Reflections on Agile Software Development: A Conversion Master Case study

Ali Ahmed
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
ali.ahmed@vuw.ac.nz

Karsten Lundqvist
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Karsten.Lundqvist@vuw.ac.nz

Jennifer Ferreira
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
jennifer@ecs.vuw.ac.nz

Craig Watterson
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Craig.Watterson@vuw.ac.nz

***Abstract*—This research to practice paper presents a reflective study showing the observations of the teaching teams and the practices during industrial projects for the Master of Software Development (MSwDev) program at Victoria University of Wellington (VUW), New Zealand. It is a conversion master program that helps the students to acquire specific skills to be ready for the transition to the software development industry, including the ability to work in an Agile team. Software development projects supported by industrial partners are an excellent means to evaluate the technical capacity of the students and the required skills. This reflective paper describes the evolution of an industry-sourced problem-based agile student project, which over three successive years, has included students from three different cohorts (User Experience (UX), Business Analysts (BA), and Software Developers (SwDev)). This year, a collaboration between different cohorts was not possible. Thus only software development students formed the agile teams.**

This paper describes and reflects on the teaching staff's useful and problematic observations over the last twelve months using only the SwDev cohort and correlates that to the previous reflections when three different cohorts formed an Agile team. Those reflections are described, and several recommendations are drawn from the experiences. The authors also discuss the lessons learnt and the effect of COVID-19 on the teams' performance.

***Index Terms*—Software Development, Teaching Agile, Tertiary Education, Industrial Projects, Conversion Masters, Teaching and learning practices, Scrum, Conversion Master Program**

I. Introduction

Teaching agile software development through capstone projects consolidates the theories and practices students need to learn for their future employment. According to [1], a "Capstone is a metaphor used to describe a final achievement that builds upon previous works and encapsulates them". A capstone project is usually a practical assessment that requires students to plan, implement, and evaluate a solution to a specific software problem. This is considered a problem-based learning strategy that utilises developing a piece of software application that meets specific functional and non-functional requirements [2]. Capstone projects have even proved useful for distance learning courses and degree programs [3], [4]. Capstone projects enable students to build a wide range of soft skills such as project management, teamwork, presentation, and communication [5], [6]. Team projects is recommended to ensure software development graduates have the quality and competitiveness required by the software development industry [7]. The ideal setting is to combine capstone projects with real-world software problems sourced through industrial partners. This maximises the benefits of the capstone projects allowing students to understand the software development process holistically.

Agile software development methodologies are the mainstream in the software industry [8], [9]. Agile methodologies support creating a Minimum Viable Product (MVP), where developing a basic set of features will promote rapid software development and increase the software product quality by incorporating the product owner efficiently in the development process. It is a challenge to integrate agile software development practices in a conventional lecturing theatre when it comes to academia [10]. However, researchers are paying more attention to integrating agile software development into university courses to augment the gap between universities and the software industry [11].

Teaching agile software development at a university level exposes the students to the current industry practices while emphasising the academic theories. Organising the software development process and dealing with divergent skill sets of software developers are challenging when teaching agile software development through lab courses. However, SCRUM helps teach Agile software development as it is easy to learn and widely adopted in the software industry. To utilise the SCRUM method efficiently, industrial partners need to be involved to ensure the process is practical and the final product matches the user's stories [12], [13].

This paper outlines the authors' observations and reflections on how a group of students from MSwDev program are

exposed to using agile software development practices to solve industry problems. The paper describes the overall components of the program and how it runs. The organisation of the paper is as follows:

1) Section II identifies recent trends in teaching Agile software development in tertiary education.
2) Section III discusses the MSwDev conversion Master's program used for this research paper.
3) Section IV demonstrates the authors' reflections.
4) Section V concludes the paper and provides recommendations by the authors.

II. Teaching Agile Software Development

Teaching Agile software development to students is a challenging task. Students must acquire skills essential to their development as software developers, such as communication, agility to requirement changes, etc. The challenge comes from the fact that the teaching is usually in a simulated environment where academics mimic real industrial projects and best practices. Observing students' behaviour during teaching agile methodologies has attracted attention in recent years as a way to evaluate the students' understanding of the mainstream software development practices in the software industry [11], [14]–[18].

Table I summarises several papers from 2018 to 2021, selected from IEEExplore, ACM Digital Library, ScienceDirect, Scopus, and Google Scholar, which investigate teaching Agile methods.

Table I shows three main categories for teaching students Agile software development. Firstly, capstone projects involve industrial partners acting as software 'clients'. Industrial partners are 'customers' with specific functional and non-functional requirements that must be implemented in the final product. This category acknowledges the importance of exposing the students to real-time problems in the software industry that span the entire life-cycle of a software project. Secondly, the use of gamification is a fun way of teaching Agile. Gamification is an approach that applies game design elements in a non-game context [47]–[49]. Gamification is an enabler in education, especially amongst younger generations. Gamification enables students to acquire the skills required for problem-solving [50]. It has been shown that gamification can be effective in education [51]. Thirdly, there is a trend in proposing supporting frameworks or architectures for teaching Agile software development methods and practices.

III. Program Structure

MSwDev conversion Master's program teaches students core programming concepts and skills they can apply to industry over an intensive 12-month period. It is worth to note it is a full-time program that runs from 10 am to 4 pm every day from Monday to Thursday. On most Fridays, the students have to attend industrial engagement seminars. The program is designed for people who want to work in the software development industry but do not already have an IT-related background. If a student has a degree in a non-IT subject, such as History, Chemistry or Accounting and wants to change careers, the program teaches them what they need to know to make the switch. If a student already has a strong IT background, MSwDev conversion Master's program is not the right choice for them.

As seen in Figure 1, the program is composed of the following papers:

1) SWEN 501: This course teaches core programming skills, concepts and techniques, including fundamental control structures, collection data structures and testing.
2) SWEN 502: The course addresses various software development skills and builds an understanding of technology and software engineering concepts and techniques. The course teaches teamwork skills, practical professional skills, and communication skills in the context of software development. Students will study various topics and then work on two projects that address multiple issues in software development and computer science. Topics include Databases, Networking, Cybersecurity, Human-Computer Interaction, Mobile Application development, and Web Applications. Each of these topics will take one week, and two projects will cover the material from the proceeding topics. Students will work on a variety of industry-relevant group projects which address a range of issues in software development and computer science.
3) SWEN 504: This course consists of a sequence of group projects, interleaved with teaching sessions, industrial seminars, industrial case studies, tutorials providing background to the projects, a review of the student work, and additional material to complement the project work. The group work and the teaching will be done in the same physical space, allowing for flexible timing of the more formal components of the course. The course includes weekly seminars from industry professionals on various topics.
4) SWEN 589: The course consists of a substantial project, working on a software research and development task. Generally, it will be done as placement in the industry but could, in some instances, be an industry-sourced (or industry-related) project at university. The project involves supervision by an academic and the industry employer and will involve formal and informal.

It is worth mentioning that the industrial partners are helpful in two components of the program:

1) During the industrial projects in SWEN 504, which span four weeks.
2) During the internship period, which spans three months.

This paper focused on the first point where the industrial partners collaborate with their groups via emails, daily stand-ups, zoom meetings, etc. The involvement of industrial partners varies from one group to another. This is due to industrial partners' tight time schedules and proximity to the teaching facilities [11]. However, the minimum required collaboration is maintained with a meeting at the beginning of the sprint to

TABLE I
STUDIES ON AGILE S/W DEV. TEACHING

Purpose	Paper	Finding
Teaching capstone projects mainly with industrial partnership	[11], [17]–[30]	1) Skills improvement including teamwork, project management, communication, problem-solving, accountability, time management, source control (using Git), testing, debugging, algorithm development, knowledge of agile software development, etc. 2) Industrial partner participation is important 3) Academic programme leaders and industry engagement managers shouldn't underestimate the coordination required 4) Positive student's attitude towards the course appreciating the engagement in team-based activities and practical experiments. 5) Challenges facing agile teams include limited availability of industrial partners, daily stand-ups with reduced frequency due to students' different availability, combining sprint meetings, and rotating scrum master from the team.
Gamification of Agile Teaching	[15], [24], [31]–[40]	1) Constant students' improvement in their learning and assessment, primarily if role-playing activities are used. 2) Realising the difference between agile and waterfall ways of software development and appreciating the good agile practices. 3) High student satisfaction. 4) Increasing students' engagement, motivation, and overall project quality.
A teaching-learning framework to support teamwork in agile software education	[16], [41]–[46]	improved team-working regarding learning and performance.

clarify the requirements and build the task backlog. At the end of the sprint, students present what they achieved. This process ensures that the SwDev students and industrial partners are on the same page.

This paper focused on the first point where the industrial partners collaborate with their groups via emails, daily stand-ups, Zoom meetings, etc. The involvement of those industrial partners varies from one group to another. This is due to industrial partners' tight time schedules and proximity to the teaching facilities [11]. However, the minimum required collaboration is maintained through a meeting at the beginning of the sprint to clarify the requirements and build the task backlog. At the end of the sprint, students present what they achieved. With such, the teaching team ensure that SwDev students and industrial partners are on the same page.

IV. REFLECTIVE ANALYSIS

The reflections listed in this section are based on the following various settings that span the last six years in the program. Some years ago, the Master of Software Development programme at VUW asked the students to work on industrially sourced software problems in two sets of three-week sprints. This was done in isolation of the business analysts, designers, testers, etc. In 2016-2017, SWEN 589 involved two iterations of industry-sourced projects developing software in Agile teams. The teams were working through two sprints and had the freedom to choose their preferred Agile development method. To accommodate the diversity of a typical Agile team, in 2017-2018, the software development students were joined part-time by user experience students. The user experience students were asked to develop wire-frames and early prototype designs for the software students.

While developing software prototypes, the user experience students would further expand the designs and test them on potential users. The students' collaboration was not fully integrated as an Agile development scenario, e.g. the user experience students were not fully involved in the teams and had other classes. However, the feedback from this experience was encouraging; thus, to enhance the collaboration, the group projects were extended in 2018-2019 to include more user experience and business analysis students. Instead of pre-designed software development projects, the industrial projects were sourced from the industry as business opportunities.

Furthermore, the projects were extended to seven weeks. The project started with the business analysis students leading the project, followed by user experience students, and then software developers. When leading the group, the students would work full-time on the project with time set aside by the other cohorts to contribute to the work throughout the week. This provided the flexibility to join the three programmes on this project while continuing the regular teaching on the other programmes. However, this administration required a massive effort from the academic staff, and the results were not as positive as hoped. One of the main issues was that the ownership of the projects was lost in the ownership changes between the cohorts, and in several groups, the cohorts blamed the other cohorts. In 2019-2020, the projects were scaled back to only include software development and user experience students. In 2020-2021, the business analysis students joined the other cohorts again. This time all three cohorts were fully involved

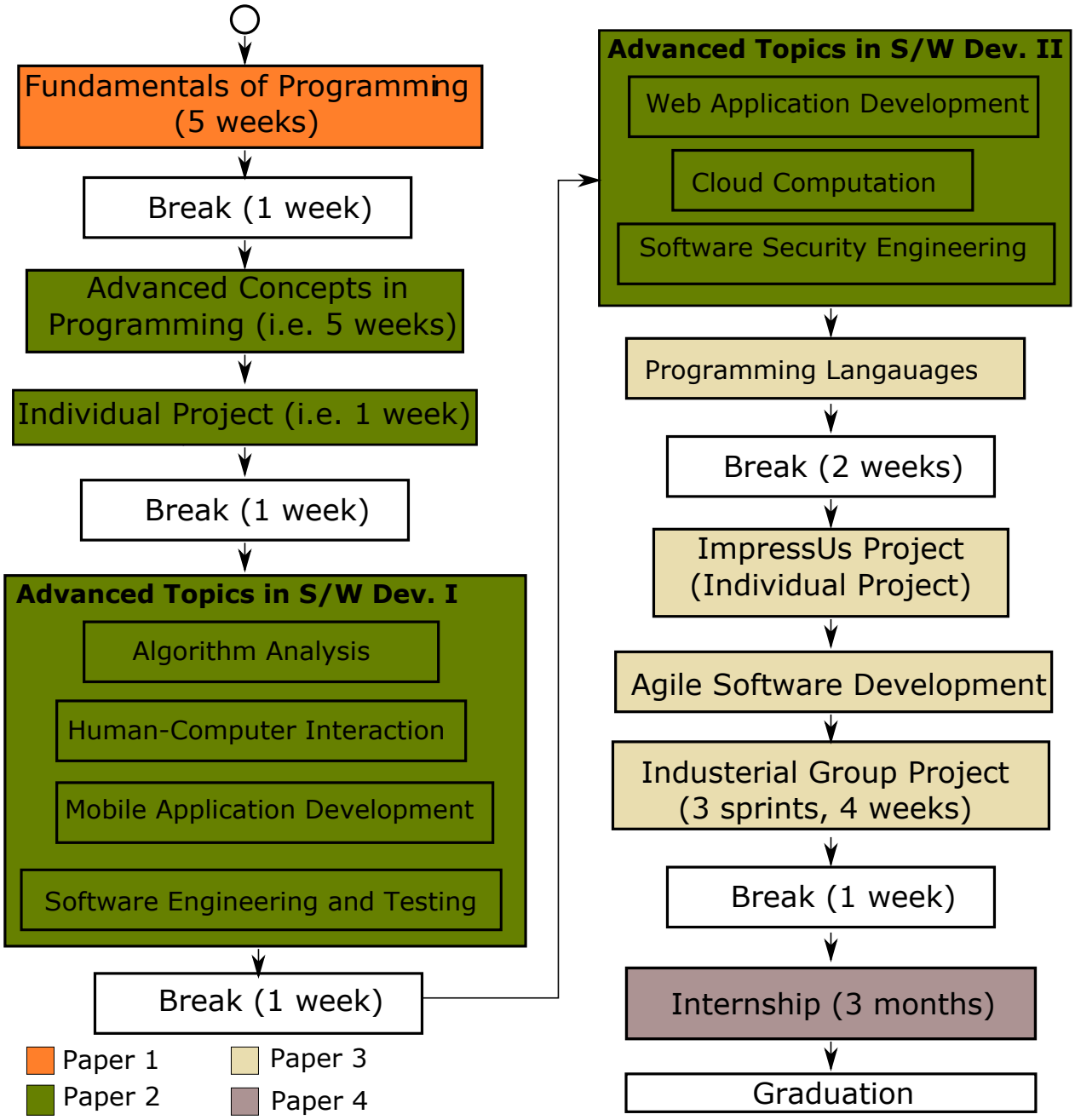

Fig. 1. The MSwDev Conversion Master Program Structure

for the project's duration to limit the issues around ownership. This provided a better experience for the students. However, it was still non-trivial to align three different programmes and coordinate the various needs (such as course learning objectives, assessments, and schedules). In 2022, the software development cohort worked alone on the industrial project without the UX and BA cohorts for three week-long sprints. This solved some issues in the Interdisciplinary Agile team but raised some other issues due to missing UX and BA cohorts.

It is worth noting that the evaluation method used in this paper is primarily based on a reflective analysis of observations made by the teaching staff. The contemplative practices of academic staff are significant in the learning/teaching cycle and have been proved as a crucial mental activity by many researchers [52]. The purpose of the reflective analysis was formative, and it is used for the improvement of courses over time. As highlighted by Lundqvist et al. (2019), "For [a] formative purpose, such as improvement of courses over time, it is generally considered best practice to follow this method instead of using quantitative tools, such as students' questionnaires, which are better at assessing individual teacher performance than improving course quality over time" [11].

The reflection methodology applied in this paper utilises four types of data: the students' feedback from the student responses to the end of the paper satisfaction survey, self-generated feedback from the authors while teaching the three papers, feedback from course moderators, and feedback from the program director. This paper discusses observations supported by the student's performance during and after the peak of the COVID-19 pandemic. It also outlines the challenges students face and supervisors when establishing a healthy relationship. No student feedback is quoted in this paper, but the students' feedback has influenced the authors' reflections. This reflection paper was not designed to be a research project; consequently, no ethical approval is required.

A. The role of the scrum master in the success of the industrial project

A Scrum master is essential for the success of an Agile software development project. They are assumed to fully understand the Scrum process theories and practices and act as an orchestrator who helps the team perform better [53]. The teaching team find this observation interesting as it is essential for the success of any SCRUM project. It relates to the skills required by the team members collectively and whether a member with proper leadership skills exists. It is always challenging to ask beginners to carry out specific roles on a team before they can practice them. It is worth mentioning that these roles were self-assigned in our case. Throughout the years of this conversion master program, the teaching team observed a wide range of behaviours (or what would be considered bad practices in the industry). This ranges from those who take a command-and-control approach (e.g. where they delegate work to team members and take control of the Scrum board rather than allowing it to be a collaborative tool) to those who take a more laissez-faire approach and can leave a team floundering. Given our program, those who do that are indeed new developers and asking them to self-organise is a challenge. The multidisciplinary teams were even more complex because they were always unsure about their 'boundaries' and were afraid of 'overstepping' even though students were encouraged to work as a 'whole team'. The main take-home lesson is that meeting facilitation skills do not constitute a good Scrum master if not combined with conflict resolution or facilitation skills and impediment manoeuvre skills.

B. Reflection 2: Multi-discipline Agile teams Vs single cohort agile team

Table II shows the agile team by year. For example, in 2017, only software developers were used in the team, while in 2019, an interdisciplinary team of software students, User Experience (UX) students, and Business Analyst (BA) students were used.

Although the teaching team has observed some communication and collaboration issues in the past when interdisciplinary cohorts are used, the teams perform better, especially customer-focused. Given that only SwDev students are used in 2016-2017 and 2021-2022, the teaching team observed the following twofold:

TABLE II
COHORT BY YEAR

Year	Cohort
2016-2017	SwDev
2017-2018	SwDev, UX
2018-2019	SwDev, UX, BA
2019-2020	SwDev, UX
2020-2021	SwDev, UX, BA
2021-2022	SwDev

1) The outputs from the years had very little focus or insights about the users built into the products. This starkly contrasted the multidisciplinary teams' outputs from previous years. It is believed this is due to the lack of UX and BA students in the team during these years and the missing customer-focused skills from the SwDev cohort.
2) The technical depth of the outputs from these years' single cohort teams was about the same as the multidisciplinary team years. In other words, the technical ability of the SwDev students did not see any change from that of the multidisciplinary teams used in the past.

C. Reflection 3: Group Performance

There is a group this year that had a big problem with separating the front-end development (UI) from the back-end (i.e. database) development to such an extent that the group could not bring the two parts together in the final product. After careful observation and investigation, this was due to the lack of openness from the Scrum master. This observation aligns well with what has been discussed regarding the role of the Scrum master. A Scrum master usually focuses on what the work is, not how the job is done. Unfortunately, within our teams, a Scrum master is also a software developer member who has to accomplish specific programming tasks. The Scrum master in this particular team could not complete his part and failed to communicate this earlier to the group until very late, when a solution was deemed impractical. It was a surprise to the Agile team and the teaching team since, in all the daily stand-ups, they assured us all was going by the plan and there were no significant impediments. This emphasises the role of communication in an Agile software development team, the need to be open to issues, and keeping 'ego' away.

Teams are also constantly challenged by the hybrid style of working. They become forced into having to learn to use digital tools before they even know how to use a backlog properly or how to manage their Scrum board properly. It is challenging as an instructor to help a team when all their team management artefacts, such as the backlog and the Scrum board, are digital since each tool has its idiosyncrasies and is

quite complex, that is unless you're a Trello [1], a Jira [2], or Miro [54] ninja. At the same time, the analogue artefacts are easy to learn, and students can get quick support when they encounter issues. Thus, the teams are encouraged to have a paper-based backlog on the wall to make it easy to see the progress.

D. Reflection 4: Student's performance in the three sprints

The students are required to submit three learning logs corresponding to the three project sprints they have. Our main observations from evaluating the first learning log are students struggled to:

1) implement agile practices such as stand-ups
2) decide the development stack to use and the amount of effort required to learn the new technologies
3) maintain good communication with the industrial partner.

It is believed this is not surprising, since it is the first time the students have worked on a real-life industrial project using the Agile software development methodology. By the end of the first sprint, the students got more familiar with and better understood Agile concepts such as Scrum master, product owner, stand-ups, acceptance criteria for 'done' tasks, etc. One interesting observation given the retrospectives at the end of the first spring is that all teams/groups had issues maintaining and using the backlog. This is mainly because most Agile team management tools are complex and require time to learn before the actual use. Unfortunately, our teams were learning those tools on the run while developing the products and practising the Agile software development method. One general issue amongst the groups was how to break down user stories into smaller tasks properly. All groups tended to add large tasks to the backlog at the beginning of the sprint, then realised those tasks were too big and needed a breakdown so multiple team members could work on them simultaneously. This is quite understandable as task breakdown is a skill software developers continuously improve over time. It was good that all teams acknowledged the problems and had plans to solve them using a comprehensive design meeting at the beginning of the sprint.

A couple of groups identified another challenge regarding the different speeds of team members to learning new technologies. The teaching team suggested the solution for this was to use pair-programming, which seemed to improve the team efficiency and gave the teams involved a better understanding of the overall product design.

During the second sprint, which is called the settling-down sprint, performance got better, especially on:

1) Managing client expectations by asking more questions and getting the industrial partner more engaged in decision-making for non-technical decisions. It also acknowledged the problem of cross-communication between technical (i.e. software developers) and non-technical people (i.e. clients).
2) Inter-team communication and collaboration. Pair programming was essential in improving communication and collaboration, especially for cross-group (i.e. a member from the front-end team is paired with another member from a back-end member) was helpful for the team members to understand different technologies.
3) Better management of backlog and work breakdown. Many teams have opted to use a physical storyboard rather than a digital version which made it easy to keep track of who is doing what and what tasks are in the pipeline. Also, good sprint planning supported good task breakdowns.
4) Some groups switched from being in-person to a hybrid model where stand-ups would have a Zoom meeting for those members working remotely due to COVID19 concerns or other personal issues. This proved to be as good as an in-person collaboration, especially during the stand-ups.

A widespread issue with most of the teams was time estimation for tasks. They tend to be overconfident in their ability to finish tasks by the time they estimated and mostly turn out to be underestimated. This is mainly due to inexperience. Proper sprint planning, which includes frank communication and comprehensive research, is required at the spring planning meeting to better estimate the time.

As students switched from learning new technologies in the third sprint, they progressed well on the technical part of building the IT product and implementing Agile practices. Apart from one team that has broken communication, as described in IV-A, all teams produced a reasonable MVP. The team struggled to identify the impediment and ask for help for that complex project. That was a failure of Agile processes. As a teaching team's belief, this should not have happened should the team culture has been more laissez-faire, not a command-and-control. Other teams with such a culture provided a reasonable MVP at the end of the sprint. For example, another team had some fun activities during the sprints to support such a good culture and their growth/progress in the project.

E. Reflection 5: Assessment

In 2017-2018 and 2018-2019, the assessments of the different cohorts were separated into other reports. This was not ideal because information could be lost between the different formats. The students also focused differently on the assessments, which introduced some friction between them. After that, the academic team started cooperating on the assessments. When projects are run with only SwDev and UX students in 2019-2020, our assessments were not well aligned/balanced among the SwDev and UX students. They did not invest their efforts equally (e.g. the UX students did not attend the final presentations because they held their presentations separately before the SwDev students' presentations). The teaching team worked on improving that in 2020-2021, and the UX, SwDev and BA students were more aligned. The plan was to have each assessment item have the exact weighting and deadline. This seems obvious, but the work

[1]https://trello.com, accessed 4 April 2022

[2]https://www.atlassian.com/software/jira, accessed 4 April 2022

coordinating that among three different university silos (i.e. essentially on three independent courses) is non-trivial.

V. Conclusion

This Research to Practice Full Paper presented a reflective study showing the observations and lessons learnt while teaching in a software conversion master program by the authors of this paper. The program is designed for people who want to change their careers in software development. The nature of such a program makes it a challenge to keep those people with slight or without any IT background up to the same level as a fresh software developer in the industry. Challenges are not only technical but also the directing of the program. One of those challenges is how to teach students Agile software development while they have no real exposure to the industries, not to mention their junior software programming skills and technicalities. Agile software development methodologies are the mainstream in the software industry. Teaching students Agile software development methodology is helpful for the students and the business owners.

Given the practices and observations from those years of teaching in that conversion master program, The following lists our recommendations for teaching Agile involving industrial partners:

1) Integrated Assessment; it is evident that an integrated assessment is needed when different cohorts are collaborating together for more consistency (i.e. avoid information loss due to other assessment formats) and fairer assessment across the different cohorts. The suggestion here is that the same assessment method (i.e. rubrics) is used across the different cohorts with the same required output (i.e. same report format).
2) Laissez-faire Scrum Master; A Scrum master role needs to be assigned to someone with minimum interference within the team instead of someone with a command and control approach. Given the inexperience students have at the time of the industrial project, it is essential to communicate, seek help, listen carefully to suggestions, etc., with the Scrum master.
3) COVID-19 impact; Compared to last year and the year before, COVID-19 did not impede the groups. This is basically because students are now more familiar with online tools such as Zoom, MS Teams, and Trello and are less stressed with the COVID-19 situation.
4) Agile is interdisciplinary; although with Agile interdisciplinary teams including UX, SwDev, and BA students, there have been some issues, the benefits outweigh the problems that arise from having a team with only SwDev. SwDev-only teams struggled to cope with the Agile process, especially with storyboards (i.e. requirements engineering), solution integration, and acceptance testing.
5) Keep it simple. Using analogue tools opposite to digital ones, such as Trello, makes it easy to track and help students. In other words, the teams are encouraged to have a paper-based backlog on the wall to make it easy to see the progress. The recommendation here is to keep a digital copy online, just in case people work from home due to COVID-19 or other interruptions.

References

[1] T. A. Ward, "Common elements of capstone projects in the world's top-ranked engineering universities," *European Journal of Engineering Education*, vol. 38, no. 2, pp. 211–218, 2013.

[2] N. Ragab, A. Ahmed, and S. M. Alhashmi, "Software engineering for security as a non-functional requirement," in *Intelligent Data Analysis and Applications, Proceedings of the Second Euro-China Conference on Intelligent Data Analysis and Applications, ECC 2015, Jun 29, 2015 - Jul 1, 2015, Technical University of Ostrava, Czech Republic*, ser. Advances in Intelligent Systems and Computing, A. Abraham, X. H. Jiang, V. Snásel, and J. Pan, Eds., vol. 370. Springer, 2015, pp. 347–357.

[3] J. Blanford, P. Kennelly, B. King, D. Miller, and T. Bracken, "Merits of capstone projects in an online graduate program for working professionals," *Journal of Geography in Higher Education*, vol. 44, pp. 1–25, 11 2019.

[4] A. Ahmed, K. Lundqvist, C. Watterson, and N. Baghaei, "Teaching cyber-security for distance learners: A reflective study," in *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1–7.

[5] D. Grant, A. Malloy, M. Murphy, J. Foreman, R. Robinson, and E. Summary, "Real world project: Integrating the classroom, external business partnerships and professional organizations," *Journal of Information Technology Education: Innovations in Practice*, vol. 9, 01 2010.

[6] F. Suleman, "The employability skills of higher education graduates: insights into conceptual frameworks and methodological options," *Higher Education*, vol. 76, 08 2018.

[7] K. Kelm and G. Miles, "Group projects in in-ground undergraduate and on-line graduatedegree programs: Guidelines for success," *Information System Education Journal*, vol. 4, no. 80, pp. 1–9, Sep. 2006.

[8] S. ATAWNEH, "The analysis of current state of agile software development," *Journal of Theoretical Applied Information Technology*, vol. 97, p. 22, 2019.

[9] M. Levy, I. Hadar, and I. Aviv, "Agile-based education for teaching an agile requirements engineering methodology for knowledge management," *Sustainability*, vol. 13, no. 5, p. 2853, 2021.

[10] D. Hussain and L. Söderlindh, "Software engineering, bridging theory and practice in an agile learning environment." in *2022 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2022, pp. 541–546.

[11] K. Lundqvist, A. Ahmed, D. Fridman, and J.-G. Bernard, "Interdisciplinary agile teaching," in *Proceedings of IEEE Frontiers in Education Conference (FIE)*, Cincinnati, Oct. 2019.

[12] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," in *2009 Agile Conference*, 2009, pp. 174–179.

[13] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Trans. Educ.*, vol. 55, no. 1, pp. 99–106, 2012. [Online]. Available: https://doi.org/10.1109/TE.2011.2142311

[14] Q. Liu, J. Ji, J. Chen, W. Zhao, and J. Yin, "Evaluating the effectiveness of situational case-based teaching: A view of concept mapping," in *Proceedings of ACM Turing Celebration Conference - China*, ser. TURC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 60–66.

[15] M. Frydenberg, D. Yates, and J. Kukesh, "Sprint, then fly: Teaching agile methodologies with paper airplanes," *Information Systems Education Journal*, vol. 16, no. 5, p. 22, 2018.

[16] J. Yang, X. L. Zhang, and P. Su, "Deep-learning-based agile teaching framework of software development courses in computer science education," *Procedia Computer Science*, vol. 154, pp. 137–145, 2019.

[17] O. Olayinka and M. Stannett, "Experiencing the sheffield team software project: A project-based learning approach to teaching agile," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 1299–1305.

[18] E. Sarikaya, S. Bagriyanik, and M. Gökalp, "Teaching agile software development using agile methods: A case study," in *2020 Turkish National Software Engineering Symposium (UYMS)*, 2020, pp. 1–6.

[19] M. Paasivaara, D. Voda, V. T. Heikkilä, J. Vanhanen, and C. Lassenius, "How does participating in a capstone project with industrial customers affect student attitudes?" in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2018, pp. 49–57.

[20] C. Matthies, R. Teusner, and G. Hesse, "Beyond surveys: Analyzing software development artifacts to assess teaching efforts," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–9.

[21] V. Hurbungs and S. D. Nagowah, *A Practical Approach to Teaching Agile Methodologies and Principles at Tertiary Level Using Student-Centred Activities*. Singapore: Springer Singapore, 2019, pp. 355–389. [Online]. Available: https://doi.org/10.1007/978-981-13-2751-3_17

[22] Z. Masood, R. Hoda, and K. Blincoe, "Adapting agile practices in university contexts," *Journal of Systems and Software*, vol. 144, pp. 501 – 510, 2018.

[23] E. Boti, V. Damasiotis, and P. Fitsilis, "Skills development through agile capstone projects," in *Frontiers in Software Engineering*, G. Succi, P. Ciancarini, and A. Kruglov, Eds. Cham: Springer International Publishing, 2021, pp. 97–112.

[24] G. Moser, R. Vallon, M. Bernhart, and T. Grechenig, "Teaching software quality assurance with gamification and continuous feedback techniques," in *2021 IEEE Global Engineering Education Conference (EDUCON)*, 2021, pp. 505–509.

[25] S. Fischer, M. Rosilius, J. Schmitt, and V. Bräutigam, "A brief review of our agile teaching formats in entrepreneurship education," *Sustainability*, vol. 14, no. 1, p. 251, 2021.

[26] M. Neumann and L. Baumann, "Agile methods in higher education: Adapting and using eduscrum with real world projects," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–8.

[27] A. Galkin, S. Blyumin, P. Saraev, and V. Pimenov, "Collaborative learning technologies for project work," in *2021 1st International Conference on Technology Enhanced Learning in Higher Education (TELE)*, 2021, pp. 256–259.

[28] J. H. Boockmann, K. Jacob, and G. Lüttgen, "Throw away student software at semester end? better not!" in *Software Engineering im Unterricht der Hochschulen (SEUH 2022)*, V. Thurner, B. Kleinen, J. Siegeris, and D. Weber-Wulff, Eds. Gesellschaft für Informatik, Bonn, 2022, pp. 23–28.

[29] M. Devare, "AMALGAMATION OF INDIRECT GAMIFICATION INTO DEVELOPMENT AND OPERATIONS (DEVOPS) COURSE TEACHING," *Information Technologies and Learning Tools*, vol. 87, no. 1, pp. 124–138, mar 2022. [Online]. Available: https://doi.org/10.33407%2Fitlt.v87i1.4619

[30] S. Fischer, M. Rosilius, J. Schmitt, and V. Bräutigam, "A brief review of our agile teaching formats in entrepreneurship education," *Sustainability*, vol. 14, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/1/251

[31] A. Heberle, R. Neumann, I. Stengel, and S. Regier, "Teaching agile principles and software engineering concepts through real-life projects," in *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2018, pp. 1723–1728.

[32] R. Bringula, R. Elon, L. Melosantos, and J. R. Tarrosa, "Teaching agile methodology through role-playing: What to expect and what to watch out," in *Proceedings of the 2019 3rd international conference on education and multimedia technology*, ser. ICEMT 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 355–359. [Online]. Available: https://doi.org/10.1145/3345120.3352733

[33] N. Naik and P. Jenkins, "Relax, it's a game: Utilising gamification in learning agile scrum software development," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–4.

[34] I. S. Elgrably and S. R. B. Oliveira, "Gamification and evaluation of the use the agile tests in software quality subjects: The application of case studies." in *ENASE*, 2018, pp. 416–423.

[35] R. Al-Azawi, S. A. Joe, M. Al-Obaidy, and J. Westlake, "The use of gamification technique in agile development methodology," in *International Workshop on Learning Technology for Education in Cloud*. Springer, 2019, pp. 3–13.

[36] M. Delen, "Bakere-design and development of a serious game for teaching user stories," Master's thesis, Department of Information and Computing Sciences, Faculty of Science, Utrecht University, June 2019.

[37] A. Baumann, "Teaching software engineering methods with agile games," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 1550–1553.

[38] O. Havazík, P. Pavlíčková, and J. Pavlíček, "Agile game in online environment," in *Advanced Information Systems Engineering Workshops*, A. Polyvyanyy and S. Rinderle-Ma, Eds. Cham: Springer International Publishing, 2021, pp. 17–25.

[39] E. O'Farrell, M. Yilmaz, U. Gulec, and P. Clarke, "Playsafe: Results from a virtual reality study using digital game-based learning for safe agile software development," in *European Conference on Software Process Improvement*. Springer, 2021, pp. 695–707.

[40] J.-P. Steghöfer and H. Burden, "One block on top of the other: Using minetest to teach scrum," in *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2022, pp. 176–186.

[41] J. M. Milićević, F. Filipović, I. Jezdović, T. Naumović, and M. Radenković, "Scrum agile framework in e-business project management: an approach to teaching scrum," *European Project Management Journal*, vol. 9, no. 1, pp. 52–60, 2019.

[42] G. B. Ghantous and A. Q. Gill, "An agile-devops reference architecture for teaching enterprise agile," *International Journal of Learning, Teaching and Educational Research*, vol. 18, no. 7, pp. 128–144, 2019.

[43] D. E. Rush and A. J. Connolly, "An agile framework for teaching with scrum in the it project management classroom," *Journal of Information Systems Education*, vol. 31, no. 3, pp. 196–207, 2020.

[44] D. Goularas, M. Ozkaya, T. Serif, and S. Goren, "A general framework for teaching software engineering and improving collaboration skills in multidisciplinary teams," *Journal of Aeronautics and Space Technologies*, vol. 14, no. 2, pp. 155–167, 2021.

[45] D. T. Avila, W. Van Petegem, and A. Libotton, "Asest framework: a proposal for improving teamwork by making cohesive software engineering student teams," *European Journal of Engineering Education*, vol. 46, no. 5, pp. 750–764, 2021.

[46] D. Tamayo Avila, W. Van Petegem, and M. Snoeck, "Improving teamwork in agile software engineering education: The asest+ framework," *IEEE Transactions on Education*, vol. 65, no. 1, pp. 18–29, 2022.

[47] K. Ofosu-Ampong, "The shift to gamification in education: A review on dominant issues," *Journal of Educational Technology Systems*, vol. 49, no. 1, pp. 113–137, 2020. [Online]. Available: https://doi.org/10.1177/0047239520917629

[48] A. Manzano-León, P. Camacho-Lazarraga, M. A. Guerrero, L. Guerrero-Puerta, J. M. Aguilar-Parra, R. Trigueros, and A. Alias, "Between level up and game over: A systematic literature review of gamification in education," *Sustainability*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/2247

[49] J. Swacha, "State of research on gamification in education: A bibliometric survey," *Education Sciences*, vol. 11, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2227-7102/11/2/69

[50] Q. Aini, U. Rahardja, and A. Khoirunisa, "Blockchain technology into gamification on education," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 14, no. 2, pp. 147–158, 2020.

[51] L. Reddy, N. Baghaei, H. Reinders, A. Ahmed, and S. A. Sardareh, "Persuasion via gamification: Supporting positive behaviour for learning (PB4L) school-wide pedagogy," *Set: Research Information for Teachers*, no. 2, pp. 20–25, 2021.

[52] A. Ahmed, C. Watterson, K. Lundqvist, and J. Ferreira, "Online student supervision: A reflective study on lessons and challenges," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–7.

[53] J. Owen and C. Wasiuk, "An agile approach to co-creation of the curriculum," *International Journal for Students as Partners*, vol. 5, no. 2, pp. 89–97, 2021.

[54] C. Skubik-Peplaski, S. Shisley, J. Edick, and W. Cook, "Agile learning and teaching with miro boards," in *Pedagogicon Conference Proceedings*, 2022.