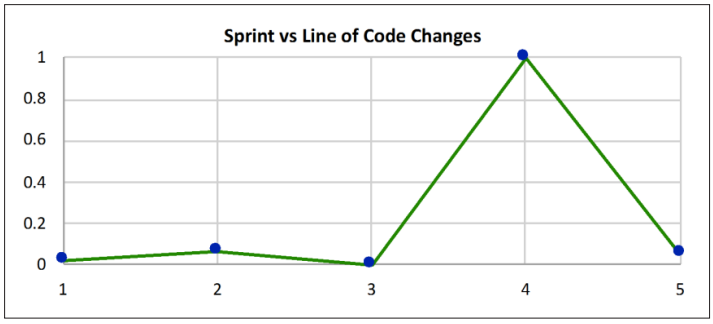

Fig. 6. Team B Code Analysis.

The above findings in Sprint 3 to 5 can be cross-validated by looking at source code analysis (see Fig. 5 and Fig. 6). We can see that a relatively large number of codes changed during these Sprints. This also indicated that, after first few Sprints, both teams had more confidence about what needed to be done and became more productive. Then, individual commitments on user story completions slightly increase.

5.1.3. *Member Satisfaction*

Team B felt very satisfied during Sprint 4 and 5, although many unfinished user story points existed. The main reason is that the team accomplished a couple of challenging features such as online lessons and avatar talking beside the lecture playing video. Their work and learning satisfaction can be found in their Sprint review and retrospective notes. We can see the selected student's statements from the following.

> *"**What did we do well?** Make progress on the lesson viewer and editor. Getting the database working without issues. Everyone can kind of work on the job."*

The planning and grooming events enabled the team to have a modular design that increased their confidence in their user story estimation. In Fig. 6, we also see many line

changes in the repository in Sprint 4. This is also evidence of the difficult feature that the team overcomes. In team B's Sprint 5, the team worked on bug fixes and relatively small improvements due to finishing a very productive Sprint, so the amount of code changes is small.

In contrast, team A finished all the planned user stories at the end. We suspected it was due to the time pressure on the project completion because many of their works were done without good quality. We saw many defects or immature considerations, for example, how real-time message was handled in their product, in the final project review. Starting from Sprint 3, team A became productive to compensate for those carry-over user stories. In their survey, all team members liked the problem domain of the project because they learned the technologies (i.e., real-time message system) that they did not know very well before this project. However, the complaints came from the project execution. Two team members complained about insufficient exercises of being a Scrum Master due to role rotation, and one team member thought the stand-up meeting added substantial efforts to manage the project. This claim can also be referred to question 9's responses in Table 8 because 50% of responses pointed out the heavy loading. Therefore, intensive stand-up meetings introduced some pressure on student's work, which could influence individual satisfaction.

5.1.4. *Team Improvement*

According to the Sprint event notes, team B enhanced the collaborations in Sprint 4 and 5. The team improvement happened after Scrum retrospective meetings. During the meetings, team members discussed how to improve their process and collaboration. The source code analysis on their repository also serves as referential support of this because individual changes were merged and deployed. Another individual improvement happened after the Scrum review. The client provided experiences on online teaching and tests to help students. Students worked together to study the way to satisfy the customer's needs. From peer reviews, we also saw students talked about their improvement after using Scrum adaption (see their responses below). As a result, students felt that they improved their capabilities in the end.

> *"I have definitely gone beyond work done."*

> *"I think I have taken on several important tasks and delivered them on time and in good quality. I learned a lot from that and knew better."*

On the other hand, team A's programming tasks are mainly on two members. Although the team does not have any complaints about their collaborations and workload distribution for their peer review. It is suspicious that their peer reviews did not reflect their true feeling about their team members. Scrum events also helped communication and let team members know everyone's status. In the class, technical tasks that required more workloads would fall on those team members who were more capable of finishing them to receive good grades. So, the way that team A collaborated seemed only beneficial to those students who wanted to spend efforts on their skill improvement. These

discoveries could also be found in the team's peer reviews. This tells us that Scrum events can bring team communications but individual commitments on tasks and shared responsibilities bring individual improvements.

5.2. *Evaluations on Industry Involvement*

Recent studies such as Lundqvist *et al.* (2019), Linos *et al.* (2020), and Sroka (2020) emphasized that having an information technology (IT) professional in Scrum is significantly beneficial to student's learning. These IT professionals become a stakeholder who can mentor the students by bringing in possible solutions from an industrial perspective and equip them with professionalism.

We want to argue that collaborating with technical mentors leads to putting our eyes on the technical side of learning. Working with industry should let students not only learn the method to deal with complex engineering problems but also understand the business side of the project and know how to communicate with a non-technical audience.

Hence, from the perspective of knowing the business, one might criticize that most industry-provided projects in the course are not critical to their core business. Therefore, students might not learn about the company's actual business from the capstone project. However, we believe students can still learn about the company's business through working on these projects. Often, conversations with the client reveal the company's important business problems. Students result in gaining business insights. This is particularly valuable to students because they realize that the knowledge learned from prior courses can be related to real-world business. In our case studies, the online real-time chatting system developed by team A was just a side project to the main business and the online classroom system built by team B was an additional experiment for the company. But, team A ended up understanding how the traditional customer service workflow and team B realized the critical elements for guiding students to plan their careers.

Capstone projects need room for failure because students are not professional engineers. Our industrial clients tended to compliment students even when students made no promising progress in Sprints. They did not hold the same professional standard to evaluate student's performance. This kind of client attitude can be a double-edged sword. On the one hand, students understand that they can make mistakes; on the other hand, irresponsible behaviors are encouraged. As a result, this requires instructors to put effort into holding students accountable.

Both teams gradually improved their communication competency when we saw that they interacted with clients during the Sprint review events or the final product presentation. Having a non-technical client trains student's communication skill. For example, students realize the importance of visual aid. They learned to use a wireframe to get feedback from the client. After the first two Sprints, both teams used the public testing site on the cloud to discuss the increment deliverable with the client. This also encouraged students to do more tests to ensure the quality before the demo.

5.3. *Scalability of the Adapted Scrum*

We envision that scaling our adapted Scrum approach is a challenge because the given condition of our proposed approach states that all the Scrum events should be handled during class time. Adding more instructors or starting Sprint at a different time (day) seems to be the intuitive choice. However, the assumption limits the scalability of the approach.

Although students were asked to take notes in the Sprint event reports, we think having a written notebook is insufficient. It is crucial to investigate student behaviors from their face or body language during the Scrum events. Therefore, our proposed adjustment is to ask the team to record the events handled in different places at the same class time. The instructor attends one of his or her mentored teams in person and watches the video afterward for evaluations.

5.4. *Additional Discoveries*

Retrospective meetings focus on good and poor practices about the process during the Sprint. However, students were often confused about its purpose, so their discussions were on personal performance instead of articulating the possible reason causing that and potential solutions to be tried in the next Sprint. Another observation about Scrum practices is student's user story estimation. We found that both teams can form a pattern to estimate story points after Sprint 4. The exercises of reaching the team consensus about the user points in the previous Sprint did help them to understand and gauge their user story in the later Sprints.

We noticed that students learned the importance of software architecture and design from our survey results. Having a software architecture lets them have an overall view of the system. They figured out this is a critical design activity to refine the user story. Students think a glooming is crucial for their task distributions, later system integration, and testing. Finally, students gave positive feedback about their learning experiences. They did not feel any inadequate handling regarding our adapted Scrum. They claimed that they knew better about the Scrum framework, knew the differences between adapted and original Scrum, and realized why Scrum needs to be changed in our project management.

6. Conclusion

The Scrum adaptation decision process was applied well to our capstone projects. Based on our discussions on various aspects, we know that students still practice Scrum events and learn well from them. Both teams claimed they learned most from Scrum review and retrospective events because industrial partners provided valuable comments to students and the team could have meaningful communications and reflections. Industrial partners provided helpful information to students. From another aspect, students improved their

user story estimation after a few Sprints. More accurate estimation also made more task completions in the later Sprints during the development.

Our adapted Scrum keeps original formats, suggested Sprint duration, and frequency except for stand-up meetings. Therefore, from the final course surveys, student's expressions on Scrum learning also implied that students were encouraged to know more about Scrum. The survey questions about the stand-up events and comparison between two different frequencies showed that the events offer effective team synchronization and reminders on personal tasks.

Along with the discoveries about applying adapted Scrum to the capstone project, teams recognized the importance of the software architecture and design. Having an efficient glooming event is critical to product development and Scrum execution. To conclude the overall adapted Scrum experiences that we investigate from these two case study groups, our proposed adapted Scrum can be used in the software product development in the class to provoke team performance, student learning, team improvement, and satisfaction. Our discovered insight can be a guide to other capstone courses.

6.1. *Limitations*

This section highlights some limitations of this work. We understand that case study students from the same course might not allow us to generalize our findings. Students only went through five Sprints. We could not have opportunities to observe the long-term effects on the project development under adapted Scrum. The time limitation also made the team ignore adding more integration tests to enhance the product quality. Nevertheless, at the same time, both teams did not have absences of any member in any Scrum event during the entire project development. This fact aids us in collecting observations and evaluations on teams. We also knew that the two teams worked on two different projects and different systems brought different challenges to students. So, we only focused on teamwork and learning discussions. The evaluation was done only by the instructor – that is why we need to use other notes to support our findings.

6.2. *Future Work*

Although our discoveries in the case study can help handle future capstone courses, we also understood that our stand-up frequency required advanced research, particularly about how flexible the interval can be. One discovery is that students need more time to practice their Scrum planning and grooming. It is helpful in further specifying the reasonable event duration according to the student's need. In addition, we also want to conduct a comparison experiment where one team will use regular Scrum and the other will use Adapted Scrum to build a system under the same requirements so that we can examine what competencies can be improved. Lastly, we want to study how software architecture influences the student's overall project experience and their competencies when the Agile development process is employed.

References

Baer, N. & Zeidman, B. (2009). Measuring Software Evolution with Changing Lines of Code. *Proceedings of the ISCA 24th International Conference on Computers and Their Applications, CATA 2009*. 264–170.

Baham, C. (2019). Teaching Tip: Implementing Scrum Wholesale in the Classroom. *J. Inf. Syst. Educ.*, 30, 141–159.

Baird, A., Riggins, F.J. (2012). Planning and Sprinting: Use of a Hybrid Project Management Methodology within a CIS Capstone Course. *Journal of Information Systems Education*, 23(3), 243–258.

Batra, D., Satzinger, J.W. (2006). Contemporary Approaches and Techniques for the Systems Analyst. *Journal of Information Systems Education*, 17(3), 257–266.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, W., Cunningham, A., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). *Agile Manifesto*.

Begel, A., Simon, B. (2008). Novice Software Developers, All Over Again.

Christov, S.C., Hoffman, M.E. (2019). Experiential learning of software project management and software development via course collaboration. *SIGCSE 2019 – Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 160–166. https://doi.org/10.1145/3287324.3287457

Coppit, D., Haddox-Schatz, J.M. (2005). Large team projects in software engineering courses. *SIGCSE Bull.*, 37(1), 137–141.

Devedzic, V., Milenkovic, S.R. (2011). Teaching Agile Software Development: A Case Study. *IEEE Transactions on Education*, 54, 273–278.

Devlin, M., Phillips, C. (2010). Assessing competency in undergraduate Software Engineering teams. 271–278. 10.1109/EDUCON.2010.5492569.

Hoskey, C., Hoskey, A. (2016). Cultivating Sprightly Students: Using Agile Development in an Information Systems Capstone Course. *Information Systems Education Conference*. Pittsburgh, PA..

Igaki, H., Fukuyasu, N., Saiki, S., Matsumoto, S., Kusumoto, S. (2014). Quantitative assessment with using ticket driven development for teaching scrum framework. In: *Companion Proceedings of the 36th International Conference on Software Engineering, ICSE Companion 2014*, pp. 372–381, New York, NY, USA. https://doi.org/10.1145/2591062.2591162

Jiménez, O., Cliburn, D. (2016). Scrum in the undergraduate computer science curriculum. *Journal of Computing Sciences in Colleges*, 31(4), 108–114.

Ju, A., Hemani, A., Dimitriadis Y., Fox, A. (2020). What agile processes shouldwe use in software engineering course projects? *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 643–649. https://doi.org/10.1145/3328778.3366864

Jurado-Navas, A., Munoz-Luna, R. (2017). Scrum Methodology in Higher Education: Innovation in Teaching, Learning and Assessment. *International Journal of Higher Education*, 6(6), 1–18.

Kamthan, P. (2016). On the Nature of Collaborations in Agile Software Engineering Course Projects. *International Journal of Quality Assurance in Engineering and Technology Education (IJQAETE)*, 5(2), 42–59.

Kharitonova, Y., Luo, Y., Park, J. (2019). Redesigning a software development course as a preparation for a capstone: An experience report. *SIGCSE 2019 – Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 153–159. https://doi.org/10.1145/3287324.3287498

Kropp, M., Meier, A. (2013). Teaching Agile Software Development at University Level: Values, Management, and Craftsmanship. In: *Software Engineering Education Conference, Proceedings*, pages 179–188.

Kropp, M., Meier, A., Biddle, R. (2016). Teaching agile collaboration skills in the classroom. In: *2016 IEEE 29th international conference on software engineering education and training (CSEET)* (pp. 118–127). IEEE.

Linos, P.K., Rybarczyk , R., Partenheimer, N. (2020). Involving IT professionals in Scrum student teams: An empirical study on the impact of students' learning. *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp.1–9.

Lundqvist, K., Ahmed, A., Fridman, D., Bernard, J. (2019). Interdisciplinary Agile Teaching. *2019 IEEE Frontiers in Education Conference (FIE)*, Covington, KY, USA, pp. 1–8.

Magana, A.J., Seah, Y.Y., Thomas, P. (2018). Fostering cooperative learning with Scrum in a semi-capstone systems analysis and design course. *Journal of Information Systems Education*, 29(2), 75–92.

Mahnic, V. (2010). Teaching Scrum through Team-Project Work : Students' Perceptions and Teacher's Observations. *International Journal of EngineeringEducation*, 26, 96–110.

Mahnic, V. (2012). A capstone course on agile software development using scrum. *IEEE Transactions on Education*, 55(1), 99–106. https://doi.org/10.1109/TE.2011.2142311

Masood, Z., Hoda, R., Blincoe, K. (2018). Adapting Agile Practices in University Contexts. *Journal of Systems and Software*, 144, 501–510.

Matthies, C., Kowark, T., Richly, K., Uflacker, M., Plattner, H. (2016). How Surveys, Tutors and Software Help to Assess Scrum Adoption in a Classroom Software Engineering Project, *IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 313–322.

Matthies, C., Kowark, T., Richly, K., Uflacker, M., Plattner, H., (2016). How surveys, tutors, and software help to assess Scrum adoption in a classroom software engineering project. In: *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16)*. Association for Computing Machinery, New York, NY, USA, 313–322. https://doi.org/10.1145/2889160.2889182

May, J., York, J., Lending, D. (2016). Play ball: bringing scrum into the classroom. *Journal of Information Systems Education*, 27(2), 87–92.

McAvoy, J., Sammon, D. (2005). Agile Methodology Adoption Decisions: An Innovative Approach to Teaching and Learning. *Journal of Information Systems Education*, 16(4), 409–420.

Nadler, D., Hackman, J.R., Lawler, E.E. (1979). Managing organizational behavior.

Owens, D., & Shekhar, G. (2018). Using SCRUM principles to transform the classroom.

Paasivaara, M., Lassenius , C., Damian, D., Raty, P., Schroter, A. (2013). Teaching students global software engineering skills using distributed scrum. In: *Proceedings of the 2013 International Conference on Software Engineering, ICSE '13*, pages 1128–1137, Piscataway, NJ, USA. IEEE Press.

Rush, D. E., Connolly, A. J. (2020). An agile framework for teaching with scrum in the IT project management classroom. *Journal of Information Systems Education*, 31(3), 196–207.

Scharf, A., Koch, A. (2013). Scrum in a software engineering course: An in-depth praxis report. *Software Engineering Education Conference, Proceedings*, 159–168. https://doi.org/10.1109/CSEET.2013.6595247

Schwaber, K. (1995). SCRUM Development Process. *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*.

Schwaber, K., Sutherland, J. (2020). The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100

Sharp, J.H., Lang, G. (2018). Agile in teaching and learning: Conceptual framework and research agenda. *Journal of Information Systems Education*, 29(2), 45–52.

Song, H., Chien, A. T., Fisher, J., Martin, J., Peters, A. S., Hacker, K., Singer, S. J. (2015). Development and Validation of the Primary Care Team Dynamics Survey. *Health Services Research*, 50(3), 897–921. DOI: 10.1111/1475-6773.1225

Sroka, C. (2020). *The Importance of Agile Methodologies Within Student-led Industry Projects*. June 13, 2017, Center for Digital Media. Accessed Feb. 19, 2020. https://thecdm.ca/news/the-importance-agile-methodologies-within-student-led-industry-projects

Stake, R.E. (1995). *The Art of Case Study Research*. SAGE Publications.

Umphress, D.A., Hendrix, T.D., Cross, J.H. (2002). Software Process in the Classroom: The Capstone Project Experience. *IEEE Software*, 19(5), 78–81.

Wagh, R. (2012). Using Scrum for Software Engineering Class Projects. AGILE India, 68–71, IEEE.

Weber, E. (2016). Performance Learning of Agile Methodology Using Paired Courses of Systems Analysis and Design and Web / Mobile Programming. *EDSIG Conference*. Las Vegas, NV.

Yin, R.K. (2009). *Case Study Research: Design and Methods*. SAGE Publications.

Yin, R.K. (2012). *Applications of Case Study Research*. SAGE Publications.

Zonabend, F. (1992). The Monograph in European Ethnology. *Current Sociology*, 40(1), 49–54. https://doi.org/10.1177/001139292040001005

H.-F. Chang (R.B. Annis School of Engineering, University of Indianapolis, Indianapolis, Indiana, United States) holds a Ph.D. in Computer Science from the University of Southern California, a M.S. in Computer-aided Engineering from Carnegie Mellon University, and a M.S. in Civil Engineering from National Taiwan University. He has more than ten years of software development experience in the industry. His work involves various software applications covering big data, the Internet of Things (IoT), recommendation systems, embedded systems, natural language processing, and model-based code generation. He developed an approach for designing complex software systems, analyzed development through mining the source codes from multiple version control repositories, and created IoT-based sensor devices and a cloud-based platform for real-time monitoring. His research interests cover software engineering, learning technologies, computer science education, intelligent systems, and artificial intelligence. His recent studies aim at the integration of these three fields. He is currently an assistant professor in the R.B. Annis School of Engineering at the University of Indianapolis and an Institute of Electrical and Electronics Engineers (IEEE) senior member.

M. Shokrolah Shirazi (R.B. Annis School of Engineering, University of Indianapolis, Indianapolis, Indiana, United States) received his PhD degree in electrical and computer engineering from University of Nevada, Las Vegas (2016). He received his BS degree in computer engineering from Ferdowsi University of Mashhad (2005) and his MS degree in computer architecture from Sharif University of Technology, Tehran, Iran (2007). Dr. Shirazi in currently an Assistant Professor of R.B. Annis School of Engineering at University of Indianapolis. Prior to join Uindy, he worked as a visiting assistant professor at Cleveland State University and he was the member of the "real-time intelligent systems laboratory" at UNLV. Dr. Shirazi researches in computer vision, machine learning, embedded systems and their applications in intelligent transportation systems, robotic and health. His dissertation "Vision-based Intersection Monitoring: Behavior Analysis & Safety Issues" received two awards as the second-winner place for UNLV, College of Engineering and the IEEE ITSS Best Dissertation Award in 2016.