Revision submitted to the Special Issue on Software Engineering Education and Training, Journal of Systems and Software, May 2018
The final authenticated version is available online at: https://doi.org/10.1016/j.jss.2018.07.011

Adapting Agile Practices in University Contexts

Zainab Masood[1], Rashina Hoda, Kelly Blincoe

SEPTA Research, Department of Electrical and Computer Engineering,

The University of Auckland, Auckland, New Zealand

Abstract

Teaching agile practices has found its place in software engineering curricula in many universities across the globe. As a result, educators and students have embraced different ways to apply agile practices during their courses through lectures, games, projects, workshops and more for effective theoretical and practical learning. Practicing agile in university contexts comes with challenges for students and to counter these challenges, they perform some adaptations to standard agile practices making them effective and easier to use in university contexts. This study describes the constraints the students faced while applying agile practices in a university course taught at the University of Auckland, including difficulty in setting up common time for all team members to work together, limited availability of customer due to busy schedule and the modifications the students introduced to adapt agile practices to suit the university context, such as daily stand-ups with reduced frequency, combining sprint meetings, and rotating scrum master from team. In addition, it summarizes the effectiveness of these modifications based on reflection of the students. Recommendations for educators and students are also provided. Our findings and recommendations will help educators and students better coordinate and apply agile practices on industry-based projects in university contexts.

Keywords: Agile Software Development; Agile Practices; Teaching; University; Adapting; Contextualization

* Corresponding author. Tel.: +64 9 923 1377; Fax: 0800 61 62 64.
E-mail address: zmas690@aucklanduni.ac.nz.
Physical address: Building 903, 368 Khyber Pass, New Market, Auckland 1023, New Zealand

1. Introduction

Agile courses have been taught at both the graduate and undergraduate levels using different approaches, such as theoretical lectures supplemented with lab sessions [1], workshops [2], and games [3]. One of the most common techniques is providing hands-on agile practice through team projects implemented by the students during the course [4-8]. Though teaching agile in a university context is acknowledged to be challenging [2, 9, 10], it is also seen as useful due to numerous benefits, such as gaining agile experience [5], customer coordination and collaboration [11], and improving job prospects [7]. It nurtures software development experience for students and helps them embrace agile practices on real projects [7].

Teaching agile in the university involves constraints that directly influence the practices followed and require adaptations to fit the university context. In this study, we present the constraints faced and adaptations made by university students working on a project within the course using agile and lean practices at the University of Auckland. We collected data over two iterations of the course, involving 135 students working in teams of 6-8 on 18 different projects.

This paper summarises the students' experiences and observations of learning agile practices through team projects and indicates how university students contextualized some of the agile practices during their projects to meet the daily and weekly challenges. We collected data through 135 individual reflection surveys and analysis of 75 student essays. We discuss the adaptations students made to fit their needs and their impact on practice, including both positive and negative. Based on the lessons learned, recommendations are presented for educators and students interested in tailoring agile for a tertiary course.

The remainder of the paper is structured as follows. Section II describes related works, section III lists the course structure and description, Section IV summarises the research methodology used in this study, section V presents the findings of this research throwing insights into constraints faced by university students while implementing agile practices during projects, the modifications to the agile practices made in a university context followed by their perceived effectiveness. Section VI discusses the findings and compares with related work with recommendations for educators and future students. Section VII concludes.

2. Related Works

Though agile software development has been around for more than a decade, teaching agile software development has only drawn significant attention in educational and research domains in the last few years [3]. Many researchers have acknowledged the need to teach agile software development in software engineering programs [11, 12] as a means to build social and ethical skills in addition to technical skills [12]. There is growing awareness that traditional theoretical lectures alone cannot help students to learn agile practices, rather students need to practically apply them for enriching learning and upskilling themselves [2, 13].

Teaching agile methods to software engineering students is reported to benefit in many ways, such as, hands-on positive experience of applying engineering practices such as Test Driven Development [4, 5], students learning to communicate with a customer [3, 5], motivating them to deliver a solution [5], building confidence and increasing the marketability of students as novice software engineers in industry [7]. However, exposing students to agile methods and practices in the university context comes with a set of challenges [2] including, differing student motivation and aims [3, 14, 15], limited availability and support from Product Owner/Customer [14, 15], lack of guidance from experienced coach (XP) or a scrum master (Scrum) role [5, 15, 16]. Some other challenges reported include short duration of courses with half of the course time being dedicated to teaching concepts cutting down time to work on projects, students required to take multiple courses in addition to their personal and professional commitments [14, 16]. As a consequence, sometimes students apparently gain experience in practicing these agile methods and practices, but they may not be able to apply them correctly [5, 15, 16].

Recent research on agile education elaborates on how instructors at universities have taught different agile methods and practices [2, 17] such as XP [3, 6, 18] and Scrum [1, 8], as optional or sometimes

mandatory courses to graduate and undergraduate students [19]. Some techniques adopted by these instructors include introducing agile theory in a traditional manner with lectures and laboratories [1] and exposure to literature on agile practices [17] while others include some practical experience through games [3], workshops [2], and interactive exercises [17]. Some universities incorporate agile practices into the curriculum through different scaled (small, medium and large) series of projects [6] while others have taken the approach of introducing agile theory in one semester followed by project course in the next [4]. Some allowed students to learn agile methods by working on projects under the supervision of tutors [8]. Very few collaborated with industry to gain practical experience of collaborating with real customers through projects [5]. Students working on existing industry systems is rare [20].

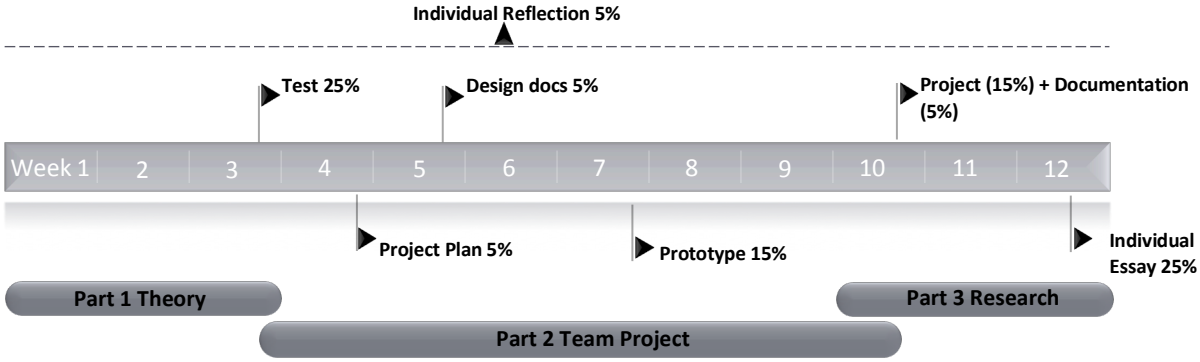

Figure 1 Course structure, assessment breakdown and timeline

Many educators have shared their experiences of teaching agile methodologies so that others can benefit from them [2, 7, 10, 19, 21, 22]. Some discussed their methods of teaching with challenges and issues faced [2, 7, 19], others reported their experiences around students' interaction with customers, weaknesses in teams, and imbalance in workload [21]. Few researchers shared challenges encountered by the students while applying agile methods to develop real projects and the lessons learned by the instructors through their experiences in the process [9, 10]. Some of these challenges reported by students are eliciting, structuring and communicating requirements [9], ineffective team communication due to busy schedules and planning issues due to the lack of experience and training [10]. In this study, we not only covered similar constraints faced by students in detail but also identified other constraints such as customer related issues, personal commitments, and the lack of students' dedication which hindered the embracing of agile practices in university settings.

It is commonly acknowledged by a number of researchers that agile must be adapted to suit the university context [6, 24]. Some of the adaptations while teaching agile methods and practices in academic settings include variations to sprint lengths, stand-up meetings, sprint meetings, and use of online digital tools and boards [1, 5, 9, 10, 14, 24, 25]. Others include variations to roles, i.e. lecturer, tutor or external, team member playing the role of customer, scrum masters rotating between team members, others had experienced coaches as their scrum masters.

Building on this growing body of knowledge, our study presents evidence of the types of constraints faced by students when applying agile on quasi-real-world projects, working in close collaboration with industry 'customers' within university contexts; and the adaptations made by student teams to agile practices to work around the constraints. Additionally, perceived effectiveness of these adaptations is discussed leading to an understanding of the impact they made on students' learning.

3. Teaching Approach

3.1. Course Description

The course SoftEng761 Agile and Lean Software Development provides theoretical agile foundations to students and exposes them to hands-on software development. The key components involve iterative and incremental software development, self-organizing teamwork, project management through project work, and finally, invokes critical research and reflection through comparison across theory, project experience in the course, and industrial practice.

The course was designed and launched by Dr. Hoda in 2013 and has had over 250 students deliver 36 projects using agile methods over the last five years. It is taken up by final year Bachelor of Engineering (Honours), Master of Engineering studies students (pre-dominantly specializing in Software Engineering and some in Computer Systems Engineering) and Masters of Information Technology students. The last couple of years have had around 75 students in the class, self-forming teams 8-10 teams of 7-8 for the project. All students are expected to have strong object-oriented programming and teamwork skills which help them to choose a project matching their skillset. In terms of workload, the students are expected to devote approximately 10 hours per week, for a total of 12 weeks of the semester and participate in all assessments, test, project, and essay.

3.2. Course Plan and Structure

Figure 1 shows the course structure, assessment breakdown and timeline. The course follows a three-tiered learning approach as elaborated below:

3.2.1. Part 1: Theory

In the first three weeks, students learn the basics of Agile, Scrum, XP, Lean and Kanban methods through lectures and materials, which helps them to adopt an agile mindset. Many hands-on simulation games and exercises are used to impart theory in addition to materials on slides. Students are tested on these fundamental concepts in a test worth 25% in week 3.

3.2.2. Part 2: Team Project

Students self-form teams for the project. In 2016, 8 teams were formed comprising 7-8 students each while in 2017, 10 teams were formed comprising 7-8 students each. Teams work in close collaboration with industry partners 'customers' to apply their technical skills acquired from previous years and agile practices learned in Part 1 of this course in the form of a team project. They develop a proof-of-concept software based on the needs of their industry partner. They simulate a "quasi-real-world" agile software environment escorted by teamwork, project management, and software process experience to implement the learned theoretical concepts. While the projects are industry-based, we refer to them as quasi-real-world because of their university context (e.g. location, schedule, facilities, lack of financial concerns or pressures of a real job) which differs from a full industry experience. However, having industry involved lends some of the real-world context (e.g. managing customer expectations, requirements engineering, regular customer collaboration, professionalism, etc.) and prepares them to encounter similar scenarios in the industry after they graduate. Project teams are free to customize the Scrum process to suit their context and preferences. However, they are expected to follow some basic Scrum practices including weekly sprints, sprint planning meetings, daily/frequent stand-up meetings, composing user stories and associated acceptance criteria, creating and maintaining product and sprint backlogs sprint review and retrospective meetings. The projects count for
50% of the overall assessment.

Local industry partners serve as the customers for these projects. We put out a yearly call for proposals to our local industry contacts in May, with proposals due in June. We typically receive about 15 proposals per year. An example of a past project is CLVR, an app that uses Artificial Intelligence to automate Behavioural Interviews done during job recruitment, and provides instant feedback on a job candidate's personality and emotional intelligence. Based on the

work done by the students, CLVR was later named a finalist in the BNZ Start-Up Alley competition[2].

The course staff reviews the proposals received from the industry partners to ensure the projects are suitable and provide the right level of scope to be completed during the course while still challenging the student teams. In some cases, we will ask for additional details or ask for minor modifications to the proposed projects. Once this is complete, we provide a list of all available projects to the students, and the teams rank the projects in order of their preference. We aim to give each team one of their top-ranked projects to ensure the teams are invested and motivated and also ensure the teams have the needed skills for their project. Since we usually have more projects than teams, the projects with the lowest ranks are not allocated.

Once the students are assigned a project, they work closely with the local industry representative, who plays the role of Product Owner. The industry partners provide, clarify, and prioritize requirements, review demos the software, provide feedback through acceptance tests, and contribute to final project and team evaluations. The teams typically meet with their industry partner once a week. Over the course of the project, i.e. 8 weeks, the teams deliver a project plan, design documents, and two formally assessed releases of the software. For each of these deliverables, the teams do brief presentations to the class.

3.2.3. Part 3: Individual Research

In this part, students apply critical analysis and reflection by comparing agile theory, practice (i.e. project experiences) and related research literature on various agile and lean topics. They produce an individual essay worth 25% on the given research topic, e.g. *common challenges of practicing Agile and Lean software development (2016) and contextualizing Agile in the university context (2017).*

[2] http://www.webstock.org.nz/bnz-start-alley-17-finalists/

4. Research Methodology

The goal of this study is to investigate how university students contextualize and adapt agile practices during a quasi-real-world project of 8 weeks to achieve project outcomes. Our data collection occurred through student surveys and analysis of student essays. The study identifies the constraints students faced and reports resulting deviations from standard ways to apply and follow agile methods in university settings. Not all adaptations proved useful to the teams, but most were seen to work well in university contexts.

The research questions driving this study are listed below:

RQ1: What are the most common constraints faced by students while practicing agile in a university course?

RQ2: Which agile practices do students choose to follow and how were those practices adapted to fit the university context?

RQ3: Which adapted practices are perceived to be beneficial and effective in terms of outcomes and which are not?

4.1. Data Collection and Analysis

This research is based on data collected from two years' running of SoftEng761, offered in second semester of 2016 (students = 60) and 2017 (students = 76). This includes 291 individual responses to 4 reflection surveys contributed by 60 unique students in 2016; and 75 individual reflection surveys and another 75 individual essays contributed by 75 unique students in 2017 (one student did not submit the survey and essay). The surveys were distributed on Qualtrics and responses saved into excel sheets. These individual surveys were used to map data to team level results. If a majority of the team members (>=75%) selected individual responses such as constraint or a practice followed, then it was perceived as team level constraint or practice.

Some of the survey questions included:

- *Based on your experience in this course, what are the challenges of agile and Lean software development?*

- *Please rate your overall level of satisfaction with your customer on a scale of 1-10; where 1 is well below standard and 10 is excellent?*
- *How did your team 'tweak' standard agile practices to fit the university context?*

The essays were between 4-6 pages long and were submitted as PDFs. In these essays, students were asked to describe, in detail, the challenges they faced associated with applying agile practices in a university context, the adaptations they made due to these challenges, and how well those adaptations worked in their experience.

The data collected was saved and analysed in NVivo, a data analysis software, using open coding and constant comparison procedures of Grounded Theory data analysis [25, 26]. During the analysis process, categories such as constraints and adaptations emerged from the coding and constant comparison enabling the researchers to identify the common patterns and dissimilarities.

Table 1 lists all the agile practices the teams used (both as standard and adapted) while implementing their projects, where the 8 teams from 2016 are referred to as T1-T8 and the 10 teams from 2017 are referred to as T9-T18. It is evident that all the teams followed some agile practices (shown in row 3) such as scrum board, daily stand-ups, sprint planning, review, retrospectives and using scrum masters as recommended in the course. Of other agile practices, 5 teams used continuous integration while only one team, T2, applied work in progress (WIP) limit concept from Kanban.

Table 1 Agile Practices used by the student teams in SoftEng761 (SB: scrum board, DS: daily stand-up, SA: self-assignment, SP: sprint planning, RM: review meeting, RP: release planning, Re: retrospective, CF: cross-functionality, SM: scrum master, PO: product owner, Est: estimation, PP: pair programming, Ref: refactoring, CCO: collective code ownership, CI: continuous integration, WIP: work-in-progress limit, DoD: definition of done)

Team #	Scrum Practices								Scrum Roles		XP Practices						
	SB	DS	SA	SP	RM	RP	Re	CF	SM	PO	Est	PP	Ref	CCO	CI	WIP	DoD
All teams	✓	✓		✓	✓		✓		✓	✓							
T1	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓		✓
T2	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓				✓	
T3	✓	✓		✓	✓		✓		✓	✓	✓						
T4	✓	✓		✓	✓		✓		✓	✓							
T5	✓	✓		✓	✓		✓		✓	✓		✓					
T6	✓	✓	✓	✓	✓		✓		✓	✓		✓					
T7	✓	✓	✓	✓	✓		✓		✓	✓	✓	✓			✓		
T8	✓	✓		✓	✓		✓		✓	✓		✓		✓			✓
T9	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
T10	✓	✓	✓	✓	✓		✓		✓	✓		✓		✓			✓
T11	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓	✓			
T12	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓		
T13	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓		✓			
T14	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓			✓
T15	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
T16	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	✓			
T17	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓
T18	✓	✓	✓	✓	✓		✓		✓	✓		✓			✓		✓

5. Findings

In this section, we describe the constraints faced by students in practicing agile methods in a university context, the adaptations made to work around the constraints, and their perceived effectiveness.

5.1. Constraints

Implementing agile practices in a university course is an important way of learning and involves collaboration, communication, dedication and motivation of the students, staff and the customers. On the other hand, it brings some constraints that hinder the students from following these practices by the book.

The constraints faced by the students during the course project of SoftEng761 were collected through an open-ended survey question from the students in the 2016 course. In 2017, we provided the list of constraints identified in 2016 and asked the students to select each constraint they faced on their project. There was also an option for the 2017 students to write in additional constraints they faced. However, the write-in responses from the 2017 students did not result in any new categories. The constraints identified by the students in both years were:

5.1.1. Schedules Constraints

Some of the constraints related to schedules. For example, several students from 12 teams (80%) [T1-T3, T9-T14, T16- T18] reported difficulty in setting up a common time for all team members to work together. Often this was because of their varying and clashing university lecture timetables [reported by all teams T1-T18] and conflicting due dates of deliverables in other courses such as assignments, tests and projects [raised by 15 teams T4-T18]. This constraint is well captured by the following comment from one of the students in T9.

"It was extremely challenging to organize times for the entire team to meet for meetings and co-located development [sessions]. This was due to the team having different class timetables and commitments."

In addition to these normal scheduling conflicts, the University schedule itself fluctuates with a mid-semester break and other University commitments throughout the semester. For example, one student said:

"On top of these regularly occurring limitations, there were large University events beyond our control that disrupted project work. These included a two-week long study break, and 'Systems Week' – a week-long project [required for all 4th year students] which prevented students from doing other coursework."

5.1.2. Team Communication Issue

Teams faced difficulties while communicating with each other [reported by three teams (16%) T3, T8, T18]. One team T4 specified that working remotely on separate devices led to similar communication issues. Less visibility for peer progress was identified as a limitation by another team T2. Overall, communication seemed challenging for the teams. Some quotes from the students that reported these difficulties are:

"...students who may lack the required communication skills for group projects."

"...lacked transparency between each team member, and didn't know how much progress they've made on a task."

".... happened to be international students, leading to difficulties communicating due to English incompetence"

5.1.3. Customer Related Issues

Collaborating with customers and product owners was reported as challenging by several students in 10 teams (56%) due to limited availability of client due to busy schedule and business commitments [T2, T9-T13, T15-T18], unavailability of customer during planned meeting time was brought up by 5 teams (28%) [T1, T2, T15-T17], difficulty in finding meeting time suiting the product owner/customer and students was reported by a majority of the students in 8 teams (44%) [T2, T9, T12- T14, T16-T18], difficulty in prioritizing due to

unrealistic customer demands was noted by 3 teams [T12-T13, T17] and distant location of product owner was highlighted by one of the teams [T7] . Some quotes from the students that demonstrate these difficulties are:

"The industry client we [T9] were working with had commitments he needed to keep at his full-time job."

"We struggled as a team [T10] to find a time we could coordinate with the product owner to meet for sprint planning, retrospectives and reviews."

5.1.4. Lack of Dedication

Some of the constraints related to dedication of team members. For example, limited dedication of team members due to other courses, or otherwise was reported by half of the teams [T3, T9-T14, T17-T18] and unavailability of some team members during planned project time was also mentioned by 50% of the teams [T8-T14, T16, T18]. For example, as noted by member of team T11:

"Some students were not motivated nor proactive, and produced low quality work (which sometimes wasn't tested and didn't work)."

Another student noted:

"Not everyone turned up on time or even turned up to the group chat without any prior notice."

5.1.5. Personal Commitments

Personal commitments outside university was reported as a constraint by 11 teams (61%) [T3, T5-T6, T9-T14, T17- T18]. These commitments were mainly related to professional activities of working team members. For example, one member of team T10 stated:

"Some team members had commitments outside of University, such as work or club activities."

5.1.6. Technical Constraints

Difficulty in estimation due to unclear scope was reported by 9 teams (50%) [T4, T9-T13, T15-T17]. Similarly two teams specified difference in technical skill level [T8] and difficulty in setting a development pace for the team members [T5] as some of the technical constraints faced by the teams. For example, one student noted that:
".. at university, the technically competency and programming skills of students vary. Some students are more proficient at coding than others."

To better understand the frequency of these constraints, the students in 2017 were given these constraints as a closed list to select the ones they faced while practicing agile in their projects. The top five constraints chosen by the students in 2017 are the unavailability of some team members during planned project time [67%], difficulty in setting up common time for all team members to work together [67%], conflicting due dates of deliverables in other courses like assignments, tests and projects [61.5%], limited dedication of team members due to other courses [52.6%] and different and clashing university lecture timetables between team members [51.3%].

5.2. Adaptations to Agile Practices

This section describes how the teams follow, adapt and modify agile practices during the execution of the team project within the university context. Some of these agile practices were adapted to overcome coordination related hurdles, both within the team and with the product owner, and improve the engagement, empowerment and culture of the team. These adaptations are presented in Table 2 and elaborated below.

Scrum Practices

The students described adaptations to daily stand-ups, scrum boards, sprint length and schedule and others.

5.2.1. Daily Stand-up

In theory, Daily Stand-ups (DS) takes place every day, face-to-face, to report updates on work done. It is one of the most common practices followed by agile teams in the industry. In university too, it was the most commonly used practice but was adapted to fit into the university context. Because of varying student schedules during the day and throughout the week, most teams preferred a virtual collaborative

environment (e.g. Slack, WeChat, Facebook Messenger and email) over traditional stand-up meetings to discuss the team's accomplished tasks, focus areas, and blockers. A few teams were doing a mix of in-person and virtual daily stand-ups. The frequency of the stand-ups was also reduced in accordance with the schedule of the team members. Some teams opted weekly when the group planned to meet while others some performed it on alternate days, biweekly and triweekly. A few teams used bots (e.g. Chatbot, Slackbot) which were configured to collect statuses from individuals through private messages at a pre-set time and triggered reminders through notifications.

5.2.2. Scrum Board

All the teams opted to use online tools (Trello, Jira, GitHub using ZenHub) as scrum boards which could be accessed remotely at any time and enabled both the students and customers to log activities and track the progress easily. Given there was no dedicated workspace for the teams, conventional physical scrum boards were infeasible.

5.2.3. Sprint Length and Weight

Agile teams in the industry are typically seen to maintain a consistent sprint length. However, in the university course, the teams adapted the sprint lengths based on their workload. Some had weekly sprints in the beginning but then during other busy times, such as inter-semester break, exams and other commitments, these were changed to biweekly. Similarly, for times when there was less load in other courses, the teams switched to shorter sprints in the agile course. For example, one of the teams defined their development period over weekly sprints as a 6-hour work week and 2-hours per week for sprint review/planning/ retrospective.

Sprint Meetings

The majority of the teams preferred having one physical meeting per week with additional team meetings occurring through an online collaborative platform (e.g. Facebook Messenger, GitHub, Slack and email).

5.2.4. Release Planning

As a standard scrum practice, release planning is done as an independent session for planning every release. Due to the relatively small scale and short duration of the projects, the teams who performed release planning held one or two sessions at the beginning of the project, which included planning for all anticipated releases such as the prototype and final product (see Fig.1). One of the teams combined release and sprint planning into one meeting and planned out tasks for all releases and sprints at the start of the project.

5.2.5. Sprint Planning

Sprint planning sessions are customarily held at the beginning of the sprint to define the sprint backlog. In SoftEng761, sprint planning sessions were mostly conducted as face-to-face meetings by the teams. One of the most common modifications to suit the university context was combining sprint planning with the sprint review into a single weekly session to accommodate to the team's and product owner's limited availability, which is quite unlike in standard practice.

5.2.6. Sprint Review

During a standard sprint review, the team usually presents the sprint work to the product owner for their feedback. In the university context, the reviews were adapted to replicate the experience virtually through a video-conferencing solution (e.g. Skype, Zoom). However, sometimes the teams were unable to demonstrate their weekly progress to the industry partner due to technical issues.

5.2.7. Retrospective

Generally, retrospectives are run after each sprint to suggest process improvements for the following sprint, but many teams reported holding it before the sprint planning and review to suit the availability of the industry partner. Another team held retrospective meetings before sprint review meetings as it was convenient for the team members. Traditionally, the Product Owner is not included in the retrospective; however, due to the product owners' vast agile experience, one of the teams found it very useful to include their product owner.

5.2.8. Cross-Functionality

In theory, scrum supports cross-functionality within teams where every team member is open to

take up any work irrespective of their skillset. Maintaining cross-functionality in a university context is challenging due to the limited project duration. One of the strategies followed by the teams was splitting teams into sub-teams (e.g. mobile and web app teams or frontend and backend teams) which helped to achieve cross functionality to some extent within sub-teams. A few teams preferred specialization, letting people work on areas where they already had expertise instead of promoting cross-functionality within teams.

Scrum Roles

The students reported adaptations around traditional scrum roles i.e. Scrum Master and Product Owner as elaborated below.

5.2.9. Scrum Master

The role of the Scrum Master (SM) is typically played by a dedicated resource in industrial contexts. In SoftEng761, teams were recommended to adopt a rotating Scrum Master role to allow each team member to gain leadership experience. Some of the teams practiced this throughout the project. One of the teams tailored this further by introducing an overseer (group leader), a person with advanced leadership skills, to assist the rotating scrum masters as they gained experience. Some teams experimented with having two scrum masters for each sprint to reduce pressure on a single person and share the responsibilities, e.g. one of the teams had a dedicated SM with a rotating secondary SM in parallel to allow learning opportunities amongst other team members. One of the teams dropped the SM role toward the end. This way team members contributed where they could, making easy for the customer to communicate equally with all.

5.2.10. Product Owner

In typical agile projects, a product owner is a dedicated role in agile team responsible for defining and prioritizing the backlog based on the customer's need. For the majority of the teams, a representative of the industry partner served as the product owner. One of the teams adapted this by introducing collective product ownership toward the end of the project as their industry partner was unable to be involved to the required extent due to professional commitments. Another team had two industry partners as their product owners.

Table 2 Constraints, Agile Adaptations and their Perceived Effectiveness

Standard Practices	Constraints	Main Adaptations	Perceived Effectiveness	
			Positives (+)	**Negatives (-)**
Daily Stand-Ups (DS)		Reduced frequency	(+) 2-3 times a week (before lecture and PO meeting with the PO) seen as more valuable than virtual	(-) obstacles reporting delayed in bi-weekly DS
	Schedule, Team Communication, Personal	Using virtual tools, including bots	(+) convenient and suited team members (+) Slack bot worked better than manual slack updates	(-) missing emotional information and interaction cues (-) missing/delayed responses (-) texting fatigue (-) time box overruns (-) attrition over time (-) messy log (-) untimely reminders (-) notifications disabled
Scrum Board	Schedule, Team Communication	Digital Scrum board	(+) tidier than a physical scrum board (+) phone notifications useful (+) constant availability and sharing	(-) updating a challenge (-) too many updates led to messy boards
Sprint Length	Schedule	Variable/short sprint length	(+) improved workload balance (+) more frequent customer collaboration due to short sprints	(-) reduced project velocity
		Working 1-2 days per week in collocated sessions	(+) increased focus (+) better communication (+) improved mutual understanding of schedule (+) improved workload balance	
Sprint Meetings	Team Communication	Combining /sequencing meetings	(+) combining/sequencing meetings proved efficient for customer collaboration	(-) exhaustion, loss of focus (-) losing essence of each meeting practice (-) underestimated user stories in early sprints due to combined release/sprint planning
		Changing sequence	(+) moving retrospective before review followed by sprint planning for next sprint suited PO and the team	
Retrospective	Schedule	Presence of PO	(+) PO in first few retrospectives provides direction	

Cross-functionality	Technical	Subdividing into sub teams (backend/frontend)	(+) better focus on one area	(-) lacking inter group communication led to some confusion and delays
		Decreased cross-functionality	(+) specialization increased productivity and efficiency of team members in short term	
Scrum Master (SM)	Schedule, Team communication	Dedicated SM	(+) single point of contact (+) experienced SM effective in running retrospectives & maintaining sustainable pace	
		Rotating SM	(+) everyone experienced role	(-) customer struggling to keep track (-) some students not suited as SM (-) reduced opportunity to consolidate learning
		No SM	(+) one of the teams dropped the SM role toward the end	
		Overseer with SM	(+) having overseer with SM worked faster and efficiently	
Product Owners (PO)	Customer related issues	Collective product ownership		(-) Collectively sharing product ownership added more responsibilities so having dedicated PO would have been better for the team to focus on work
		Multiple POs		(-) reduced velocity due to private reconciliations of (two) PO disagreements
		Condensed collaboration	(+) short (2 minute) updates appreciated by PO over full SPM sprint planning meetings (+) combined meetings	
Pair Programming	Technical, Schedule	Remote Pair Programming	(+) knowledge transfer (+) collaborative learning (+) remote pair programming worked via screen sharing (+) combined with code review for improved code quality	(-) scheduling common time (-) confusion in remote pair programming sessions sometimes
Customer Communication	Customer related issues	Online communication	(+) easy approach to customer	(-) difficulties communicating with customer during sprints due to technical issues (-) Emailed tasks not accepted well by team

XP Practices

5.2.11. Pair Programming

As an XP practice, two programmers sit side-by-side at a single machine to write code; one writes while the other reviews it simultaneously. Many teams were seen to practice pair programming with varying consistency. Some did this on an as-needed basis, some only very occasionally, and some had dedicated time slots every week for this. A few teams performed them remotely through skype screen sharing since it was difficult to arrange a common time suiting all team members. It was seen that usually the same people, and often friends, would pair up due to similar schedules and common understanding.

5.3. Perceived Effectiveness of Adaptations

As described in section 4, we asked the teams. Which tweaked [adapted] practices did and did not prove beneficial and effective in terms of outcomes and why? In response to this, they shared their experiences with the adapted practices and we then synthesized the responses to determine the effectiveness of these adaptations as summarised in Table 2 and few elaborated below. It is evident from the results that the adaptations had both positive and negative effects.

To address the problem of conflicting schedules and physically meeting each other every day, every team adapted daily stand-ups someway to fit the university context. Physical stand-ups were more structured, expressive and useful for team communication. However, virtual stand-ups replaced physical stand-ups most of the time but some teams had difficulties around time-boxing them, setting up a time suiting everyone in the team, less number of people showing up on time, etc. Additionally, teams faced some technical issues (e.g. sometimes messages were lost in the channel or poor voice quality when video chatting). The strategy that was most commonly reported to work well was having at least 1-2 physical stand-ups (typically held before or after the weekly lectures or customer meetings) accompanied by a few virtual meetings through an online communication channel such as Slack, Facebook messenger, WeChat.

Most of the teams started with rotating the scrum master role among team members every sprint to allow all team members to experience the role. After a few sprints, many teams found it somewhat confusing and counterproductive for the team due to the following reasons:

- a single sprint was too short for the new scrum master to understand and gain confidence in the role,
- passive or introverted team members found it challenging to lead the team,
- added communication confusion for the industry partners as they were not sure which person to contact.

Adaptations like using a digital scrum board and having a variable sprint length were perceived as valuable as they helped the teams overcome some of the university constraints such as differences in student schedules. Digital scrum board such as Trello, Jira, Visual Studio Team Services (VSTS) was adopted by all the teams and reported to benefit in the university context if regularly updated by all team members. The main benefit was that the team members and industry partners could access it remotely at any time, still few team members reported keeping the digital board up to date cumbersome.

Using online communication for meetings did not prove beneficial at all times and email was reported as challenging for the teams (e.g. missing details). There were instances when technical issues led to difficulties hearing and responding to the customer or to each other leading to confusions. Another constraint reported though occasionally was that the teams were unable to demonstrate work to the customer due to limitations of the webcam when done virtually.

5.4. Relationship Between Constraints & Adaptations and Outcomes

Applying agile to university projects exhibits constraints such as balancing workload with other courses, difficulty in setting up a common time to work together, conflicting due dates of deliverables in

other courses, and limited dedication of team members due to demands from other courses. This leads to students adapting agile practices to better suit the university context.

As an example, it was impossible for the students to meet every morning to perform a stand-up meeting. Also, due to other courses classes, tests and assignments, it was unlikely for team members to be able to show daily progress. Therefore, the teams adapted the practice in many different ways by performing standups physically but less frequently, daily but virtually, physically once weekly accompanied with bi/tri-weekly virtually meetings, and only daily virtual meetings. These adaptations had both positive and negative effects, some of which are shown in Figure 2 (and also summarized in Table 2 earlier).

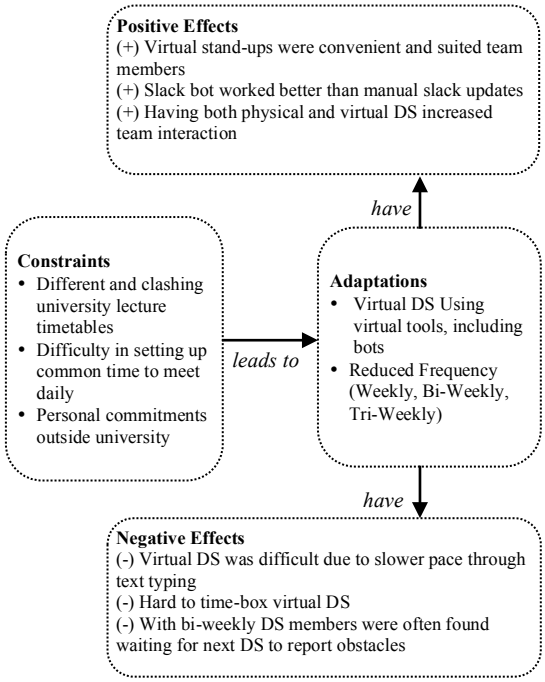

Figure 2. Relationship between Constraints, Adaptations and their Effects for Daily Stand-ups (DS)

6. Discussion

Applying agile practices to university contexts is not a straightforward process and requires modifications to standard ways to work around the constraints imposed by university settings. A good number of agile practices were adapted to suit university settings; but not all were found effective to students in learning agile practices. The adaptations perceived to be most effective were virtual stand-ups combined with 1-2 physical ones per week, digital scrum boards, and variable sprint lengths to suit the university's schedule.

However, there are certain limitations to what extent students can modify practices in a university agile project. The short-term nature of university projects often causes students to focus on different aspects than would be typical in a real-world project. For example, task estimates typically improve over time as a team gains experience. However, many teams noted their early estimates were very inaccurate; and, thus, they stopped doing estimations altogether in later sprints. Further, many teams admitted that they optimized finishing the project and obtaining high marks over learning new skills. For example, team members were often reluctant to learn new technologies when another team member had expertise in that area and were mostly not open to practicing cross-functionality or collective code ownership. Thus, students may not have invested as much time as necessary to really learn the agile practices, rather they prioritized project completion. This was perhaps particularly pronounced due to the short project duration and marks incentives that apply in a university setting.

The scale of the modifications of agile practices varied. Some practices introduced major tweaks (e.g. replacing physical stand-ups by virtual stand-ups or eliminating the role of scrum master). Others were minor tweaks or slight variations to standard practices such as less frequent physical standups, or shorter sprints during peak workload.

Often, teams experimented by modifying a few practices at the beginning of the project, but with time they realized it did not work well and so adapted it another way, thus being agile with their practices. For example, one of the teams initially started with daily virtual standup meetings, but they found that due to other commitments they were not working on the project every day, so the meetings were not always productive or necessary. They later reduced the number of standups and that worked well for the team as each team member had something useful and new to report at each standup. Similarly, another team started off with a rotating

scrum master, but they found it was hard to keep track of who was in charge for the current sprint for both the customer and the team. They also reported that often the new person in charge was unaware of what was expected as it was their first time being in such a role and that by the time they got up to speed it was time to rotate a new scrum master. After a couple of sprints, the team decided to assign a dedicated scrum master, which fit well in their context. Everyone knew who to contact when there were any troubles, and the customer preferred having one consistent point of contact.

Most of the adaptations were around scrum practices. While the students followed several XP practices, very few deviated from standard ways in relation to these practices. Some of the adaptations were pair programming through screen sharing, and pair programming between friends. One clear point that emerges here is that Scrum practices were seen to be adapted more in the university context than Extreme Programming (XP) practices. This could be because Scrum practices generally involve the whole team and/or the customer. Since schedule conflicts and limited customer availability were some of the biggest constraints, these collaborative Scrum practices were most affected and needed adaptations. Other reasons could be that XP practices may not require any drastic changes to fit in a university context, so teams were able to adhere fairly closely to standard practices or due to a lack of experience with XP practices, the teams might have felt it was risky to adapt them as doing so incorrectly could have a negative impact on team performance and/or product quality. Future research should investigate this in greater depth.

6.1. Comparison to Related Work

Prior work on contextualizing agile for university settings highlighted how students adapted agile to fit their needs. Some of these, e.g. [2], used an electronic task board and dedicated working hours every week on projects. Based on their experiences, students suggested intensive working week over dedicated working hours and using physical task boards over electronic boards due to performance issues. In another study, stand-up meetings were conducted during class times [23], which slightly differs from the adaptations we identified. Our teams utilized lab times and space for group meetings, but stand-ups did not take place during class time as most of those were utilized for theory and team presentations.

Some similarities are reported around the role of scrum master in another study. In a three-sprint scrum, students adapted the scrum master's role as either a rotating scrum master, or with two members sharing the role or one permanent scrum master [1]. This is quite consistent with how our teams adopted this role in the university context. Similarly, some previous studies reported that the customer [2, 13] or a team member [1, 9] acted as the product owner. In our case, industry partners were mostly the product owners, with the exception of one team who collectively played the role of product owner.

It is evident from related studies that sprint length is typically kept fixed for university projects [2, 9, 13]; however, Werner et al. [1] described a university project with teams adapting different sprint lengths, from 1 to 4 weeks. All our teams started with one-week sprints. As the project progressed, many teams increased the sprint length to two/three weeks for a limited time. This adaptation worked to balance the workload and ensure the completion of planned tasks.

6.2. Implications

Based on their experiences and lessons learned, the students made recommendations for the course instructors and future students to keep into account while teaching and implementing agile practices in similar courses through projects.

6.2.1. Recommendations for Students

- 'Daily' stand-ups should be done face-to-face but less frequently, starting with at least 2 stand-ups weekly. However, the frequency of can be adjusted to suit the needs. Combining physical and online stand-ups can serve as a good compromise.
- Sprint length can remain fixed most of the time but consider varying the sprint length occasionally to balance student workloads during busy assessment and test periods at the university.
- Having a combined sprint review and planning session can help make the most of the limited customer availability.
- Utilizing online tools to simplify communication within teams such as team emails, messengers, chat channels, digital scrum boards, virtual pair programming tools, bots to update direct from chats to digital task boards, and bots to send daily reminders for stand-ups are by and large useful.
- Sprint planning and retrospectives should be done in person where possible, as immediate feedback from team members can be extremely useful.
- Having one stable scrum master and others rotating can benefit the teams and members individually.

6.2.2. Recommendations for Educators

- Where resources are available, provide teams with experienced tutors or make masters students the Scrum Master for initial sprints if not throughout the project.
- Adequate support should be given to the Scrum Master by the team. They may be given a lighter developmental role to accommodate for management efforts.
- When sourcing industry partners, preference should be given to the ones located close to the University and having enough collaboration time available.
- Where possible, teams should be provided a dedicated place for them to arrange collocated code sessions and have their physical Scrum boards.
- Educate teams on best practices for virtual communication.
- Allocating time for daily-standups during classes and lab sessions can prove to be useful.
- Allocate a training period before the project begins where students can practice techniques such as Pair Programming or Test Driven Development so they do not have to experiment during the project.

7. Conclusion

It has long been acknowledged that learning agile is best done through practical hands-on projects. Yet, university projects cannot fully replicate real-world scenarios. Students face many constraints such as difficulty in setting up common time for all team members to work together, unavailability of some team members during planned project time, limited availability of customer due to busy schedule. This article reports how students tailor standard agile practices to suit the university context and mitigate these constraints. The most common adaptations were daily stand-ups with reduced frequency, combining and sequencing sprint meetings, and rotating scrum master from team. We found that many of these adaptations helped the students to overcome the hurdles they faced while some were not as effective as others. In addition to the presenting the common constraints and adaptations, we also provide a list of recommendations for both students and educators of future course teaching agile software development practices for effectively adapting agile in university contexts.

Acknowledgments

We would like extend our thanks to all the students of SOFTENG761 for sharing their experiences.

References

1. Werner, L., D. Arcamone, and B. Ross, *Using Scrum in a quarter-length undergraduate software engineering course.* Journal of Computing Sciences in Colleges, 2012. **27**(4): p. 140-150.
2. Kropp, M. and A. Meier, *Teaching Agile Software Development at University Level: Values.* Management, and Craftsmanship, CSEE&T, 2013.
3. Goldman, A., et al., *Being extreme in the classroom: Experiences teaching XP.* Journal of the Brazilian Computer Society, 2004. **10**(2): p. 5-21.
4. Muller, M.M. and W.F. Tichy. *Case study: extreme programming in a university environment*. in *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*. 2001. IEEE.
5. Schneider, J.-G. and R. Vasa. *Agile practices in software development-experiences from student projects*. in *Software Engineering Conference, 2006. Australian*. 2006. IEEE.
6. Fenwick, J.B. *Adapting xp to an academic environment by phasing-in practices*. in *XP/Agile Universe*. 2003. Springer.
7. Hanks, B. *Becoming agile using service learning in the software engineering course*. in *Agile Conference (AGILE), 2007*. 2007. IEEE.
8. Scharf, A. and A. Koch. *Scrum in a software engineering course: An in-depth praxis report*. in *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. 2013. IEEE.
9. Matthies, C., et al. *How Surveys, Tutors and Software Help to Assess Scrum Adoption in a Classroom Software Engineering Project*. in *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*. 2016. IEEE.
10. Lu, B. and T. DeClue, *Teaching agile methodology in a software engineering capstone course.* Journal of Computing Sciences in Colleges, 2011. **26**(5): p. 293-299.
11. Shukla, A. and L. Williams. *Adapting extreme programming for a core software engineering course*. in *Software Engineering Education and Training, 2002.(CSEE&T 2002). Proceedings. 15th Conference on*. 2002. IEEE.
12. Hazzan, O. and Y. Dubinsky, *Why software engineering programs should teach agile software development.* ACM SIGSOFT Software Engineering Notes, 2007. **32**(2): p. 1-3.
13. Bruegge, B., M. Reiss, and J. Schiller. *Agile principles in academic education: A case study*. in *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*. 2009. IEEE.
14. Schneider, J.-G. and L. Johnston. *eXtreme Programming at universities: an educational perspective*. in *Proceedings of the 25th international conference on Software engineering*. 2003. IEEE Computer Society.
15. Bunse, C., R.L. Feldmann, and J. Dörr. *Agile methods in software engineering education*. in *International Conference on Extreme Programming and Agile Processes in Software Engineering*. 2004. Springer.
16. Wainer, M. *Adaptations for teaching software development with extreme programming: An experience report*. in *XP/Agile Universe*. 2003. Springer.
17. Steghöfer, J.-P., et al. *Teaching agile: addressing the conflict between project delivery and application of agile methods*. in *Proceedings of the 38th International Conference on Software Engineering Companion*. 2016. ACM.
18. Hedin, G., L. Bendix, and B. Magnusson, *Teaching software development using extreme programming*, in *Reflections on the Teaching of Programming*. 2008, Springer. p. 166-189.
19. Deveď zic, V., *Teaching agile software development: A case study.* IEEE Transactions on Education, 2011. **54**(2): p. 273-278.
20. Rico, D.F. and H.H. Sayani. *Use of agile methods in software engineering education*. in *Agile Conference, 2009. AGILE'09.* 2009. IEEE.
21. Anslow, C. and F. Maurer. *An experience report at teaching a group based agile software development project course*. in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 2015. ACM.
22. Campbell, J., et al. *Scrum and agile methods in software engineering courses*. in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 2016. ACM.
23. Nejmeh, B. and D.S. Weaver. *Leveraging scrum principles in collaborative, inter-disciplinary service-learning project courses*. in *Frontiers in Education Conference (FIE), 2014 IEEE*. 2014. IEEE.
24. Mahnic, V., *A capstone course on agile software development using Scrum.* IEEE Transactions on Education, 2012. **55**(1): p. 99-106.
25. Glaser, B.G., *Basics of grounded theory analysis: Emergence vs forcing*. 1992: Sociology Press.
26. Hoda, R., J. Noble, and S. Marshall, *Developing a grounded theory to explain the practices of self-organizing Agile teams.* Empirical Software Engineering, 2012. **17**(6): p. 609-639.