The Journal of Systems & Software 186 (2022) 111156

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Understanding students' software development projects: Effort, performance, satisfaction, skills and their relation to the adequacy of outcomes developed ☆

Sherlock A. Licorish [a,*], Matthias Galster [b], Georgia M. Kapitsaki [c], Amjed Tahir [d]

[a] Department of Information Sciences, University of Otago, New Zealand
[b] University of Canterbury, New Zealand
[c] University of Cyprus, Cyprus
[d] School of Fundamental Sciences, Massey University, New Zealand

ARTICLE INFO

Article history:
Received 14 April 2021
Received in revised form 8 August 2021
Accepted 23 November 2021
Available online 15 December 2021

Keywords:
Course project effort
Student performance
Student satisfaction
Software engineering skills
Project challenges

ABSTRACT

Given the inclusion of (often team-based) course projects in tertiary software engineering education, it is necessary to investigate software engineering curricula and students' experiences while undergoing their software engineering training. Previous research efforts have not sufficiently explored students perceptions around the commitment and adequacy of effort spent on software engineering projects, their project performance and skills that are developed during course projects. This gap in skills awareness includes those that are necessary, anticipated and learned, and the challenges to student project success, which may predict project performance. Such insights could inform curricula design, theory and practice, in terms of improving post-study software development success. We conducted a survey involving undergraduate across four universities in New Zealand and Cyprus to explore these issues, where extensive deductive and inductive analyses were performed. Among our findings we observe that students' commitment of effort on software engineering project seems appropriate. Students are more satisfied with their team's collaboration performance than technical contributions, but we found that junior students seemed to struggle with teamwork. Further, we observe that the software students developed were of higher quality if they had worked in project teams previously, had stronger technical skills and were involved in timely meetings. This study singles out mechanisms for informing good estimation of effort, mentoring technical competencies and suitable coaching for enhancing project success and student learning.

1. Introduction

When completing team-based course projects at university (e.g., either as a course project or capstone project), students develop important skills associated with the management of diversity, mutual goal setting and group dynamics (Zolduoarrati et al., 2017; Exter and Ashby, 2019; Akdur, 2019; Boldyreva et al., 2020). In addition, software engineering education projects should meet the needs of industry by ensuring that students learn skills that will be relevant for post-study practice (Ryan, 2020) and for ensuring a sustainable software development community (globally). Such skills should cover various software engineering processes (e.g., requirements analysis, prototyping and design) and practices (e.g., version control), as well as transferable and soft skills (e.g., team work). However, several studies have identified gaps between graduating students' skills and industry expectations (Radermacher et al., 2014; Radermacher and Walia, 2013). Notwithstanding challenges related to aligning skills taught at university with industry needs, various complications arise when students work in groups, ranging from conflicts of opinions and shared beliefs at the team level, to personality differences at the individual level (Magana et al., 2019; Kruchten et al., 2019; Wehrs, 2020). In fact, there are times when such differences remain over student projects, to the point where teams never achieve functional status (Vartianen, 2019). Understanding these issues will help software engineering educators improve the learning experience of their students. It will also support the software engineering community's efforts to improve software engineering practice.

To this end, previous works have investigated project-based software engineering education and have assessed various educational methods via students' feedback (Marques et al., 2017;

☆ Editor: Patricia Lago.

* Corresponding author.
E-mail address: sherlock.licorish@otago.ac.nz (S.A. Licorish).

https://doi.org/10.1016/j.jss.2021.111156
0164-1212/

Schneiderm et al., 2020). However, students' perception of project success in relation to effort and team collaboration remains an issue that requires further investigation in different contexts, and in alignment with existing curricula design. In fact, limited awareness exists around the skills that are necessary, anticipated and learned during student software engineering projects, and the challenges to student project success, which may predict project performance. This is necessary in order to draw useful conclusions about how software engineering courses should be designed and which skills should be targeted. The need for evidence-based software engineering has been highlighted by Kitchenham et al. (2004), as what is assumed may not always be what is reality. In satisfying the tenets of evidence-based software engineering, researchers attempt to provide concrete evidence that may affect the success of future software development projects. This work follows the same evidence-based approach to assess the state of practices followed in software engineering capstone projects.

Fuelled by a drive to enhance software engineering outcomes and connected due to similarities in the courses we lecture, in this work, we focus on students' perceptions on project results in team-based course projects in software engineering education and link these to the adequacy of their project outcomes. We discuss relevant implications for the design of software engineering curricula and post-study industry expectations. For this purpose, we have conducted a survey in four universities in New Zealand and Cyprus involving a total of 105 students in four different courses, where both quantitative and qualitative data were collected. We analysed factors related to students effort, commitment, performance and satisfaction in these projects. Outcomes spotlight the need for good estimation of effort, mentoring technical competencies, appropriate skills development, and suitable coaching for enhancing project success.

The main contributions of this work lie in: (a) the provision of a survey framework that examines different dimensions of students' perceptions that may affect their overall learning, i.e., their effort, performance, skills and satisfaction, and linking these with curriculum design; (b) an extrapolation of knowledge from multiple software engineering educational contexts (New Zealand, Cyprus) that assists in achieving a holistic view of software engineering education, while also providing comparisons; and (c) the development of a baseline of perceptions and their relation to software development outcomes that supports reassessment in the face of course changes or improvements.

To drive our research, we have formulated five research questions (RQs):

- **RQ1.** How much effort do students commit to software engineering projects, and do students perceive this effort to be adequate?
- **RQ2.** How do students perceive their performance during software engineering projects?
- **RQ3.** How satisfied are students in software engineering projects with **(a)** the skills they acquire and project outcomes they deliver, and **(b)** their team?
- **RQ4.** What variables predict the adequacy of outcomes students develop during software engineering projects?
- **RQ5.** How do students perceive software engineering courses in terms of **(a)** the skills that are necessary, anticipated and learned, and **(b)** the challenges to their success?

The remaining sections of this paper are organised as follows. We provide our study background and related work in Section 2. We next provide our study setting in Section 3, before providing results in Section 4. We then discuss our findings and their implications in Section 5, and consider the threats to the study in Section 6. Finally, we provide concluding remarks and outline future research directions in Section 7.

2. Background and related work

2.1. Student software engineering project effort

The effort required to complete a software engineering course project depends on many factors such as the hours allocated and the number of credit hours set for each course, the weight of the project as an assessment item, as well as the expectations from institutions, teaching staff and students. However, students typically find that projects generally take longer than estimated. Vanhanen et al. (2018) found that students usually have issues with estimating the effort required to complete specific tasks within projects. This was mirrored in another study by Broman et al. (2012), who reported that time management is among the main skills that students needed in university projects. In a prior study by Vanhanen et al. (2012), students' feedback showed that a course involving a project generally requires more effort per credit unit compared to most other similar courses at the same level without a project. In particular, a student survey found collaboration and communication to take more effort and time than anticipated (Paasivaara et al., 2018). The study showed that 57% of students reported improvement in their understanding of the importance of collaboration and communication with other team members, while 44% of the students indicated that it was less difficult to learn new programming languages or other technologies. On the other hand, Ahtee and Poranen analysed potential risks in 76 final reports from students' software projects and found that scheduling problems and learning tools are among the main risks to success of students' projects (Ahtee and Poranen, 2009). These issues may be deep-seated in students' lack of commitment or underestimation of the effort that is required for successful software course projects.

2.2. Student performance in software engineering projects

One of the studies that investigated students' assessment of their performance in university projects found that, after completing their projects, students had improved their familiarity with agile development practices, programming and communication skills (Vanhanen et al., 2012). Bastarrica et al. (2017) found that students perceived that they performed well in their project and improved their soft skills (i.e., communication and collaboration). However, the perception of the value of technical skills identified by the students prior to the start of the projects usually drops by the end of the course. A study of students' perception of their performance in a capstone project by Dunlap (2005) found that after completing the project students felt significantly more confident in addressing real-world demands specific to software projects. There was a positive change in students' perception about their abilities to be software development professionals due to the confidence gained at the end of the project. The aspects of students' performance that stand out and the way such perceptions may relate ultimately to project outcomes remain nevertheless, an open area that has not been investigated previously.

2.3. Student satisfaction in software engineering projects

Software engineering projects can assist in the development of different kinds of skills depending on the specific course objectives and students' expectations. A previous systematic literature review presents self-reflection, conflict resolution, communication and teamwork as the main soft skills developed in software engineering education (Groeneveld et al., 2019). In the study of Bastarrica et al. 38 students from 7 project teams participated in a survey on the relative value obtained from their participation in

a software engineering capstone course (Bastarrica et al., 2017). As discussed before, students revealed that the value of soft skills grows, whereas the value of technical challenge drops over project duration. Students' satisfaction with their communication skills also improved when they contributed to open source projects in a Software Engineering class project (Marmorstein, 2011). Different soft skills acquisition was acknowledged by 48 students of a capstone project, with the main skills indicated by most students being external communication, organisational, internal communication, critical thinking and problem solving, as well as openness to changes and adaptability (Khakurel and Porras, 2020). In another study, computer science students who participated in a 16-week capstone course in software engineering experienced an increase in their self-efficacy, making them more confident to engage in similar activities in the future (Dunlap, 2005). Reflexive weekly monitoring (RWM) was used to study coordination, effectiveness and sense of belonging (Marques et al., 2017; Silvestre et al., 2015), where a higher sense of team belonging and satisfaction was observed for teams experiencing this intervention. Teamwork when combining undergraduate and graduate students in software engineering project teams was examined by Kapitsaki and Loizou (2018), where undergraduate students improved subject knowledge, teamwork and communication skills, whereas postgraduate students indicated improvements in leadership, communication and team management skills. Similarly, Schneider et al. adopted a blended agile development approach during capstone projects, where students reported that management and communication skills influenced the projects' success, and that project management tools used in these projects were useful for facilitating teamwork activities (Schneiderm et al., 2020).

Jacob and Faily studied the expectations of students at the beginning of a second year undergraduate project course of 27 weeks involving 35 teams of 5–6 students and compared reflections at the end of the same course (Jacob and Faily, 2019). In detail, Jacob and Faily asked: What expectations do students have in terms of team work at the start of a software development group project?, How do undergraduate students perceive the reality of team work after completing a software development group project?, and How do the expectations match the reality students face when developing software as a team? Results indicate that students are enthusiastic about group projects at the beginning and are not aware of specific problems related to collaborative software development. Also, the study found that over time students report problems related to the team and its members, the skills required to complete the project, related processes and the team environment. Finally, students' initial expectations and the reality as described at the end of the project were not aligned.

However, an investigation into skills acquisition more widely is still necessary for the community to further understand and implement appropriate changes to software engineering curricula in helping to develop prudent software engineers.

2.4. Predicting the adequacy of outcomes students develop

Insights about predicting the adequacy of outcomes students develop during software engineering projects can help educators adjust their teaching based on the behaviour of students (the class, teams or individuals), which could lead to post-project awareness during students' later software engineering careers. Some works have been conducted in the area of e-learning where researchers aimed to develop models to predict student performance. For example, early works by Minaei-Bidgoli et al. (2003) applied data mining to student assignment data available in an e-learning platform (e.g., attempts to get a right answer, time taken to answer questions, total time spent on a problem) to identify students at risk early and provide appropriate advising in a timely manner. Another area of related work is student assessment and monitoring. For example, Tubino et al. (2020) explored different "dimensions of assessment" (e.g., professional behaviour and accountability) and personas of students, observing that combining assessment dimensions and personas helps define expectations of how students should operate and perform in a team context. Similarly, some studies looked into the "treatment" of students and how it impacts behaviour and performance. For example, Marques et al. (2017) found that closely monitoring students in a student project through weekly reflections did not lead to higher productivity. On the other hand, students that were closely monitored felt a higher sense of team belonging and satisfaction. Paasivaara et al. (2018) investigated how industry capstone projects shape students' attitudes towards managing difficulties in projects. While their study did not investigate variables that predict the outcome of learning in a project course, it shows a shift towards "desired" attitudes because of the experience that students gain in an industry capstone project, where collaboration is singled out. It still remains unclear which factors affect the adequacy of outcomes students develop, and by extension, the skills that are central to their delivery of successful projects. We anticipate that such insights may apply to projects in industry, especially where students exit project courses to join the software engineering practitioner community.

2.5. Skills necessary, anticipated, learned and challenges to project success

Works which present reflections of instructors on skills necessary, anticipated and learned date back to the early days of software engineering education. For example, in 1994 Moore and Potts hypothesised that course projects help students learn how to handle size and complexity (Moore and Potts, 1994). Studies on the perception of students on the other hand are scarcer, and there are few studies that explore the skills necessary, anticipated and learned in student projects. For example, Abad et al. (2019) explore how authentic software project courses can be. Regarding skills learned, the study found that students made progress in some areas of problem solving skills. Regarding challenges, the study found that students struggled with their social skills (e.g., people handling skills, negotiations skills and organisational skills), understanding software quality and adaptability. Furthermore, there are various studies that collect the feedback of students at the end of a course project. For example, Raibulet and Fontana (2018) surveyed their students at the end of a project course to gather feedback on their collaboration and teamwork experience and on the use of a software analysis assessment tool. Feedback indicated that the students were enthusiastic about working in teams and about learning how to use tools they can also use in industry. Some studies investigated how students perceive what they learn in a course. For example, Melnik and Maurer (2005) explored whether students see a value in agile methods. They found that students prefer to continue to use agile practices at the workplace. In earlier works Sanders (2002) explored how students perceive techniques like extreme programming. He found that the majority of students were opposed to using extreme programming, but favoured pair programming. Initial insights here suggest that the software development community could benefit from a comprehensive understanding of the skills that are necessary, anticipated and learned by students during software project courses. In addition, knowledge about the challenges that affect student project success and their attainment of these skills could be noteworthy, in terms of designing curriculum that enables software engineering skills delivery and uptake. We set out such an agenda in this work.

3. Study setting

3.1. Instrument development

We conducted a web-based survey (in English) to gather data from software engineering students undertaking a project course. An initial draft of the survey was developed at University of Otago, and piloted among a small sample of students at this university in October of 2018. We then got together as a team across University of Otago, University of Canterbury, Massey University and University of Cyprus to revisit the instrument for wider distribution of the survey in 2019. We aligned the initial pilot survey instrument against the IEEE and ACM 2014 Software Engineering Education Knowledge (SEEK) and curriculum (T.J.T.F. on Computing Curricula, 2014), in view of teasing out various dimensions for data capture. The IEEE and ACM SEEK and curriculum provides guidance on what software engineering graduates should know and how such skills should be taught if they are to lead a successful software engineering career. In addition, it also aims to standardise software engineering education.

The guide includes professional knowledge (e.g., about standards and ethics), technical knowledge (e.g., covering problem analysis and design, development, testing, and deployment), teamwork (e.g., in relation to self-motivation, time management, team communication and teamwork), end-user awareness (e.g., user experience, client negotiation, leadership and questioning skills), design solutions in context (e.g., understand different design guidelines for various domains), perform trade-offs (e.g., involving feasibility evaluations, refining goals and objectives and ability to compromise), and foster continuing professional development (e.g., curiosity and initiative, ability to learn and thirst for knowledge and critical thinking). Software engineering curricula should thus be designed to ensure the aforementioned knowledge areas are taught in preparing professional software engineers.

Grounding our instrument in SEEK, the first section of our survey recorded general information. We followed previous surveys (Paasivaara et al., 2018; Bastarrica et al., 2017) in recording age, gender, university, course, academic major, year of study and the job the student aspire to do in the future. We then captured students' previous experience and the time they spent on the course, including if the students worked in a project team before, were employed in a salaried job, previously learned about project management (and software engineering), previously occupied a team role, previously acquired technical skills, the adequacy of the time that was allocated to the course, the average number of hours committed weekly by the students, the students' assessment of their workload, the name of the project they were involved and their project team size. We next captured students self-assessment of their project covering all areas of project management, ranging from meetings to more technical aspects (e.g., coding and deployment). Students' satisfaction with their project and skills they acquired were then captured, and then they provided an evaluation of their team, covering its size, all areas of project management and students' willingness to work with their team again. Finally, four open-ended questions were used to triangulate our quantitative data, capturing insights around (1) the skills students believe are necessary before entering the project, (2) the skills they anticipated to acquire during the project, (3) the important skills that were learned, and (4) the biggest challenges students faced. The survey received ethical approval and is available here.[1] We would be happy to share administrative access to the instrument with educators interested in running the survey. We provide an actual summary of various parts of the survey below:

(i) **Demographics and Experience:** What job would you like to do in the future?, Prior to doing this course, I previously worked in a project team at university with a minimum of 3 members?, Do you work or have worked in a salary job that is related to your study?, Have you learned about project management prior to enrolling in this course?

(ii) **Project Assessment (with optional scores: 1 = poor, 2 = fair, 3 = good, 4 = very good, 5 = excellent):**
I would rate my overall performance in the project as follows:
- Attendance at team meetings and lessons
- The quality of my work on project tasks
- My technical skills during the project
- My teamwork skills during the project
...

(iii) **Satisfaction and Performance (with optional scores: 1 = poor, 2 = fair, 3 = good, 4 = very good, 5 = excellent):**
In terms of the project overall:
- I have acquired new technical skills in completing my project
- I have acquired new project management skills having worked on this project
- I would rate my satisfaction with the project outcome
- I would rate my satisfaction with the team
...

(iv) **Team Assessment (with optional scores: 1 = poor, 2 = fair, 3 = good, 4 = very good, 5 = excellent):**
Referring to your team's performance on the project:
- In terms of technical competence, how would you rate your team?
- How would you rate the team's knowledge of project management?
- How would you rate the atmosphere of meetings in terms of members being given the opportunity to express their ideas?
- How would you rate the division of labour? Did everyone contribute?
...

3.2. Data collection

Undergraduate students that completed software engineering or software project management courses where a major software project was done were invited to participate in the survey. Candidate courses were the *Software Engineering* and *Software Project Management* courses at University of Canterbury, University of Cyprus, Massey University and University of Otago. To maintain anonymity of the students taking these courses we refer to the courses as UoC, UCY, MU and UoO, representing the abbreviations of the names of the universities. These courses typically run over 12 to 14 weeks, and are taken between years two and four in the Computer Science, Information Science/Technology or Software Engineering majors (Bachelors level). At the end of these courses, students will have practised the fundamental skills required to develop software systems using modern tools, practices and development environments (see Table 1). These courses build on, apply and extend material introduced in previous courses (e.g., software engineering processes, analysis, design, programming skills, programming paradigms, testing, web and mobile development, human computer interaction, etc.). These courses are practice-based, and they provide the first opportunity for students to undertake a sizeable piece of practical work that spans sufficient time to expose the complexities of modern software development.

Students undertaking the abovementioned courses typically operate in teams of between 4 to 7 members, and may participate

[1] https://tinyurl.com/y6nfqfl7

Table 1
Software development techniques, tools, programming courses, artefacts and team assignment method.

University	Techniques used	Tools used	Avg # Programming courses completed	Artefacts provided	Team assignment method
UoC	Up front analysis to understand domain and develop initial (but evolving) business idea, incremental, iterative development and testing, weekly stand-ups, weekly retrospections, peer reviews, coaching and feedback sessions with staff, continuous integration, and unit testing.	GitLab, GitLab CI, Maven, Cucumber, Trello, Slack, Clockify, JUnit, Eclipse, IntelliJ	4	Use cases, requirements, stakeholder analysis, design documents, UI prototypes, risk register, (evolving) project plan, the actual product, user documentation, source code, technical documentation, unit tests, and acceptance tests.	Educators' assigned
UCY	Up front analysis to understand domain and develop business idea, incremental, iterative development, bi-weekly meetings, continuous integration, unit and automated testing.	Git (GitHub), KanbanFlow, Selelnium, Bootstrap, Android Studio, PHP editor of choice	4	Use cases, requirements, design documents, progress reports, customer feedback, the actual product, user documentation, source code, technical documentation, unit tests, and automated tests.	Students' and Educators' assigned (educators assign additional students to existing teams when a team had only a few members)
MU	Up front analysis to understand domain and develop initial (but evolving) business idea, incremental, iterative development, weekly stand-ups, continuous integration, and unit testing.	Git (GitHub, GitLab), Travis CI, Jenkins, JUnit, PyUnit, Selenium, Eclipse, Pycharm, IntelliJ, Visual Studio, Flask, Diango, Angular, Trello, Slack and Discord	6	Use cases, requirements, design documents, prototypes, project plan, the actual product, user documentation, source code, unit tests, and acceptance tests.	Educators' assigned
UoO	Up front analysis to understand domain and develop initial (but evolving) business idea, incremental, expert estimation, iterative development, weekly stand-ups, weekly retrospections, coaching and feedback sessions with staff, continuous integration, and unit testing.	Git (GitBucket and GitHub), Gradle, Taiga, Slack, Discord, Trello, NetBeans, Visual Studio, JUnit, Eclipse, Android Studio	6	User stories, requirements, design documents, prototypes, risk register, project plan, the actual product, user documentation, source code, technical documentation, unit tests, and acceptance tests.	Educators' and Students' assigned (students could request to work with or avoid specific members)

UoC = University of Canterbury, UCY = University of Cyprus, MU = Massey University, UoO = University of Otago.

Table 2
Projects summary.

University	Project name	Description	Completed surveys	Team size	Client
UoC	Cashew Outside	Desktop app for food trucks operator	1	6	internal
UoC	FoodByte	Desktop app for food trucks operator	3	7	internal
UoC	FoodStart	Desktop app for food trucks operator	1	6	internal
UoC	iFOS	Desktop app for food trucks operator	2	6	internal
UoC	KaiTech	Desktop app for food trucks operator	2	6	internal
UoC	my truck	Desktop app for food trucks operator	1	6	internal
UoC	QuickBytes	Desktop app for food trucks operator	2	6	internal
UoC	Rosemary	Desktop app for food trucks operator	3	6	internal
UCY	E-Pottery	Web application for a pottery shop	3	5	external
UCY	LightGlide App	Web and mobile application for a small glide plane	4	4	external
UCY	Go Fit Gym	Web application for a gym	5	5	external
UCY	Real Estate Agency System	Web application for a real estate agency	4	5	external
MU	New Employee System	Web application for new employee registration	2	4	external
MU	Q&A Bot	Web application for questions and answers bot system	3	4	external
MU	Online Irrigation System	Web application for managing irrigation systems	2	4	external
MU	CI & Recipe Management App	Web application and stand-alone CI management system	2	4	external
MU	Embedded System Installer	Stand-alone scripting application	1	4	external
MU	Wiki Offline Editing Addon	Browser add-on	2	4	external
MU	Turitea Web Resources 1	Web-resource application for a landscape area	2	4	external
MU	Turitea Web Resources 2	Web-resource application for a landscape area	3	4	external
UoO	Childcare System	Web application for childcare facility	3	5	external
UoO	Contract Management System	Web application for contract and customer relationship management	2	5	external
UoO	E-commerce	Web application for online shopping	4	5	external
UoO	Gig-Guide	Tourism mobile application for events and festivals	4	6	external
UoO	Image Library	Web application for managing an image library	6	6	external
UoO	Library Occupancy Tracker	Web application for managing library occupancy (startup idea)	4	5	internal
UoO	Medicine Delivery	Mobile application for managing medical prescriptions (startup idea)	5	5	internal
UoO	Student Support Mobile App	Mobile application for supporting international students	4	6	external
UoO	Receipt Collector	Mobile application to manage receipt (startup idea)	3	4	internal
UoO	Review Analytics	Web application for performing business analytics	1	5	external
UoO	Restaurant Table Manager	Web application that manages restaurant bookings (startup idea)	5	5	internal
UoO	Roll Call	Mobile application for managing students' attendance	4	5	external
UoO	Room Booking	Web application for timetable management (startup idea)	4	5	internal
UoO	University Print	Mobile application for managing printing services	5	6	external
UoO	Waste Less App	Mobile application for managing groceries and pantry (startup idea)	3	6	internal

UoC = University of Canterbury, UCY = University of Cyprus, MU = Massey University, UoO = University of Otago.

in industrial projects involving an external client (see Table 2). Projects were balanced based on the number of members present in the team. Previous research has shown that companies use these projects for: recruiting developers, getting the software developed and researching new technologies (Paasivaara et al., 2019). Where students worked with internal clients (i.e., university staff), these clients replicate processes that are typical when students are working with external clients, allowing for the same processes of requirements elicitation and scoping (done via interviews), negotiation of schedules, verification and testing, etc (see Table 1). The central point of assessment for the courses is the project itself rather than a test or exam. Assessments cover business vision, use cases, requirements and stakeholder analysis, prototypes, design documentation, risk analysis, project plans, the actual software product, user documentation, issue tracking, source code, technical documentation, testing and test protocols (unit, acceptance, integration, etc.), and deployment (see Table 1). To obtain individual grades, instructors also consider source code repositories, issue trackers, project management logs, contribution statements and activity logs, and experiences from close collaboration with the students. Technical tutorials and mentoring sessions further validate students' individual contributions.

As noted in Table 1, students implement their projects using agile practices, involving incremental development with some up-front analysis to understand the domain and decompose initial (but evolving) business ideas. Students practise weekly stand-ups, weekly retrospections, other team meetings, coaching and feedback sessions with staff, and weekly technical tutorials on tools and technologies (e.g., build systems and configuration management, version control, unit testing and continuous integration). Students also aim for frequent delivery and incorporate continuous client feedback. As part of software delivery, students are expected to employ good software packaging and deployment techniques, and they use various software metrics for product and process improvement. All of the above is common to the four courses from which we drew students for this study, see Table 1. However, we still examine differences in outcomes across the courses, as there were subtle differences at times.

For instance, as noted in Table 1, in terms of techniques, while weekly stand-ups and retrospections were performed by teams at University of Canterbury and University of Otago, bi-weekly meetings were done by University of Cyprus, and only weekly stand-ups were done at Massey University. Similarly, coaching and feedback sessions with staff were done at University of Canterbury and University of Otago, however these techniques were not used at the other two universities. In the case of University of Cyprus, feedback was integrated in the bi-weekly meetings. While the tools used across the four universities were different at times in Table 1, they all supported project management and communication, distributed version management, continuous integration and testing. Students at University of Canterbury and University of Cyprus were exposed to four programming courses prior to completing their projects, while those of University of Otago and Massey University were a bit more exposed (having completed six courses on average). Table 1 shows that artefacts provided by students across the four universities covered the entire software development life cycle, from project scoping and requirements documents to the tested and packaged software. However, at times there were differences in the actual form of the artefact that was provided (e.g., User stories versus Use cases). Students at University of Canterbury and University of Otago

were also required to provide a risk register, while those at the other two university were not required to consider this artefact. Furthermore, teams were assigned exclusively by educators at University of Canterbury and Massey University, while a mixed approach was used by educators at University of Cyprus and University of Otago. In particular, educators assigned additional students to existing teams when a team had only a few members at University of Cyprus; however, students could request to work with or avoid specific members at University of Otago (see Table 1).

Of 49 students enrolled in UoC at University of Canterbury, 46 were local students (from New Zealand) and three were international students. Nineteen (19) of the total students agreed to take part in the study. The total cohort of students enrolled in UCY was 22, where 16 agreed to take part in the study. All students enrolled at University of Cyprus were local students. Massey University had 32 students enrolled in the MU course (18 local student and 14 international students), where 22 students agreed to participate in the study. Finally, 79 students were enrolled in UoO at University of Otago (71 local and 8 international), where 58 agreed to complete the survey. Given the disparities in numbers for local and international students in the courses we did not differentiate between the two groups during data collection and analysis. Also, all students in the courses were completing the local university programme, so there was no difference in the actual education that these students were receiving.

Altogether, 115 students from the four universities completed the survey (UoC = 19, UCY = 16, MU = 22 and UoO = 58), however 10 responses were removed as students did not clearly identify their project teams, leaving 105 completed and useful responses that were suitable for analysis. This represents a 57.7% response rate (see Section 4 for further details). For most projects more than half of the members in the team completed the survey, thus providing a detailed view of students' project realities. Table 2 provides a summary of the projects across the four universities, where it is seen that projects (35 altogether) covered the full spectrum of environments, including desktop (standalone), web and mobile applications. University of Canterbury (UoC) projects were similar in scope and size but addressed different solutions for various food truck operators. Languages include `Java`, `JavaScript`, `C#`, `Ruby`, `HTML`, `CSS`, `SQL`, `Delphi` (for legacy integration), `.NET`, `PHP` and `Python`.

3.3. Data analysis and measures

We used statistical analysis to answer our research questions. We then adopted an inductive (bottom-up) approach to content analysis to test whether themes appeared in the qualitative data/open ended responses (Patton, 1990). The procedure involved open coding where students' responses were read and re-read for familiarisation and initial codes were identified based on explicit, surface-level semantics in the data, rather than implicit responses and preconceptions (see Braun and Clarke, 2006). Through axial coding, codes were recombined, and connections were formed between ideas. Then, we used thematic mapping to restructure specific codes into broader themes. Finally, following Braun and Clarke's 2006 selective coding procedure, the resulting themes were refined and organised into a coherent, internally consistent account, and a narrative ("story") was developed to accompany each theme. The outcomes from our analyses were used to answer the five research questions in Section 1.

Students' responses in relation to their commitment of effort to the software development projects and perception of its adequacy were analysed to answer RQ1. Effort was measured based on the average number of hours students committed each week. We next analysed students' responses in relation to their performance during the software development projects to answer RQ2. Students provided responses to measure their performance along several dimensions from the IEEE and ACM 2014 SEEK, which were graded on a five point scale. Dimensions include *Attendance*, *Activeness*, *Task Difficulty*, *Teamwork Responsibility*, *Individual Work*, *Quality of Work*, *Willingness*, *Negotiation*, *Listening*, *Project Management (PM) Skills*, *Technical Skills*, *Presentation* and *Teamwork*. Students' responses in relation to their satisfaction with the skills they acquired and the project outcomes they delivered and their satisfaction with their team were analysed to answer RQ3.a and RQ3.b respectively. Also motivated by IEEE and ACM 2014 SEEK, students' satisfaction was measured on a five point scale along the following dimensions: *Technical Skills*, *PM Skills*, *Project Outcomes*, *Project Team* and *Project Assigned*. Students' satisfaction with their team was measured along the dimensions of *Technical*, *Knowledge*, *Meetings*, *Agenda*, *Preparedness*, *Willingness*, *Atmosphere*, *Work Grade*, *Direction*, *Labour Division*, *Communication* and *Outcome Quality*. We then analysed the variables that predict the adequacy of software development outcomes students develop to answer RQ4. All measures used to study RQ1, RQ2 and RQ3 were then used in our prediction model, and the adequacy (and quality) of software development outcomes was measured based on project *score/grade* that was awarded by course lecturers. Demographic measures also contributed to our modelling.

Finally, we analyse students' responses for the skills they perceive to be necessary, skills they anticipated and skills they learned to answer RQ5.a, and responses for the challenges to their success during software development to answer RQ5.b. This latter analyses took on a qualitative (inductive) tone, where outcomes in response to these dimensions were interpreted in relation to the SEEK knowledge base, and used to triangulate the earlier quantitative findings (to answer RQ1–RQ4).

4. Results

4.1. Students' demographics

Of the 105 respondents, 79 (75.2%) identified as male and 26 (24.8%) as female. Given the disparities in numbers for the male and female students, we did not heavily differentiate between the two groups during analyses for answering RQ1, RQ2 and RQ3. However, we check the effect of all variables (including gender) in our prediction models to answer RQ4, and where necessary, directions are provided for our analyses to answer RQ5.

Table 3 provides summary statistics of the surveyed students. The average age of respondents was 22.3, where students were typically studying in their fourth year on average, with the exception of those from University of Canterbury who were completing their second year of study. In terms of Majors, students were completing Computer Science (30, 28.6%), Information Science/Technology (43, 40.9%) and Software Engineering (32, 30.5%), with the highest proportion being Information Science students enrolled at University of Otago (40, 38.1%). Looking at the future jobs respondents are aspiring to perform, it is observed in Fig. 1 that Software Developer, Software Engineer and Business Analyst outweighed all other jobs.

Of the 105 students, Table 4 shows that 30 students were already employed in a (salaried) software development (SWD) job. On the subject of previous knowledge (whether gained at university or elsewhere), Table 4 shows that a large percentage of the students (80.0%) had previous software project management (PM) experience, and by extension, also occupied a team role (i.e., 77 students or 73.3%).

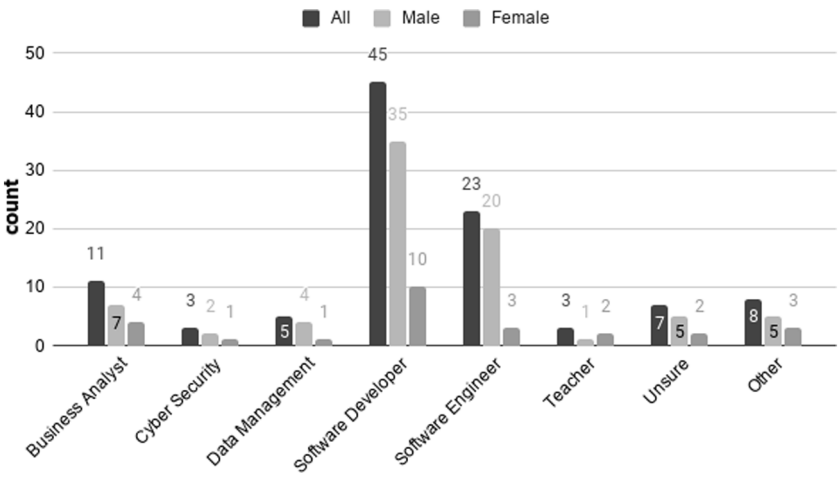

Fig. 1. Future jobs respondents are aspiring to perform.

Table 3
Students' summary statistics.

University	# Students (%)	Age (avg)	Years of study (avg)	# Majors
UoC	15 (14.3)	19.5	2.0	SE = 15
UCY	16 (15.2)	22.3	3.7	CS = 16
MU	17 (16.2)	27.2	3.8	CS = 6 IT = 3 SE = 8
UoO	57 (54.3)	21.6	3.8	CS = 8 IS = 40 SE = 9
Overall	105 (100)	22.3	3.5	CS = 30, IS = 43 SE = 32

SE = Software Engineering, CS = Computer Science, IS = Information Science, IT = Information Technology.

Table 4
Students' previous experience and knowledge statistics.

University	Previous SWD job (%)	# Previous PM knowledge (%)	# Previous SWD role (%)
UoC	1 (3.3)	12 (14.3)	10 (13.0)
UCY	6 (20.0)	15 (17.9)	15 (19.5)
MU	7 (23.3)	13 (15.5)	12 (15.6)
UoO	16 (53.3)	44 (52.4)	40 (51.9)
Overall	30 (28.6)	84 (80.0)	77 (73.3)

Table 5
Student's effort and perception of project workload.

University	# Adequate time allocated (%)	Weekly hours (avg)	# Workload rating (%)	Team size (avg)
UoC	14 (13.6)	11.9	8 (11.6)	6.1
UCY	15 (14.6)	13.9	11 (15.9)	4.6
MU	17 (16.5)	17.4	13 (18.8)	4.0
UoO	57 (54.8)	8.2	37 (55.3)	5.2
Overall	103 (98.1)	11.1	69 (65.1)	5.1

4.2. Students' effort (RQ1)

Examining the effort students expended on their projects in Table 5, we observed that nearly all students felt that they were able to allocate adequate time to their project (103 students or 98.1%). The pattern of outcomes across universities almost duplicated the overall data set in Table 3. We observe in Table 5 that students typically dedicated around 11 h on average each week to their projects.

There was variance in the average number of hours dedicated each week to project work across universities, with Massey University students working the most hours each week (17.4 h), and University of Otago students working the least (8.2 h). This survey question had three possible answers: too much, just the right amount (adequate), and too little.

Table 5 shows that only 69 students (65.1%) felt like the project workload was adequate, while 31 students felt like it was too much and 5 felt like it was too little. Team size averaged around five members, with some teams being a bit larger than others, however overall, all teams were within the typically recommended agile guidelines (which recommend 7 members plus or minus 2 Schwaber and Beedle, 2002).

We further examine these outcomes for statistical differences. We use the Kruskal–Wallis H test in confirming to the four assumptions that are required for its suitability and validity of use (i.e., interval and ratio scale data, four categories representing the universities, independent observations and similar data variability) (Kruskal, 1952). We found statistical differences across the universities for our Kruskal–Wallis H tests for the number of hours dedicated each week by students ($X^2(3) = 23.914$, $p < 0.01$) and their team sizes ($X^2(3) = 62.826$, $p < 0.01$). Post hoc pairwise comparisons with appropriate Bonferroni adjustments (Vogt and Johnson, 2011) show that students enrolled in UoO committed significantly less hours weekly to their projects than those enrolled in UoC and MU (Bonferroni adjusted significance, $p = 0.007$ and $p < 0.001$). For team size, UoC stood out, showing statistically significant differences to the other teams (UoC-UCY, $p < 0.001$; UoC-MU, $p < 0.001$; UoC-UoO, $p = 0.004$).

4.3. Students' assessment of performance (RQ2)

Students graded their performance between one and five along several dimensions in Table 6. A grade of one is assessed as poor while five is assessed as excellent. On average students graded their attendance at meetings (4.6), activeness on project tasks (4.2), solving difficult tasks (4.1), taking responsibility for teamwork (4.2), taking responsibility for individual work (4.4), the quality of the work done (4.2), willingness to participate in teamwork (4.4), negotiation skills (4.1), listening skills (4.4)

Table 6
Students' perception of their performance (avg (x) scores).

University	Attendance	Activeness	Task difficulty	Teamwork responsibility	Individual work	Quality of work	Willing teamwork	Negotiation	Listening	PM skills	Technical skills	Presentation	Teamwork
UoC	4.7	4.1	3.9	4.1	4.3	4.1	4.3	3.9	4.2	3.5	3.7	3.9	4.1
UCY	4.5	4.6	4.2	4.4	4.3	4.4	4.4	4.1	4.3	4.2	4.1	4.2	4.4
MU	4.5	4.2	4.4	4.4	4.4	4.2	4.5	4.1	4.2	3.9	4.1	3.9	4.4
UoO	4.6	4.2	4.0	4.2	4.4	4.1	4.4	4.2	4.5	3.9	3.6	3.5	4.4
Overall	4.6	4.2	4.1	4.2	4.4	4.2	4.4	4.1	4.4	3.9	3.8	3.8	4.3

Table 7
Students' project satisfaction (avg (x) scores).

University	Technical skills	PM skills	Project outcomes	Project team	Project assigned
UoC	3.9	3.9	3.8	4.0	3.9
UCY	4.6	4.5	4.5	4.2	4.2
MU	4.2	4.0	3.8	3.9	4.2
UoO	4.1	4.4	4.1	4.2	4.0
Overall	4.2	4.3	4.1	4.1	4.1

and teamwork skills (4.3) close to excellent (between 4 and 5). However, project management (3.9), technical skills (3.8) and presentation skills (3.8) were rated lower (between good and very good, 3–4). This pattern of lower grading for the latter skills was consistent across all universities, as was the case for the higher grading of the former skills. Overall, students across all universities awarded themselves comparable scores.

We follow up with formal statistical testing using the Kruskal–Wallis H test as was done for RQ1, where statistically significant differences was only observed for students presentation skills ($X^2(3) = 8.371$, $p = 0.039$). Post hoc pairwise comparisons with appropriate Bonferroni adjustment show there was only statistically significant differences between students of UCY and UoO ($p = 0.034$), confirming that UCY students recorded significantly higher scores.

4.4. Students' and teams' satisfaction (RQ3)

4.4.1. Students' satisfaction — skills and outcomes (RQ3.a)

As above for RQ2, students graded their satisfaction between one and five along several dimensions, where a grade of one is assessed as poor while five is assessed as excellent. Table 7 provides students' grading of their satisfaction with the skills they achieved, project outcomes, project team and project assigned. It is shown that students consistently graded their satisfaction with the technical skills gained (4.2), project management skills gained (4.3), project outcomes delivered (4.1), project team (4.1) and the project they were assigned (4.1) close to excellent (between 4 and 5). Across universities it is observed that University of Canterbury students (students in their second year and the "youngest" and least experienced cohort in the sample) were the least satisfied with the skills gained, project outcomes, team and project assigned, while University of Cyprus students (close to their fourth year) were the most satisfied (scoring an average of 3.9 versus 4.4 overall).

We follow up with formal statistical testing using the Kruskal–Wallis H test as was done above, where statistically significant differences was only observed for students' satisfaction with the project outcomes delivered ($X^2(3) = 8.635$, $p = 0.035$). However, these results did not hold when post hoc pairwise comparisons with appropriate Bonferroni adjustments were performed ($p > 0.05$ for all comparisons), confirming that students across universities responded similarly to these questions.

4.4.2. Students' team satisfaction (RQ3.b)

Regarding students' satisfaction with their team, we explored two aspects:

(i) **Team Dynamics, Performance and Atmosphere:** Table 8 provides students' assessment of their team's dynamics, performance and atmosphere, as above, graded between one and five, where five is excellent. Overall, students graded team meetings (4.0), willingness (4.1), atmosphere (4.3) and communication (4.0) excellent. On the other hand, team technical competency (3.9), knowledge (3.7), conforming to meeting agenda (3.8), preparedness (3.7), grade of work (3.7), direction for follow up tasks (3.8), division of labour (3.7) and the quality of outcomes (3.9) were scored lower (between good and very good, 3–4).
Across universities, students assessed their teams' knowledge and preparedness least favourable, with an average grade of 3.6, and particularly scoring their team's meetings and preparedness poorly in the case of University of Canterbury (2.9 and 2.7 out of 5 respectively). Students at University of Cyprus graded their team's dynamics, performance and atmosphere most favourably (average of 4.3), and were particularly pleased with the atmosphere of meetings in terms of members being given the opportunity to express their idea freely (scoring this dimension 4.7 on average). This pattern resembles the pattern regarding students' satisfaction in the previous paragraph for RQ3.a where satisfaction was lower amongst the less experienced students and students at an earlier stage of their degree.
Formal statistical testing done using the Kruskal–Wallis H test confirmed statistically significant differences for team meetings ($X^2(3) = 20.663$, $p < 0.01$), willingness ($X^2(3) = 9.682$, $p = 0.021$), atmosphere ($X^2(3) = 16.120$, $p < 0.01$), communication ($X^2(3) = 12.808$, $p < 0.01$), knowledge ($X^2(3) = 10.101$, $p = 0.018$), agenda ($X^2(3) = 8.704$, $p = 0.033$), preparedness ($X^2(3) = 17.078$, $p < 0.01$), direction for follow up tasks ($X^2(3) = 10.179$, $p = 0.017$) and division of labour ($X^2(3) = 8.390$, $p = 0.039$). Post hoc pairwise comparisons with appropriate Bonferroni adjustments show that students enrolled in UoC returned significantly lower scores than UoO, UCY and MU (at times) for many of these dimensions. Students at UoC recorded significantly lower scores for team meetings, willingness, communication and agenda when compared to those at UoO (Bonferroni adjusted significance, $p < 0.01$, $p = 0.039$, $p = 0.019$ and $p = 0.020$ respectively). These students at UoC also recorded significantly lower scores than those at UoO and MU for atmosphere (UoC-UoO, $p = 0.008$; UoC-MU, $p = 0.029$), lower scores than those at UoO and UCY for knowledge (UoC-UoO, $p = 0.028$; UoC-UCY, $p = 0.033$) and lower scores than those at UoO, MU and UCY for preparedness (UoC-UoO, $p = 0.002$; UoC-MU, $p = 0.016$; UoC-UCY, $p = 0.001$). Furthermore, students at UoC recorded significantly lower scores than those at UCY for direction for follow up tasks and division of labour ($p = 0.011$ and $p = 0.033$ respectively).

(ii) **Team Size and Future Involvement:** We summarise students' perception of their team size and their willingness to work on future projects with their teams in Table 9. For this questions, students had a choice of one of the three options: too large, just the right size or too small. Overall, 79 students (75.2%) felt like their team was *"just the right*

Table 8
Students' team assessment (avg (x) scores).

University	Technical	Knowledge	Meetings	Agenda	Preparedness	Willingness	Atmosphere	Work Grade	Direction	Labour Division	Communication	Outcome Quality
UoC	3.9	3.0	2.9	3.1	2.7	3.6	3.9	3.5	3.3	3.2	3.4	3.6
UCY	4.3	4.1	4.1	4.0	4.2	4.4	4.7	4.3	4.5	4.4	4.3	4.3
MU	3.8	3.5	3.6	3.8	3.7	3.7	3.8	3.6	3.8	3.8	3.5	3.8
UoO	3.9	3.8	4.3	4.0	3.8	4.4	4.5	3.7	3.9	3.6	4.2	4.0
Overall	3.9	3.7	4.0	3.8	3.7	4.1	4.3	3.7	3.8	3.7	4.0	3.9

size", with 20 students noting that their team was too large and six recording that their team was too small.

Examining students' responses for their willingness to work with the team again in Table 9, of the three possible answers provided (yes, maybe or no), we observed that 90 students responded *"yes"* or *"maybe"*, while 15 students responded *"no"*. The results are consistent across universities, and resemble the total data set (e.g., 12 of the 15 University of Canterbury (UoC) students that completed the survey would be happy to work with the team again).

We follow up with formal statistical testing using the Kruskal–Wallis H test as was done above, where statistically significant differences were not observed for either of these dimensions. This outcome confirms that students across universities responded similar to these two questions.

Table 9
Students' willingness to work with team again and perception of team size.

University	# Work with team again (%)	# Team size adequacy (%)
UoC	12 (13.3)	12 (15.2)
UCY	15 (16.7)	14 (17.7)
MU	12 (13.3)	11 (13.9)
UoO	51 (56.7)	42 (53.2)
Overall	90 (85.7)	79 (75.2)

Table 10
Variables which correlated with project score (significant correlations only).

Variable	r	p-value
Study year	0.255	0.004
Previous project	0.302	0.001
Previous knowledge	0.217	0.013
Previous skills		
Planning	0.195	0.023
Analysis design	0.206	0.017
Testing integration	0.258	0.004
Release deployment	0.315	0.000
Time allocation	0.283	0.002
Performance activeness	0.169	0.042
Performance quality	0.170	0.041
Performance PM skills	0.303	0.001
Project PM skills	0.253	0.004
Project outcome satisfaction	0.257	0.004
Team performance meetings	0.373	0.000
Team performance agenda	0.315	0.000
Team performance preparedness	0.301	0.001
Team performance willingness	0.206	0.017
Team performance work grade	0.306	0.001
Team performance direction	0.245	0.006
Team performance outcomes quality	0.251	0.005
Team performance communication	0.207	0.016

4.5. Predicting students' performance (RQ4)

Students had an average project score of 83.7 for their performance (out of 100), with standard deviation of 9.6. We observed that students from University of Cyprus had the highest score on average (89.8), with University of Canterbury students scoring the lowest (69.0). The average project score for students at Massey University was 82.8, and those at University of Otago scored 86.2. We explore the variables that predict the adequacy of outcomes students produced (gauged via the project score). Our data followed a normal distribution (see Fig. 2), and thus, we first perform Pearson's correlation analysis, before then modelling the data with a multiple regression. Table 10 provides the statistically significant (p-value < 0.05) correlation outcomes for the variables which correlated with project score. These results are assessed considering Cohen's classification (Cohen, 2013), in terms of low ($0 < r \leq 0.29$), medium ($0.3 \leq r \leq 0.49$) and high ($r \geq 0.50$) correlations. It is seen in Table 10 that 21 variables positively correlated with the project score students were awarded (italics denote medium correlation). Those most noteworthy (with medium correlation) in Table 10 include: if students had worked in project teams previously (Previous Project, $r = 0.302$), if students had previous technical skills (Previous Skills — Release Deployment, $r = 0.315$), if students demonstrated good project management skills (Performance PM Skills, $r = 0.303$), if teams had timely meetings (Team Performance Meetings, $r = 0.373$), had good agendas when they met (Team Performance Agenda, $r = 0.315$), if members were very prepared (Team Performance Preparedness, $r = 0.301$), and tasks delivered were assessed to be of a high grade (Team Performance Work Grade, $r = 0.306$).

We next modelled the relationship between the variables (predictors) in Table 10 and the project score students attained (dependent variable) using multiple regression. The results indicated that the model explained 25.0% of the variance and that the model was a significant predictor of project score, $F(3,102) = 11.34$, p-value < 0.001. The variables that significantly predicted students project score in our regression model were: if students had worked in project teams previously (Previous Project, $\beta = 5.740$, p-value $= 0.011$), if students had previous technical skills (Previous Skills — Release Deployment, $\beta = 4.602$, p-value $= 0.011$) and if teams had timely meetings (Team Performance Meetings, $\beta = 2.477$, p-value $= 0.011$). Other variables seen in Table 10 were not significant predictors of the awarded project score in our regression model, notwithstanding the significant correlation when modelled in isolation. The Variance Inflation Factor (VIF) for our predictors were Previous Project $= 1.041$, Previous Skills — Release Deployment $= 1.040$ and Team Performance Meetings $= 1.070$, which are all below the range that is typical for multicollinearity (i.e., 5) (Kock and Lynn, 2012). The Condition Index (CI) statistics were also all below the range for multicollinearity (below 10) (Hair et al., 1998). A histogram of the regression standardised residual for the project score and Normal probability plot (P-P) plot suggests that the observed standardised residuals follow a normally distribution (see Figs. 2 and 3).

4.6. Skills necessary, anticipated, learned; challenges to student success (RQ5)

Inductive analysis was done on students' open ended comments to understand how students perceive software engineering courses in terms of the skills necessary, skills anticipated, skills learned and challenges to their success. In addition, our analysis provides triangulation and insights into why students performed

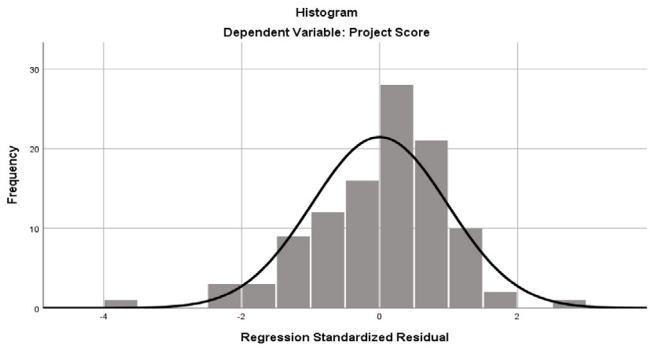

Fig. 2. Histogram for regression standardised residual for project score.

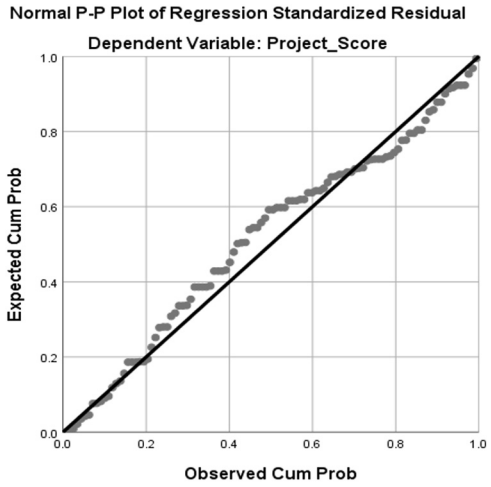

Fig. 3. Normal probability plot of regression standardised residual for project score.

better if they had previous experience in project teams and software deployment skills, and if meetings were conducted in a timely manner (findings for RQ4). We explore students' responses to four open-ended questions (students' pooled comments), focusing on the skills necessary before entering a project (56 comments), the skills students anticipated to acquire (33 comments), the most important skills learned (50 comments) and the biggest challenges students faced in the project (54 comments). As noted above, at times comments presented multiple themes, and thus, such comments were assigned multiple codes. While we followed an inductive approach to this analysis, all comments were coded by one author before a second author was given the same comments to code, confirming their agreement with the first code assigned. Agreement between coders was then checked using Holsti's coefficient of reliability measurement (C.R.) (Holsti, 1969), which revealed a 93.9% agreement. We did not further discuss these codes given the almost perfect agreement and our approach to develop a narrative (''story'') to accompany each theme, where the actual comments are included (see below). We anticipate that this transparency allows for the appraisal of the objectivity of our inductive outcomes. We analyse and present students' responses in turn below to answer RQ5.a and RQ.5b.

4.6.1. Skills necessary, anticipated and learned (RQ5.a)

(i) **Skills Necessary:** At the high end of the scale, students expressed that technical skills (e.g., coding, testing and deployment) were necessary for succeeding in a software project course. Of the 56 comments that were provided in response to this question, technical skills were recorded 39 times, meaning that 69.6% of the students alluded to the relevance of technical skills for succeeding in a software project course. In many instances students were specific in their recommendation, including in terms of the value of having knowledge of multiple languages and the software deployment process in order to succeed (refer to Fig. 4). Students noted that you definitely need (P = participant):

> P10 ''Technical Skills - particularly the language that you were using.''
> P13 ''Web frameworks, knowledge of deployment in industry''.
> P21 ''Coding skills. Knowledge in programming was very important for this course''.
> P33 ''A high level of programming skills with at least two languages''.

Interestingly, students also highlighted the value of soft skills, including communication skills (alluded to 13 times or by 23.2% of the students), teamwork skills (alluded to 9 times or by 16.1% of the students), and time management skills (alluded to 5 times or by 8.9% of the students). Students noted that:

> P1 ''What is more important is being able to work well in a team, being enthusiastic about what you're working about, and being able to learn''.
> P4 ''Time management skills are also important and most team members were sufficient at this''.
> P37 ''Good communications skills definitely help. Understanding each team member has different experience and skills and those may be different than your own''.
> P103 ''Communication, teamwork, decent coding skills, time management''.

Evidence here add context to the regression outcomes above for RQ4, where it was observed that previous experience in project teams and software deployment skills, and if meetings were conducted in a timely manner, students performed better. Above we see emphasis on both technical skills (e.g., software deployment) and soft skills (e.g., communication and time management). If students were previously exposed to an environment where such skills were acquired, they were able to build on these competencies, outperforming those that were now learning these skills. For instance, previous experience in other team settings seemed to enhance students' ability to manage team meetings in a timely manner.

At the lower end of the scale, Fig. 4 shows that students reported the ability to learn, self-confidence, self-motivation, organisation, leadership and planning skills as useful when joining a software project course.

(ii) **Skills Anticipated:** Students joined the software project course anticipating to develop coding skills the most (refer to Fig. 5). Of the 33 students that provided a response for this question, 15 students (or 45.5%) anticipated this skill. That said, students also relished the opportunity to learn project management (6 students) and planning skills (2 students), testing skills (3 students), deployment skills (2 students) and meeting skills (2 students). Students' comments demonstrated their eagerness to develop diverse skills, notwithstanding coding being most central to their enthusiasm. Students commented that they anticipated to develop:

> P5 ''I think some of the back-end database stuff I anticipated learning, but didn't really learn too much about the stuff others did when they didn't have problems''.

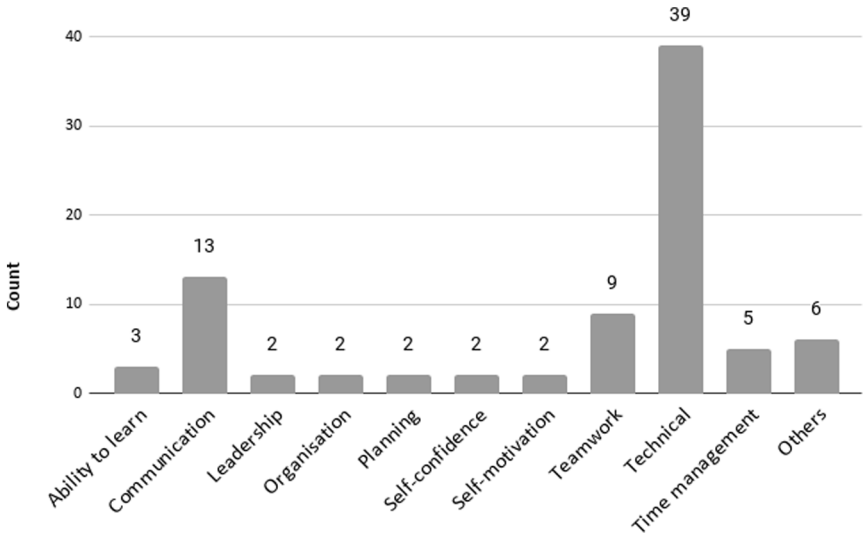

Fig. 4. Skills that are necessary before entering the software projects.

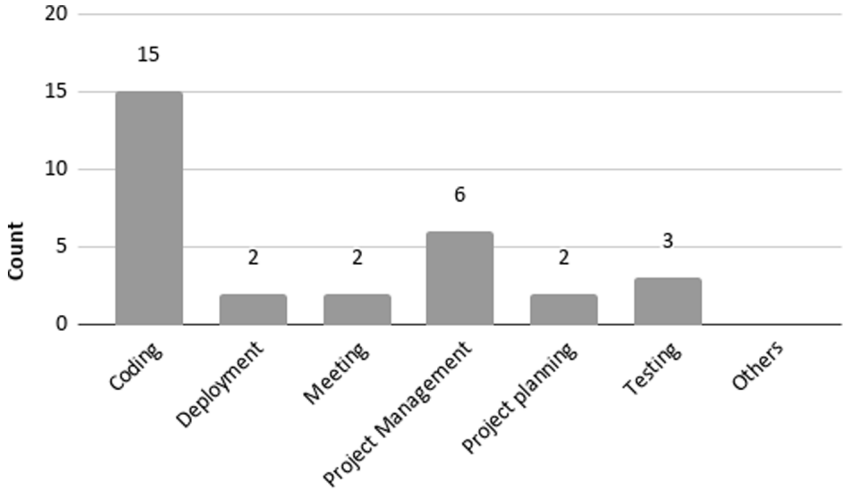

Fig. 5. Skills that were anticipated before entering the software projects.

P16 "Project management, back end development".
P8 "Real world project deployment i.e. complete build/deploy pipeline fully implemented".
P9 "How to handle meetings and presentations - seem like key things for team projects".

(iii) **Skills Learned:** While students anticipated learning certain skills, Fig. 6 shows that by the end of their project they relished specific skills that were learned. Of the 50 comments that were coded, technical skills stood out, where 20 students (i.e., 40%) considered this to be the most important skill they learned. Teamwork (16 students), project management (12 students) and communication (5 students) skills were also mentioned frequently by students. While many of the less frequently mentioned skills (e.g., problem solving, questioning, self-motivation) cut across those that were singled out, most students seemed to join the projects with these skills. Thus, students did not perceive the skills that were least mentioned to be noteworthy in terms of their learning. Students' comments indeed reflect their appreciation for technical, teamwork and project management skills, including the awareness to feed off of each other's variable strengths, the satisfaction of overcoming specific and general coding challenges, and how these accomplishment were achieved under software project management principles and guidelines. Students mentioned:

P4 "The most important skill I learned throughout the project was working with team members whose skills and standards were on a much lower level than mine and resolving issues caused by this to ensure tasks were still completed".
P5 "Creating android apps. Sourcetree especially. Was most proud of adding in the email functionality, wasn't a major technical undertaking but got the buzz you get from successfully coding something that I forgot about".
17 "I wanted to learn how to develop an application which I have. I also learnt how to work better as a team (sharing tasks) as I typically like to work alone".
P92 "Coding experience, something I lacked. Working in teams following the project management steps".

4.6.2. Challenges to success (RQ5.b)

Finally, students were asked to comment on the biggest challenges they faced during the project, where 54 responses were

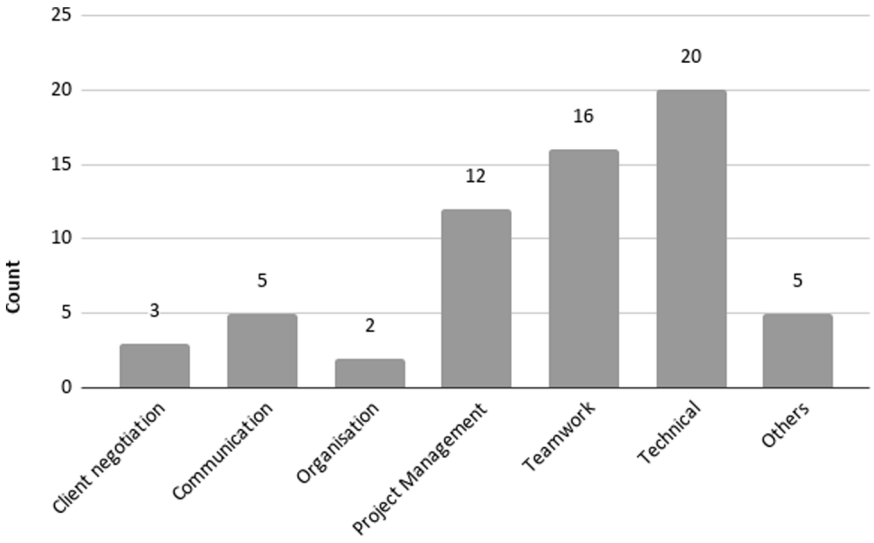

Fig. 6. Most important skills learned during software projects.

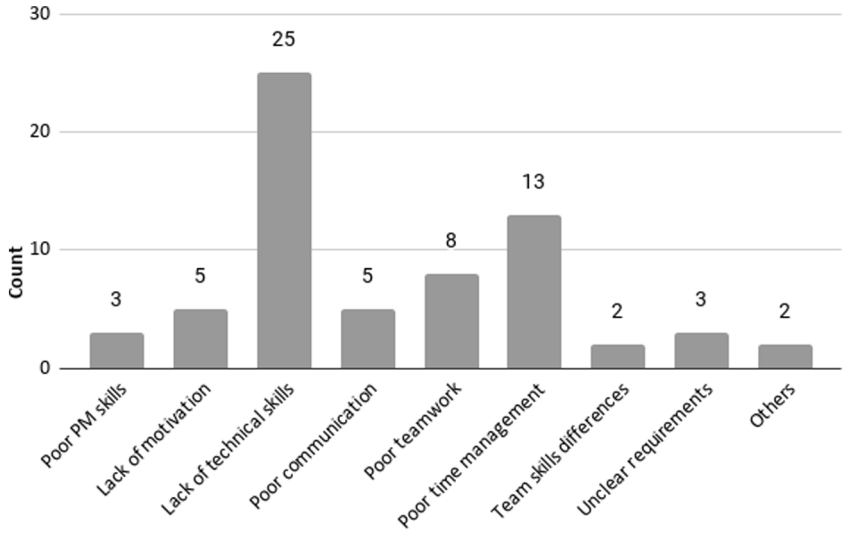

Fig. 7. Challenges to the success of software projects.

provided. Results from these comments are visualised in Fig. 7, showing lack of technical skills (alluded to 25 times or by 46.3% of the students) to be the biggest challenge. In fact, technical skills stood out in all of the four questions (including skills necessary, skills anticipated and skills learned). Poor time management (alluded to 13 times or by 24.1% of the students), poor teamwork (alluded to 8 times or by 14.8% of the students), poor communication (alluded to 5 times or by 9.3% of the students) and lack of motivation (alluded to 5 times or by 9.3% of the students) were also notable mentions (refer to Fig. 7).

Central to the challenges posed by the lack of technical skills was skill disparities, which at times impacted teamwork and stymied progress. For instance, students noted that:

P4 "Working with team members who were lacking in technical skills while still ensuring tasks were completed was the biggest challenge and took significant time and energy".

P6 "One group member took over control of the project and did not ask for other members ideas. When we bought it up he got defencive. The only reason the project continued the way that member wanted, was because they were ridiculously highly skilled compared to the rest of us. They are an exceptionally skilled personal, just their communication ruined the friendliness of the project and turned it into a task/problem, compared to a good project".

P33 "My team was uneven in terms of expertise, there were two strong coders and the rest had very limited coding skills and this proved very difficult both with planning and sharing ideas and with implementation".

Teamwork seems to have also been affected due to technical skills limitations and lack of motivation in the teams, which at times resulted in a stressful team environment due to schedule pressures. This in turn affected positive group dynamics and was the source of conflicts among members. Students mentioned that, the biggest challenge to our project was:

P9 "Teammates procrastinating and failing to meet our team's personal deadlines. Had to pick up some of the slack due to this when the deadline was fast approaching".

P17 "Time management was a bit of a struggle, we had so many ideas but struggled to prioritise which ones we would be able to get done over the course of the project".

P96 "Dealing with severely unmotivated and argumentative teammates who were demoralising the team".

While poor communication was also a problem at times, this largely resulted because of technical skills disparities (noted above), the haste to communicate and language barriers.

P8 "English as a second language" communication".
P37 "Conveying ideas and best practice to the others in the group. Some didn't understand, some I didn't explain clearly enough".

We discuss these findings and those above in response to the other research questions next.

5. Discussion and implications

5.1. How much effort do students commit to software engineering projects, and do students perceive this effort to be adequate?

Our results show that nearly all students felt that they were able to allocate adequate time to their project. However, at times some students felt like they spent too much time on their project (as much as 17 h each week). The results are generally consistent across all universities, and the overall outcomes was that students generally felt that they committed "just enough" time to their projects. That said, students enrolled at one of the universities in our sample committed significantly less effort that those of the other three (i.e., 8.2 h each week on average compared to as much at 17.4 by others), albeit these students operated in reasonably sized teams. One may ask: is it that these students were more efficient software developers or they did not cope with the demands of software engineering? We examine other aspects of these students responses and the adequacy of the outcomes they developed in later subsections.

Overall, however, these are interesting findings as previous studies found that students usually underestimate the time required to complete their projects (Vanhanen et al., 2018; Broman et al., 2012), and thus they end up spending more time completing the project tasks than initially anticipated. This then causes them to feel like there was inequality in the time that was required for their software development efforts and that which they had available. However, the comparison is sensitive to the nature of the setup of the course undertaken by the students, as this could depend on the course's structure or the course credits. For instance, in Europe, each course has a number of credits following the European Credit Transfer and Accumulation System — ECTS. Accordingly, the University of Cyprus course has 7.5 ECTS, which means that students are required to allocate 14–15 h to the course each week. Courses in New Zealand are typically 15–18 points, requiring students to commit around 10–12 h each week. That said, outcomes here may also be related to the estimation competencies of students in this study, and the techniques that were used (e.g., expert estimation — planning poker). Still, the results presented here provide evidence for the effort that is needed to plan, design and deliver successful software engineering projects. Here we see that students should be prepared to spend at least eight hours each week on such courses, and thus, instructors should be aware of their other commitments when scoping projects.

It should be noted that, effort estimation is reported to be a challenge area for software developers in industry. While various expert estimation methods are used to provide some degree of relief, especially when agile methods are used (e.g., Scrum and Xtreme Programming), the accuracy of developers' estimates is still far from perfect (Fernández-Diego et al., 2020). Thus, practices aimed at systematically mentoring students' effort estimation skills could be noteworthy for helping with their post-study competence. For instance, if students are using planning poker for their estimation they should properly observe the rules of the game during estimation and be sure to record actual time spent on tasks, so that they could use project data to incrementally improve their estimation skills.

5.2. RQ2. How Do students perceive their performances during software engineering projects?

Students awarded themselves very high grades for several activities, including: attendance at meetings, activeness on project tasks, solving difficult tasks, taking responsibility for teamwork, taking responsibility for individual work, the quality of the work done, willingness to participate in teamwork, negotiation skills, listening skills and teamwork skills. Similar findings have been reported by Vanhanen et al. (2012) and Bastarrica et al. (2017), who also established that students perceived improvements in their soft skills (e.g., communication skills) over the course of their projects. Students felt that communication between the development team members and clients got better with time. On the other hand, our results show that although students were satisfied with their skills overall, they felt less certain about their project management, technical and presentation skills. This finding somewhat contrasts with other studies. For instance, Vanhanen et al. (2012) showed that students indicated that their technical skills improved after they completed software engineering projects. That said, although our assessment criteria are slightly different to those used in previous study, we observe that students studied in this work seemed to be generally satisfied (based on the high grades they gave themselves) with their overall performance on the projects. Also, as mentioned before, our results suggest that students in general are more satisfied with their team collaboration performance than with their technical contributions. An improvement to the way that technical tasks are managed should be taken into consideration in future projects that students pursue (and more generally). For instance, a technical champion role may be created by teams, where such a member could be responsible for cataloguing relevant tutorials for covering the technical skills that are needed by members of the team (e.g., for developing good unit tests).

That said, our findings here for students' satisfaction with their soft skills is noteworthy and augurs well for their future, particularly if they became agile software developers. Agile methods stress people over processes, and such techniques are used heavily in the software development industry (Licorish et al., 2016). In fact, even where traditional approaches are used, there is evidence that agile practices tend to infiltrate (Klünder et al., 2019). Thus, students perfecting soft skills (e.g., negotiation skills, listening skills and teamwork skills) are likely to transition well into industry-based agile teams.

5.3. RQ3.a How satisfied are students with the skills they acquire and project outcomes they deliver for software engineering courses?

Students were satisfied with the range of skills they gained on their software development project, including technical and project management skills. In previous work that combined undergraduate and graduate students in the same team, project management skills (e.g., leadership and time management) were valued more by the graduate students that acted as project managers in the teams (Kapitsaki and Loizou, 2018). In our case, all team members are undergraduate students that had to perform project management activities within the team, albeit students from one university (i.e., University of Canterbury) were in their second year of study and considered less experienced. However, we did not observe any differences in the outcomes across these

students. In addition, students were largely pleased with the project outcomes they delivered, their project teams and the specific project they were assigned. That said, students that expressed higher satisfaction with their project management skills and project outcomes delivered tended to score higher in the courses.

Our results verify that, overall, students are satisfied with the presence of project-based software engineering courses in the curricula, and they value the skills they obtain via these courses. There is a little room for improvement in the area of team composition and team assignment however. Educators could thus improve overall software engineering course outcomes by being careful with project assignment and members' selection to project teams. This particular issue has been noted in previous work, where recommendations are provided for improving team outcomes (Akbar et al., 2018). For the students studied in our work, those at University of Cyprus were allowed to self-assign their teams and projects, and our evidence shows that these students were generally most satisfied. This outcome somewhat contrasts to teams at University of Canterbury who were assigned to teams, and also studying in their second year, suggesting that they had less experience on team projects and general team work.

5.4. RQ3.b How satisfied are students with their team when working on software engineering courses?

In the consideration of teams' dynamics, performance and atmosphere, students graded team meetings, members' willingness, team atmosphere and communication as excellent. However, team technical competency, knowledge, confirming to meeting agenda, preparedness, grade of work, direction for follow up tasks, division of labour and the quality of outcomes were scored lower. Division of labour or task allocation is a complex issue but an essential element to the student experience. Inequitable task allocation has been recognised in a previous study involving first year students in a project-based engineering course (Fowler and Su, 2018). In that study, students indicated, among other things, that they would avoid tasks they were not confident with, whereas others felt they had no choice in their selection or assignment of tasks. These themes may reflect also in our work, explaining the relatively lower scores given for labour division. The more technically competent members may monopolise critical and challenging tasks on the premise that less skilled members may be incapable of solving such tasks. This in turn may result in the less skilled members not having the opportunity to develop pertinent skills, and thus, thereby contributing less to the project. Labour division in the university differs however, from labour division in the industry, as students need to undertake various roles, e.g. coding, testing, and documentation writing, that are defined in a more precise way in industry and are separated to a great degree. Nevertheless, in the industrial setting a good understanding of the overall system is vital.[2] There is need for follow up work to probe the issue of labour division in software engineering project courses.

In fact, we observe that the more junior students seemed to be less satisfied with many aspects of their teamwork. These students seems to struggle with managing team meetings, members willingness to be involved in the project, team communication and working through meeting agendas. This in turn seems to affect team atmosphere, knowledge sharing, members' preparedness, direction for follow up tasks and division of labour. Here we see that time spent at the university seems to result in students becoming more prepared for the rigours of software engineering project work, and teamwork in general. This may also translate into software development practice, and has implications for post-study team performance. If students exit university without pertinent group work skills we expect that they would underperform in industry, as there is recently a stronger need for team-playing skills in job advertisements, for instance for the case of software testers (Florea and Stray, 2018). Empirical evidence has also shown that various soft skills come to bear during real software development projects (Licorish and MacDonell, 2014), and such skills are used even in environments where there is anticipation that the allocation of tasks dictates otherwise (Licorish and MacDonell, 2013).

When team size and students' willingness to work on future projects with their teams were considered, students largely felt that their teams were just 'the right size' and would be happy to work with their team again. Across all universities, team size ranged from 4 to 7 students, albeit only one of the 35 projects studied had 7 student. In Scrum, agile teams consist of 7 members plus or minus 2, but the project complexity in the university setting is expected to be lower, explaining the students' satisfaction with the team size (Schwaber and Beedle, 2002). Overall, the students' attitude towards their teams was positive, indicating that team creation was performed in an adequate way (albeit self-assigned members at University of Cyprus were most pleased). Team formation is an aspect that needs to be considered in relevant curricula, as it might affect learning outcomes and students' performance. Previous work has shown that self-assigned teams had slightly better performance than teams assigned by instructors (Lø vold et al., 2020). Educators may adopt a blended approach, helping with team assignment where members may find it difficult to self organise, and allowing self-assignment otherwise.

5.5. RQ4. What Variables predict the adequacy of outcomes students develop during software engineering projects?

When examining the variables that predict the quality of the outcomes students delivered, we observed that there were several notable variables that significantly correlate with project score: if students had worked in project teams previously, if students had previous technical skills, if students demonstrated good project management skills, if teams had timely meetings, had good agendas when they met, if members were very prepared, and if tasks delivered were assessed to be of a high grade by the team. Notable in our regression outcomes were predictors for if students had worked in project teams previously, if students had previous technical skills and if teams had timely meetings, which all significantly affected the variance in project score (adequacy of outcomes). While other studies (e.g., Minaei-Bidgoli et al., 2003) explored such variables in the context of shorter assignments and collecting data from e-learning platforms to identify individual students at risk, we identified variables for student projects that span longer time periods. Also, the variables we identified are closely tied to the behaviour and characteristics of students, rather than data collected in an e-learning platform while students work on an assignment. Our findings that timely meetings (as one form of team communication) affects project score confirms the importance of communication in software development practice (see for example early studies on communication in distributed teams Holsti, 1969). Unlike Marques et al. (2017) we did not find that weekly reflections led to higher productivity. Our results can be used to develop assessment, monitoring and coaching mechanisms. For example, Tubino et al. (2020) recommended that software engineering projects should be assessed using portfolio-based assessments. Our variables help make such aspects concrete and tangible. For instance, timely and regular meetings have a positive impact on project score,

[2] https://gcallah.github.io/DevOps/divOfLabor/divOfLaborPaper.html

so instructors could gently "enforce" regular meetings and status reports throughout student projects, which may be part of a communication portfolio. As found by others (e.g., Marques et al., 2017), this could also increase the sense of team belonging and satisfaction. Furthermore, our results can support instructors in the creation of teams. While there are theories on composing teams in software engineering group projects (for example, Oakley et al., 2004) and team formation techniques have been the main subject of many previous research works, as a recent systematic mapping study shows Costa et al., 2020, our results provide concrete implications on the potential behaviour of such teams depending on their technical skills and background.

5.6. RQ5.a How do students perceive software engineering courses in terms of the skills that are necessary, anticipated and learned?

Students expressed that technical skills (e.g., coding, testing and deployment) were most integral for succeeding in a software project course, including knowledge of multiple languages and the software deployment process. Students also highlighted the value of soft skills, comprising communication skills, teamwork skills and time management skills. Indeed, the use of Agile development in industrial settings requires more communication between team members and was one of the challenges in Agile transition, indicating that including such skills in software engineering education may be more vital now than it was one decade ago (Pikkarainen et al., 2009). If students were previously exposed to an environment where such skills were acquired, they were able to build on these competencies, outperforming those that were now learning these skills. The question that follows then is, with increasing project opportunities, do the competencies of students eventually converge such that they eventually become capable software developers with experience?

Students joined the software project course anticipating to develop coding skills the most, however, they also relished the opportunity to learn project management and planning skills, testing skills, deployment skills and meeting skills. This is similar to the findings of others who noted that students were enthusiastic about learning how to use tools that they can use in industry, see Raibulet and Fontana (2018). Also, as others have found (e.g., Abad et al., 2019), these type of skills (social skills and quality-related skills) are typically the ones that students struggle with the most.

Students expressed most appreciation for technical, teamwork, project management and communication skills. In particular, the awareness to feed off of each other's variable strengths, the satisfaction of overcoming specific and general coding challenges, and how these accomplishment were achieved under software project management principles and guidelines were highlighted by students.

5.7. RQ5.b What challenges students' software development project success?

A lack of technical skills was the biggest challenge presented to students in developing their projects, which is an issue encountered in any development environment – industrial or not – when developers are asked to work with new technologies or with technologies they have limited experience on. Poor time management, poor teamwork, poor communication and lack of motivation were also notable mentions. In terms of the lack of technical skills, it was noted that skill disparities impacted teamwork and stymied progress. Procrastination resulted due to technical skills limitations and lack of motivation in the teams, which at times resulted in a stressful team environment due to schedule pressures. This in turn affected positive group dynamics and was the source of conflicts among members. Poor communication was also a problem at times for teams, which largely resulted because of technical skills disparities, the haste to communicate, and communication challenges generally. Information disparity, defined as knowledge and perspective difference, has been identified as a source of conflict for software development teams also in industry settings (Liang et al., 2009).

As stated in Section 1, this work aims to provide an evidence based assessment of success factors and challenges of software engineering capstone projects. Although some of the results reported above may be expected, following such an evidence-based approach helps to confirm what we anticipate and/or believe about factors that may influence the performance of capstone projects, and also provides empirical evidence for such factors (Kitchenham et al., 2004). This approach has been used in the past to confirm what is believed by developers versus actual data obtained directly from software engineering projects. For example, the study of Devanbu et al. (2016) investigated prior beliefs of developers working at Microsoft, and the relationship of these beliefs to actual empirical data and evidence from the projects that those developers have worked on.

In general, our findings here could help assess curricula and course contents against the perception of students. Instructors could then develop teaching methods that help students learn skills and techniques based on their perception. For example, if students lack technical skills and tend to procrastinate, dedicated technical tutorials in an otherwise unstructured project course might help. Similarly, dedicated training of communication skills in typically unstructured and flexible project courses could help mitigate team frustrations due to poor communication. Our assumption is that these steps would improve student experience on software engineering courses, and lead to well-developed junior/novice software development practitioners, whereas soft skills training is usually part of training courses offered by organisations for their employees in any industrial domain.

5.8. Educators' reflections

The courses studied focus heavily on preparing students for software engineering careers, bringing students close to the setting of an industrial software engineering project. Our results indicate that students anticipate different skills acquisition from such courses. In particular, technical skills were at the heart of the skills anticipated by students, and the specific skills that were most enhanced over project duration. While it is also helpful for students to enter project courses with good technical skills, students do not always possess such skills beforehand. This can negatively affect students' development of soft skills (and mainly teamwork and communication) during the project execution, as they can be too preoccupied with the technical aspects. In fact, at times the use of many new technologies can inflate the technical requirements of projects. It does not help also that students can cater to a variety of customers, as well as to a constantly changing environment in terms of programming languages' and frameworks' popularity. These issues are a source of conflict, as our results show. However, addressing this proactively and efficiently is not an easy task. We have suggested some directions to alleviate project issues and support student learning and enjoyment, such as technical tutorials, but it is clear to us that the competing skills (especially technical skills) requirement needs to be handled on a per case basis. For example, project courses may only be available to students in certain degree programmes, or require specific prerequisites. Similar to Jacob and Faily (2019), we also observed that students underestimate the complexity of teamwork and soft skills in group projects. In our experience,

mitigating team problems requires close coaching and advising students along the way, rather than meeting students every couple of weeks. Here, we suggest separating the project process (and progress) from the learning process (and progress) (Gary, 2015). For instance, a student or a team may progress fast on coding activities and deliver functional software, but may not understand how to properly apply design patterns for maintainable code. This also means that educators must be able to change hats, from instructor to coach to project stakeholder, etc. Consequently, this disparity in technical knowledge, team formation, communication and time management are the main challenges that similar courses need to address.

Based on our reflections and the findings from this study, we see several emerging follow-up questions, for example:

- How much time do different students on projects devote to different project tasks, and how does this vary amongst team members or projects? While we did not study this question in detail, we found that the effort devoted to different tasks varied. For example, significant effort was devoted to coding-related activities, but less effort to documentation or testing. However, this could be due to the fact that documentation of source code and automated unit testing may be considered part of coding rather than separate activities (in particular in agile-inspired projects). Coding tasks may also be preferred by some students.
- What does success look like for a student group project? Success can mean different things for different students. For example, at UoC, one of the first activity that students perform as a team is to discuss expectations of the different team members. These expectations vary from achieving a high grade, to improving technical skills, to learning how to work in a team, to creating a reference project that could be used in future job applications. Furthermore, given that all projects required students to deliver a working product, one may ask whether a fully working software at the end of the course would mean success. Given that projects were conducted in an educational setting, we believe that success is not so much defined based on the quality of the final product only, but based on achieving the learning outcomes of a course. For example, learning outcomes could be about understanding and applying good design principles and practices; the ability to recognise and describe the impact of design alternatives on quality; the ability to implement high-quality code from a design using tools, environments, frameworks and existing code; and understanding and applying methods and tools for parallel implementation in a team. Some "non-technical" learning outcomes that define success could be about applying methods for identifying and mitigating software project risks, critically reflecting on individual practices and performance, and the practices and performance of others; understanding complexities of working in a team setting and functioning effectively in team member roles; analysing and solving open technology-based problems through self-directed learning; understanding, planing and documenting all phases of a software development project; as well as presenting (orally and verbally) work to peers and non-technical audiences. The product developed by students is then only one artefact (in addition to logs, reflections, code quality, test quality, etc.) to assess if and to what degree students achieved the learning outcomes.
- What is necessary to deliver — for an external client, or an internal client? We believe that how and what to deliver in a project course also depends on the type of client involved in a project. External clients, such as industry partners may be very much interested in obtaining a useable solution at the end of the project. We have also found that external clients often use course projects as "extended job interviews" to assess the skills of potential job candidates in more detail. On the other hand, this may cause potential conflicts of interest for teaching staff who are to ensure academic integrity and student learning, rather than producing solutions for industry. Internal clients, such as teaching staff, may be more flexible in adjusting their expectations regarding the produced product and may accept non-technical contributions to balance low-quality technical contributions and product quality (see previous point regarding "success" in project courses).
- What kind of project and code metrics are useful to use in project courses? We found a combination of metrics useful to assess student progress and learning. For example, we used traditional metrics such as the lines of code contributed, but also the amount of surviving code, the number of commits, the size of commits, and the time spent on the project and various activities (e.g., do students contribute to a variety of tasks or only to coding or only to documentation or testing). We complement metrics with subjective and personal peer evaluations and reflections of teams and individuals about their achievements and ways of working. Ultimately, however, the end product always attracts the most scrutiny, and especially in light of the overall goal to develop acceptable and useable software.
- How to scope projects? How to scope projects is a challenge for both instructors and students. The projects in software engineering project courses are not meant to the programming assignments, but students are expected to practise the full range of software engineering skills (i.e., requirements related activities, design, testing, coding, documentation, processes and practices) and learn to understand the impact of designs made in projects. Therefore, project descriptions given to students may be open-ended (either intentionally by teaching staff, or unintentionally by industry clients who lack a proper understanding of the problem or potential solution). It is then the responsibility of students to scope the features of their product that are feasible to complete and satisfy potential stakeholders, as well as communicate their reasoning. This may involve research and consultation with teaching staff. On the other hand, this causes the risk of assessing different projects in a course that are not comparable, or exposing students to the risk of feature creep. That said, such challenges serve the purpose of providing useful lessons for the future, including for transitioning into the corporate world.

6. Threats to validity

Notwithstanding the importance of the knowledge that is provided by this study, we concede that there are threats to the study that may affect its generalisability. To provide a deep assessment of these threats we use the framework provided by Stavru (2014), which provides guidelines for evaluating the *thoroughness* and *trustworthiness* of surveys. Thoroughness considers survey definition, design, implementation, execution, and analysis, while trustworthiness accounts for internal, construct and external validity and consistency.

Thoroughness and Reliability: We define the survey in Section 3.1, after clearly establishing the study objectives in Section 1. The survey instrument design was motivated by the IEEE and ACM SEEK and curriculum guidelines, which cover the software engineering graduate expectations of the software development industry. The survey instrument was piloted prior to

implementation, and benefited from students' feedback and those from the wider research team comprising software engineering academics across four universities. Our target population and sample frame were accurately described in Section 3.1, where we recorded responses from 57.7% of the students. Data collection was also done after students had completed the courses, and thus their reflections benefited from currency of their experiences which stake strong claims for reliability. In the execution of the survey, we traversed a rigorous ethical approval process in preserving the rights of students, who were properly introduced to the study before they were allowed to provide their responses. Each section of the survey was properly introduced and analysed, where students were provided and responded to adequate context relevant to the software engineering courses that all confirmed to the IEEE and ACM SEEK and curriculum.

That said, the survey was conducted among students taking software engineering courses at university where they were required to correctly identify the project they were working on. Thus, there is a threat to reliability that students may have answered survey questions in a way that they assume would be pleasing to their instructors. In addition, other factors may impact students' perceptions and responses, and in the process pose a threat to the reliability of the outcomes in the work. For instance, students may find specific instructors more appealing than others, thereby responding more willingly and favourably to their guidance. Given that our study was executed across four universities, and benefited from a response rate of over 57%, we anticipate that such threats would be reduced during our analyses. In addition, we conducted both deductive and inductive analysis in this study, thereby providing triangulation for students' responses across the various sections of the survey.

Trustworthiness and Validity: Formal statistical analyses are likely to mitigate potential measurement errors, however measures were also defined in relation to the SEEK knowledge base (refer to Section 3.2) ensuring internal and construct validity. Our efforts to pilot the instrument, align its development with the established SEEK knowledge base and distribute the instrument for data collection across four universities also support claims for consistency. In the consideration of external validity, given our well-informed study design and the collection and analyses of data from student across four universities, we believe that our outcomes may be generalisable to similar software engineering project courses where the IEEE and ACM SEEK and curriculum guidelines have informed the course design. The SEEK knowledge base also aligns with the Systems Engineering Body of Knowledge (SEBoK) (IEEE, 2020), which provides guidance for the software engineering profession more generally. In particular, our outcomes may generalise to environments that feature incremental and iterative development, weekly stand-ups and retrospections, stringent version control, unit testing and continuous integration, frequent delivery and continuous client feedback, and the use of various software metrics for process improvement.

That said, the survey focuses on the views and perceptions of students, which were complemented by instructors' reflections (as outsiders). The work would benefit from perceptions of the employers of these students, as a third tier of validation. Finally, while we are aware of the requirement of open data for facilitating replication studies, ethical restrictions limit our ability to share the raw data set. However, we would be happy to share any of the statistics.

7. Conclusion and future work

This study explored students' responses in relation to the effort required to successfully deliver software projects, their performance, satisfaction with the skills they develop and project outcomes and teams they collaborate on. The work also investigated the factors that predict the adequacy of outcomes students developed and the skills that are necessary, anticipated, learned and challenges to students' success. Among our findings, we observed that students generally allocated enough time to their projects, albeit at times the effort required was excessive. Students assessed their performance satisfactorily on most project dimensions (e.g., quality of work done), however, they were less satisfied with the development of some skills (e.g., technical skills). While teams were happy overall, self-assigned teams were most positive about their project experiences, and junior students struggled with teamwork. We observed that outcomes students developed were adequate if they had worked in project teams previously, had technical skills and if teams had timely meetings. Furthermore, lack of technical skills was the biggest challenge to project success. Outcomes here suggest that estimation skills should be singled out for prolonged attention. Strategies may also be implemented to encourage the development and management of technical skills and team formation involving students. Furthermore, software engineering courses should offer teams the opportunity at repeated teamwork exposure and instill the value of timely team meetings. We encourage replication studies to further understand the variables that predict software project success. In particular, follow up work is needed to investigate the division of labour among teams, and to further consider our open questions in Section 5.8.

CRediT authorship contribution statement

Sherlock A. Licorish: Conceptualization, Literature review, Methodology, Data curation and results, Discussion, Writing - original draft, Writing - review & editing. **Matthias Galster:** Literature review, Methodology, Results, Discussion, Writing - review & editing. **Georgia M. Kapitsaki:** Literature review, Methodology, Discussion, Writing - review & editing. **Amjed Tahir:** Literature review, Methodology, Discussion, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abad, Z.S.H., Bano, M., Zowghi, D., 2019. How much authenticity can be achieved in software engineering project based courses? In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). IEEE, pp. 208–219.

Ahtee, T., Poranen, T., 2009. Risks in students' software projects. In: 2009 22nd Conference on Software Engineering Education and Training. IEEE, pp. 154–157.

Akbar, S., Gehringer, E., Hu, Z., 2018. Poster: Improving formation of student teams: A clustering approach. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion). IEEE, pp. 147–148.

Akdur, D., 2019. The design of a survey on bridging the gap between software industry expectations and academia. In: 2019 8th Mediterranean Conference on Embedded Computing (MECO). IEEE, pp. 1–5.

Bastarrica, M.C., Perovich, D., Samary, M.M., 2017. What can students get from a software engineering capstone course? In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET). IEEE, pp. 137–145.

Boldyreva, E., Penskoi, A., Platunov, A., 2020. Formalization and evolution of learning path in embedded systems. In: 2020 Wave Electronics and Its Application in Information and Telecommunication Systems (WECONF). IEEE, pp. 1–6.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101.

Broman, D., Sandahl, K., Baker, M.A., 2012. The company approach to software engineering project courses. IEEE Trans. Educ. 55 (4), 445–452.
Cohen, J., 2013. Statistical Power Analysis for the Behavioral Sciences. Academic Press.
Costa, A., Ramos, F., Perkusich, M., Dantas, E., Dilorenzo, E., Chagas, F., Meireles, A., Albuquerque, D., Silva, L., Almeida, H., Perkusich, A., 2020. Team formation in software engineering: A systematic mapping study. IEEE Access 8, 145687–145712. http://dx.doi.org/10.1109/ACCESS.2020.3015017.
Devanbu, P., Zimmermann, T., Bird, C., 2016. Belief & evidence in empirical software engineering. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, pp. 108–119.
Dunlap, J.C., 2005. Problem-based learning and self-efficacy: How a capstone course prepares students for a profession. Edu. Technol. Res. Dev. 53 (1), 65–83.
Exter, M.E., Ashby, I., 2019. Preparing today's educational software developers: voices from the field. J. Comput. Higher Educ. 31 (3), 472–494.
Fernández-Diego, M., Méndez, E.R., González-Ladrón-De-Guevara, F., Abrahão, S., Insfran, E., 2020. An update on effort estimation in agile software development: A systematic literature review. IEEE Access 8, 166768–166800. http://dx.doi.org/10.1109/ACCESS.2020.3021664.
Florea, R., Stray, V., 2018. Software tester, we want to hire you! an analysis of the demand for soft skills. In: International Conference on Agile Software Development. Springer, pp. 54–67.
Fowler, R.R., Su, M.P., 2018. Gendered risks of team-based learning: A model of inequitable task allocation in project-based learning. IEEE Trans. Educ. 61 (4), 312–318.
Gary, K., 2015. Project-based learning. IEEE Comput. 48, 98–100.
Groeneveld, W., Vennekens, J., Aerts, K., 2019. Software engineering education beyond the technical: A systematic literature review. In: Proceedings of the 47th SEFI Conference 2019; 2019. SEFI-European Society for Engineering Education; Brussels.
Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., Tatham, R.L., et al., 1998. Multivariate Data Analysis, vol. 5. Prentice hall Upper Saddle River, NJ.
Holsti, O.R., 1969. Content Analysis for the Social Sciences and Humanities. Reading. MA: Addison-Wesley (Content Analysis).
IEEE, 2020. Guide to the software engineering body of knowledge (SWEBOK): Version 2.2. Technical report.
Jacob, C., Faily, S., 2019. Exploring the gap between the student expectations and the reality of teamwork in undergraduate software engineering group projects. J. Syst. Softw. 157, 1–18.
Kapitsaki, G.M., Loizou, S.K., 2018. Bringing together undergraduate and postgraduate students in software engineering team project: experiences and lessons. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, pp. 320–325.
Khakurel, J., Porras, J., 2020. The effect of real-world capstone project in an acquisition of soft skills among software engineering students. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T). IEEE, pp. 1–9.
Kitchenham, B.A., Dyba, T., Jorgensen, M., 2004. Evidence-based software engineering. In: Proceedings. 26th International Conference on Software Engineering. IEEE, pp. 273–281.
Klünder, J., Hebig, R., Tell, P., Kuhrmann, M., Nakatumba-Nabende, J., Heldal, R., Krusche, S., Fazal-Baqaie, M., Felderer, M., Bocco, M.F.G., et al., 2019. Catching up with method and process practice: An industry-informed baseline for researchers. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, pp. 255–264.
Kock, N., Lynn, G., 2012. Lateral collinearity and misleading results in variance-based SEM: An illustration and recommendations. J. Assoc. Inform. Syst. 13 (7).
Kruchten, P., Nord, R., Ozkaya, I., 2019. Managing Technical Debt: Reducing Friction in Software Development. Addison-Wesley Professional.
Kruskal, W.H., 1952. Use of ranks in one-criterion variance analysis. J. Amer. Statist. Assoc. 47 (260), 583—621.
Liang, T.-P., Jiang, J., Klein, G.S., Liu, J.Y.-C., 2009. Software quality as influenced by informational diversity, task conflict, and learning in project teams. IEEE Trans. Eng. Manage. 57 (3), 477–487.
Licorish, S.A., Holvitie, J., Hyrynsalmi, S., Leppänen, V., Spínola, R.O., Mendes, T.S., MacDonell, S.G., Buchan, J., 2016. Adoption and suitability of software development methods and practices. In: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC). IEEE, pp. 369–372.
Licorish, S.A., MacDonell, S.G., 2013. The true role of active communicators: an empirical study of Jazz core developers. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering, pp. 228–239.
Licorish, S.A., MacDonell, S.G., 2014. Understanding the attitudes, knowledge sharing behaviors and task performance of core developers: A longitudinal study. Inf. Softw. Technol. 56 (12), 1578–1596.
Lø vold, H.H., Lindsjø rn, Y., Stray, V., 2020. Forming and assessing student teams in software engineering courses. In: International Conference on Agile Software Development. Springer, pp. 298–306.
Magana, A.J., Seah, Y.Y., Thomas, P., 2019. Fostering cooperative learning with Scrum in a semi-capstone systems analysis and design course. J. Inform. Syst. Edu. 29 (2), 4.
Marmorstein, R., 2011. Open source contribution as an effective software engineering class project. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, pp. 268–272.
Marques, M., Ochoa, S.F., Bastarrica, M.C., Gutierrez, F.J., 2017. Enhancing the student learning experience in software engineering project courses. IEEE Trans. Educ. 61 (1), 63–73.
Melnik, G., Maurer, F., 2005. A cross-program investigation of students' perceptions of agile methods. In: Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005. IEEE, pp. 481–488.
Minaei-Bidgoli, B., Kashy, D.A., Kortemeyer, G., Punch, W.F., 2003. Predicting student performance: an application of data mining methods with an educational web-based system. In: 33rd Annual Frontiers in Education, 2003. FIE 2003, vol. 1. IEEE, T2A–13.
Moore, M., Potts, C., 1994. Learning by doing: Goals and experiences of two software engineering project courses. In: Conference on Software Engineering Education. Springer, pp. 151–164.
Oakley, B., Felder, R.M., Brent, R., Elhajj, I., 2004. Turning student groups into effective teams. J. Stud. Cent. Learn. 2 (1), 9–34.
Paasivaara, M., Vanhanen, J., Lassenius, C., 2019. Collaborating with industrial customers in a capstone project course: The customers' perspective. In: IEEE/ACM 41th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).
Paasivaara, M., Voda, D., Heikkilä, V.T., Vanhanen, J., Lassenius, C., 2018. 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). IEEE, pp. 49–57.
Patton, M.Q., 1990. Qualitative Evaluation and Research Methods. SAGE Publications, inc.
Pikkarainen, M., Conboy, K., Karlström, D., Still, J., Kerievsky, J., 2009. What skills do we really need in agile software development? – discussion of industrial impacts and challenges. In: Abrahamsson, P., Marchesi, M., Maurer, F. (Eds.), Agile Processes in Software Engineering and Extreme Programming. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 267–270.
Radermacher, A., Walia, G., 2013. Gaps between industry expectations and the abilities of graduates. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, pp. 525–530.
Radermacher, A., Walia, G., Knudson, D., 2014. Investigating the skill gap between graduating students and industry expectations. In: Companion Proceedings of the 36th International Conference on Software Engineering, pp. 291–300.
Raibulet, C., Fontana, F.A., 2018. Collaborative and teamwork software development in an undergraduate software engineering course. J. Syst. Softw. 144, 409–422.
Ryan, K., 2020. We should teach our students what industry doesn't want. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training, pp. 103–106.
Sanders, D., 2002. Student perceptions of the suitability of extreme and pair programming. Extreme Programm. Perspect. 168–174.
Schneiderm, J.-G., Eklund, P., Lee, K., Chen, F., Cain, A., Abdelrazek, M., 2020. Adopting industry agile practices in large-scale capstone education. In: IEEE/ACM 42th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).
Schwaber, K., Beedle, M., 2002. Agile Software Development with Scrum, vol. 1. Prentice Hall Upper Saddle River.
Silvestre, L., Ochoa, S.F., Marques, M., 2015. Understanding the design of software development teams for academic scenarios. In: 2015 34th International Conference of the Chilean Computer Science Society (SCCC). IEEE, pp. 1–6.
Stavru, S., 2014. A critical examination of recent industrial surveys on agile method usage. J. Syst. Softw. 94, 87–97.
T.J.T.F. on Computing Curricula, 2014. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical Report, Association for Computing Machinery.
Tubino, L., Cain, A., Jean-Guy, S., Thiruvady, D., Fernando, N., 2020. Authentic individual assessment for team-based software engineering projects. In: IEEE/ACM 42th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).
Vanhanen, J., Lehtinen, T.O., Lassenius, C., 2012. Teaching real-world software engineering through a capstone project course with industrial customers. In: 2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex), IEEE, pp. 29–32.
Vanhanen, J., Lehtinen, T.O., Lassenius, C., 2018. Software engineering problems and their relationship to perceived learning and customer satisfaction on a software capstone project. J. Syst. Softw. 137, 50–66.
Vartianen, T., 2019. Moral problems perceived by industry in collaboration with a student group: balancing between beneficial objectives and upholding relations. J. Inform. Syst. Edu. 20 (1), 7.

Vogt, W.P., Johnson, B., 2011. Dictionary of Statistics & Methodology: A Nontechnical Guide for the Social Sciences. Sage.
Wehrs, W., 2020. An assessment of the effectiveness of cooperative learning in introductory information systems. J. Inform. Syst. Edu. 13 (1), 6.
Zolduoarrati, E., Mohammadi, R., Lotter, A., Alessi, N., Michael, K., Licorish, S.A., 2017. Reflections on the use of Agile practices and associated tools in university settings for an Android project. In: Proceedings of the 8th Annual Conference of Computing and Information Technology Research and Education New Zealand(CITRENZ), Retrieved from http://www.citrenz.ac.nz/conferences/2017/pdf/2017CITRENZ_1_Licorish_Agile.pdf.

Sherlock Licorish is a Senior Lecturer in the Department of Information Science at University of Otago, New Zealand. He was awarded his Ph.D. by Auckland University of Technology (AUT), and joined University of Otago in 2014. His research portfolio covers agile methodologies, practices and processes, teams and human factors, human computer interaction, research methods and techniques, software code quality, static analysis tools, machine learning applications and software analytics. Sherlock occupies several service roles across the university, nationally and internationally.

Matthias Galster is an Associate Professor in the Department of Computer Science and Software Engineering at the University of Canterbury in Christchurch, New Zealand.

Georgia M. Kapitsaki is an Associate Professor in the Department of Computer Science at the University of Cyprus. She received her Ph.D. from the National Technical University of Athens, Greece (2009). Her research interests include software engineering, open source software, privacy enhancing technologies, and context-awareness. She has published over 70 papers in international conferences and journals. She has been involved in EU projects and has worked as a software engineer in the industry.

Amjed Tahir is a Senior Lecturer in Software Engineering at Massey University, New Zealand. He obtained a Ph.D. degree from the University of Otago, New Zealand, in 2016. Amjed research is mainly in empirical software engineering, with special interest in software maintenance and testing problems. He has served in the organising and program committee of a several software engineering conferences. Amjed is an Executive member of Software Innovation New Zealand.