2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)

Adopting Industry Agile Practices in Large-scale Capstone Education

Jean-Guy Schneider, Peter W. Eklund, Kevin Lee, Feifei Chen, Andrew Cain, Mohamed Abdelrazek
School of Information Technology
Deakin University, Geelong
Geelong, Victoria
{jeanguy.schneider,peter.eklund,kevin.lee,feifei.chen,andrew.cain,mohamed.abdelrazek}@deakin.edu.au

ABSTRACT

This paper presents the practice and experience in adopting an agile organizational model for a final-year capstone program in Software Engineering. The model developed is motivated by having real (and developing) software artifacts with incrementally changing team members working on a product-line. This in turn results in more sophisticated capstone student-project outcomes. The model proposed supports student mentoring and promotes, through its internal organization, leadership and personal responsibility. The students are supported by professional software engineers, up-skilling workshops, and academic supervisors who act as a personalized reporting and grading point for the team. The academic supervisors are themselves supported by a tribe leader, a faculty member who assumes overall responsibility for a product-line, and who acts as a report to an external industry client/sponsor. This paper describes the motivation for the capstone model, its adoption, and some preliminary observations.

CCS CONCEPTS

• **Social and professional topics** → **Model curricula**; • **Software and its engineering** → *Programming teams.*

KEYWORDS

Software engineering education, agile software development, capstone education

ACM Reference Format:
Jean-Guy Schneider, Peter W. Eklund, Kevin Lee, Feifei Chen, Andrew Cain, Mohamed Abdelrazek. 2020. Adopting Industry Agile Practices in Large-scale Capstone Education. In *Software Engineering Education and Training (ICSE-SEET'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3377814.3381715

1 INTRODUCTION

To better prepare students for the life-long learning required in contemporary workplaces, and help them transition from an educational environment to a professional workplace, educational institutions are focusing on project-based learning in teams. This activity is mainly focused in the latter years of both Bachelors and Masters degrees.

In order to give students an as authentic learning experience as possible, many software-focused degrees are adopting agile approaches in their team-based capstone projects (as discussed in more detail in Section 3). While there are many reasons for adopting an agile approach to Software Engineering education, this is not a trivial undertaking and a number of constraints need to be taken into account.

To be successful with agile requires devolved collaborative decision-making with self-organising teams [7], something students in a Higher Education setting adapt to with difficulty [34]. To create software using a sustained, iterative way, a team needs to have a stable base to work from, something that takes time to establish [16] and may not always be achievable in the short time-frames in which educational projects operate [14]. A capstone Software Engineering experience should also satisfy the graduate learning outcomes for an ICT-degree, which should in turn, if properly designed, reflect industry expectations on what ICT-graduates should "look like" when they complete their education.

An additional challenge in many growing ICT-departments (such as our own) is a significantly increased student cohort that transitions into capstone projects, requiring the capstone project organisation to exhibit scalability without diminishing the quality of the experience or the student outcomes, respectively. This can be particularly challenging if the capstone experience relies on the availability of suitable projects from external (*i.e.* industry) stakeholders.

To address these challenges, we propose a novel approach, called ACE, a model for managing capstone projects in education by (i) adapting Spotify's squads and tribes [2] model – a model designed to deal with multiple teams in a product development organisation and the ability to manage agile with hundreds of team members [13] – to an educational setting, (ii) introduce learning outcomes that define the outcomes we specify Software Engineering students need to demonstrate on completion of their capstone project, (iii) proposing a teaching period timeline that provides students with a guiding framework to succeed in an agile environment. Throughout the rest of this paper, we motivate and illustrate the key elements of the model in detail and report on the evaluation of the initial implementation in a tertiary setting. Although the model is still evolving, our experiences show that the students are receiving a capstone experience that is a more realistic match to industry practice than comparable Software Engineering capstone models.

ICSE-SEET'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7124-7/20/05…$15.00
https://doi.org/10.1145/3377814.3381715

The remainder of this paper is organized as follows: Section 2 describes the motivation for the work; Section 3 presents a literature review on approaches to capstone projects in tertiary education; Section 4 proposes a new organisational model to group project-based capstone projects; Section 5 presents an evaluation of the proposed approach in the context of higher education; Section 6 discusses the outcomes of the work; Finally, Section 7 summarizes the main findings and presents future work.

2 MOTIVATION

Project-based learning is a proven way of achieving the critical graduate learning outcomes of communication, problem solving and teamwork. Group project-based capstone units[1] allow students to develop valuable inter-personal and project management skills. In the computing sciences, it also allows students to work on solving interesting and relevant real-world problems. Students often cite their capstone experience as the most rewarding, challenging and memorable of their degree[2] [24]. Therefore, it is important to effectively resource and support capstone projects. This is challenging with the commonly large cohorts in Australian computing degrees (and likely elsewhere) and pressures on academic staffing, respectively.

There are other constraints to consider as well. A capstone Software Engineering experience should satisfy the graduate learning outcomes for an ICT-degree, which should in turn, if properly designed, reflect well industry expectations of what an ICT-graduate should "look like" when he/she completes his/her education. The capstone organisation and delivery can therefore not drift too far from the defined graduate learning outcomes from the degree it supports.

The following introduces some high-level goals for supporting successful capstone units from experience – summarized here.

Motivation 1: Industry relevant experience
Motivation 2: Authentic learning experience
Motivation 3: Continuity of learning experience
Motivation 4: A successful client experience
Motivation 5: Scalability of approach
Motivation 6: Effective support of students

There is a need for students to have an **Industry relevant experience** which allows them to work on a project that has obvious societal benefits. Evidence shows that capstone projects with an industry focus benefit both the student learning experience and the project outcome [5, 24].

Typically, ICT professionals do not act in isolation, so it is important that students have a **Authentic learning experience** in a group project setting. This means recognising that software developers spend most of their time working on projects modifying others' code and using domain-specific tools. Embedding these characteristics in the capstone design enables a more sustainable model of delivery [27]. Furthermore, the artefact, project, theme, and/or idea that the students work on should be "realistic" and exhibit sufficient complexity and detail to allow the students to respond to it with work packages that are suited to the effort expected from each team member. Student capstone projects that start from scratch every teaching period rarely exhibit the complexity to generate the hours-of-work required. The capstone organisation should be continuous and sustainable, this to support complex outcomes that represent artefacts/projects/themes or ideas that continue from one teaching period to the next.

Due to external pressures, students are increasingly undertaking university studies in flexible ways (part-time, full-time, online, distance), and driving their own learning experience. It is common to split the capstone experience into components to support such flexibility – with individual students taking different pathways from their peers. This means that students frequently do not take capstone units in consecutive teaching periods.[3] It is therefore vital that any approach to capstone ensures a **Continuity of learning experience and supervision**. The ideal situation is for clients to offer long-term projects, for supervisors to oversee the same projects each teaching period, and for students to work on the same project within their capstone experience, respectively.

When designing an approach to capstone, it is important to take into account the needs of all stakeholders [41]. In particular, **A successful client experience** ensures continuity – an unhappy client is unwilling to be involved in future projects. For a computing capstone client, success is generally measured in terms of the software product produced, and clients often also have some short- or long-term recruitment goals. Not all student group projects can result in successful software prototypes, so a desirable characteristic of capstone projects is the possibility of multiple groups working on the same project in parallel. In this way, the client can take the best of the student solutions and build on these in future iterations.

To effectively support capstone projects with large cohorts of hundreds (to thousands) of students, it is important to have the **Scalability of approach** as a consideration when designing the capstone experience. There are particular challenges with large cohorts in capstone due to having to: (i) form and manage groups; (ii) gain and support clients; (iii) support the effective supervision of groups; and (iv) fairly assess and moderate the work achieved. With generally increasing numbers, ensuring a consistent experience is also challenging.

There are increasing pressures on University students that can have an impact on their studies. There is also an increasing number of students in employment during their studies, increasing the time pressure. Capstone projects can place additional stress on students due to their increased (actual or perceived) value and the additional pressure of working with others in a team. It is therefore important to have a system in place to allow the **Effective support of students**. This is especially true for online or distance-learning students who may have compounded difficulties when communicating as groups.

The following section will examine approaches to achieving these goals for group project-based capstone projects.

[1] Components of a degree include modules, subjects, units of competency or units, the completion of which leads to the award of a degree. In our a case study a "unit" represents 1/4 of full-time study or 120 hours of effort, involving 1-hour a week contact with the squad supervisor and up to 4-hours a week with other squad members.
[2] A.k.a. programs or courses, we will use degree hereafter for consistency.

[3] We use the generic term "teaching period" to represent a 12-week teaching period or semester.

3 LITERATURE REVIEW

Group-based project management in higher education is mostly exercised as capstone units at both undergraduate (UG) [19, 32] and post-graduate (PG) [20] coursework degrees. The capstone units are increasingly widespread around the globe, as a result of both accreditation expectations, and strong consensus among educator and industry stakeholders that students need hands-on practical project experiences [32]. In capstone units students typically demonstrate their understanding of collaboration and community as part of a team-based project. These projects, involving real-world experiences, are a form of authentic learning which allows students to turn information into applied knowledge [25].

Meier *et al.* [21] reported experience with group-based project work for large teams in Web development. Experiences with game development in a capstone project is reported in [37]. However, the capstone units we support are very diverse, with many groups developing projects that can be cyber-physical systems or computer games, alongside projects that follow a more traditional software development life-cycle, such as Web apps. Therefore, our approach needs to accommodate the diversity of ICT-skills it supports, and reflect the industry relevance of the projects on offer.

Furthermore, in recent times, agile methodologies have become increasingly popular in the ICT-industry because of improved project performance and rapid development time [8]. In addition, research has shown that adopting agile methodologies improves management of the development process and decreases the amount of overtime, by actively involving the client throughout the project life-cycle, thus increasing client satisfaction [11]. There is a certain inevitability that the ICT-development methodology followed should be a variant of agile [18]. The organisation and assessment of capstone units, therefore, needs to accommodate the transition from waterfall to agile [12], as a reflection of the dominant methodology in contemporary Software Engineering practice.

Agile methodologies (*e.g.*, Extreme Programming (XP) [3, 17], Scrum [36], Feature-driven Development (FDD) [26], Dynamic System Development Method (DSDM) [39], Crystal Clear [6] and others [1]), have been widely adopted in ICT capstone unit teaching in the past by higher education institutions. An embedded system design capstone course offered for Computer Engineering Technology students is reported in [22, 23]. Poženel [28] reports experiences in a software engineering capstone course requiring students to follow the Scrum methodology. Fan [9] discusses the lessons learned in implementing Software Engineering capstone courses, focusing on project management.

Knudson and Radermacher [15] present how they integrate agile practices and principals in a computer science and Software Engineering capstone. Mahnic [19] describes an undergraduate capstone unit in Software Engineering using Scrum for the first time, focusing on students' estimation and planning skills. Rover *et al.* [31] documents the use of scrum for a two-semester senior design project by profiling the experience of the student design team, customer, and faculty mentor. Rico and Sayani [30] reported a capstone unit in which students were able to choose any agile methodology for their project. A pedagogical approach was reported in [40], in which the authors proposed a hybrid course of Information Systems based on XP, Scrum and FDD. Other reports on the introduction of agile methodologies in team-based projects include work presented by Hedin *et al.* [10], Schneider *et al.* [35], or Williams *et al.* [43] (just to name a few).

However, none of the above-mentioned approaches suits capstone projects with large cohorts of hundreds (to thousands) of students due to the lack of scalability. The approach proposed by Stansbury *et al.* [38] uses agile methodologies and tools to better support larger and more multi-disciplinary teams for Computer Engineering and Software Engineering capstone units. Each large capstone team (12 to 16 students) was divided to four sub-teams and each sub-team worked on different features of a complex project. However, similar to other approaches discussed above, this approach does not support continuity of learning experience and supervision, as the students were required to complete short-term capstone projects in the same academic year.

In order to address the above-mentioned limitations of existing models, a new model is implemented for the organisation of capstone units. It is a hybrid, based on Spotify's squad and tribes [2], as Spotify's main purpose is to deal with multiple teams in a product development organisation, focusing on the ability to manage agile with hundreds of team members [13]. Our model is described in the section that follows. The model has been applied at both UG and PG capstone project units; however for the purposes of this paper, we report only on the PG capstone units offered to Masters level students[4] because these units implement individual assessment using DoubtFire,[5] a web-based learning management system, that provides students with a task-oriented approach to assessment by portfolio development (capstone assessment is reported in a companion paper [42]).

4 ACE: AGILE CAPSTONE IN EDUCATION

In this section, we present a new approach to managing the group project-based capstone experience. We first introduce the learning outcomes we desire students to demonstrate at the completion of their capstone experience. We then present the structure of, and the rationale behind, the new model. This is illustrated with an associated teaching period timeline that provides students with a guiding framework to succeed in an agile environment in general and, more specifically, in the context of the proposed model. Finally, we discuss the support structure we put into place in order for each of the elements to work together effectively and link the elements of the model back to the motivations introduced in Section 2.

[4] A Masters degree in Australia is a qualification that meets the criteria of the Level 9 of the AQF (Australian qualification framework) see https://www.aqf.edu.au. In the Bologna process this is equivalent to 120 ECTS in the 2nd cycle of study. There is no federal standard for Masters degrees in the US, rather federal accreditation legislation, "The Higher Education Act" expired in 2013, and is currently before Congress, see Council for Higher Education Accreditation https://www.chea.org

[5] https://doubtfire-lms.github.io/doubtfire.io/

(1) *Apply professional practice, including mentoring of team members, active and consistent participation, effective communication, contribution of technical expertise, and adherence to ethical codes of conduct as a member of an Software Engineering project team.*
(2) *Analyse stakeholders' needs, project goals, and team capabilities to identify, prioritize, and schedule project deliverables, and take responsibility for the execution of the project plan (usually a feature-based plan for the current sprint) and associated reporting.*
(3) *Given a project plan and deliverables, identify, select, justify, and apply contemporary Software Engineering methods, tools and techniques to meet project outcomes, and provide guidance to team members on effective application of these.*
(4) *Showcase personal skills developed, including areas for future learning, and key project outcomes, including lessons learnt and recommended future directions, to specialist and non-specialist audiences.*
(5) *Reflect on, and take responsibility for their own learning, effectively manage their own time and processes, and provide mentoring to members of the project team as a means of managing continuing professional development.*

Table 1: Unit Learning Outcomes for second capstone project unit.

4.1 Unit Learning Outcomes

The cornerstone of model is the introduction of well-defined *Unit Learning Outcomes* (ULOs)[6] that define the outcomes we want students to demonstrate at the completion of their capstone project experience. The specifics of the ULOs are guided by the principle of having students solve real-world, ill-structured problems, improving students outcomes on conceptual understanding, problem-solving and meta-cognitive skills [29], as well as giving them insights into technology, product development, project-based learning, and teamwork [5]. We were also guided by accepted professional Software Engineering practices as well as the desire to enable our students as life-long learners.

At Deakin, the students of all of our ICT-degrees (including Software Engineering students) are required to complete two interlinked capstone project units. The Unit Learning Outcomes for the second capstone project unit are given in Table 1. The ULOs for the first capstone unit are similar, but with less emphasis on *leading* and more emphasis on *learning* what a team-based project work is. Both the proposed project model and the teaching period timeline we introduce in the next two sections, combined with suitable assessments, enable students to demonstrate achievement of these learning outcomes after successfully completing the second capstone unit.

[6]Unit Learning Outcomes are comparable the *Knowledge Units* in the "Computing Curriculum – Software Engieering", *cf.* http://sites.computer.org/ccse/know/FinalDraft.pdf

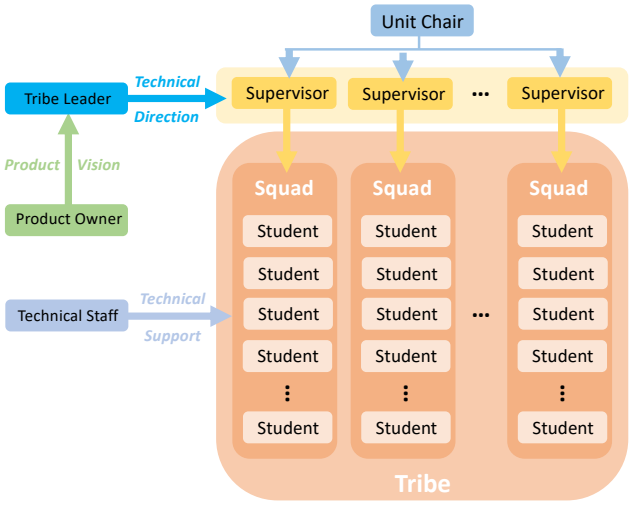

Figure 1: Tribes and squads of proposed capstone model.

4.2 Overview of Model

Figure 1 illustrates the key components of our capstone project model. The model is inspired by Spotify's "Tribes and Squads" [2] but adjusted to an educational setting.

The overarching building block of the model is a **Product Tribe** where a large, *long-term* product undergoes *enhancements* over a number of teaching periods. The vision of the product tribe is set by a **Product Owner** – an industry representative – consistent with the approach of Schneider and Johnston [33] – that brings a non-academic perspective to the project setting – and supported by a **Tribe Leader** (an academic staff member with a specific interest in the project) who translates the vision into manageable *work packages* for each teaching period. Within a product tribe there are a number of **Squads** (*i.e.* teams of students) that are responsible to deliver a set of work packages over the duration of a teaching period. Based on what each squad manages to deliver, the vision of the product tribe is reviewed in-between teaching periods and a new set of work packages are identified.

The Tribe Leader also acts as a "gateweay" between students and the Product Owner and streamlines any communication required, to clarify scope of work packages, requests for feedback etc. By doing so, the Tribe Leader shields the external Product Owner from the challenges students face with externally-facing communication, such as relevancy or frequency, and hence improves the experience of Product Owner (one of the key goals stated in Section 2). Communication channels between Tribe Leader and students vary, often the Tribe Leader communicates with the supervisor who in turn relays advice. In other tribes, the Tribe Leader meets frequently with the students. Squads and their supervisor are organised around Microsoft Teams groupware and communication between students and supervisor is both written and in person. There were 29 supervisors in the 2nd teaching period in 2019 for the two PG capstone units, and 16 unique tribes.

Although in our setting students are required to complete two capstone project units, they do not have to enroll in these units

in consecutive teaching periods. In order to minimize effort in inducting students into a product tribe, but still allowing them the flexibility of a gap between the two capstone project units, squads are comprised of (a roughly equal number of) students from *both* capstone units: we refer to students in the first capstone unit as *Junior* students whereas students in their second unit are *Senior* students.[7] The Senior students are mainly responsible to "drive" the squad, whereas the Junior students are there to learn about the product tribe, and increasingly support the senior students in working on deliverables.

During the first two weeks of a teaching period, Junior students express their preferences for particular project tribes, but not a specific squad within these tribes. Once allocated to a product tribe, a student remains in their product tribe across both capstone units, giving them the necessary continuity, even if they choose to have a gap between the two units. However, the squads of a product tribe are reset at the start of each teaching period and Senior students do not necessarily work with the same peers as before. This gives students a team-work experience similar to many real-world workplaces, where the composition of work teams changes routinely.

Based on past experience "experimenting" with the size of project teams, we opted for large(r) squads of 10 to 12 students. This is a little larger than the 7±2 recommend in the Agile Manifesto [4] but not unheard of depending on the agile technique [36]. In the case of student squads, this size allows us to maintain squad viability even if there is shrinkage.[8] This also allows the students to work on larger, more complex work packages and gives them a sense of achievement if they manage to contribute to deliverables of a scale and/or complexity typical in industrial settings. The size of the squad is also more in-line with what happens in industry, and provides enough capacity to engage junior students more gradually (and not insisting they be "productive" from the get go). It also mitigates the risk that 1 or 2 students pull-off the entire set of deliverables in an "all-nighter" or two, and where the rest of the squad mops up and writes documentation.

Each squad has a dedicated **Project Supervisor** (generally an academic staff member) who oversees the students' work, mentors them in project activities, engages in weekly squad meetings, and serves as a first point of contact if/when problems arise within the squad. As with the students, we aim to have project supervisors stay within the same product tribe from teaching period to teaching period in order to reduce their workload in familiarizing themselves with the context of their product tribe.

Finally, two **Unit Chairs** with overall responsibility for grading and the coordination and collection of assessment (one for each of the two project units) oversee all activities during a teaching period. They are also responsible to moderate expectations across squads in order to ensure that all students get an equal learning experience and are assessed as fairly and transparently as possible.

Another benefit of the model is that it reduces the work to source suitable external projects. In 2019, we had a total of 16 product tribes with PG students catering for app. 300-350 students per teaching period. With a more "traditional" approach to capstone projects (*i.e.* a smaller team of 4-6 students etc.) we would have had to source many more projects.

[7] There is no implied (nor inferred) hierarchy, we call them C1 and C2 students, short for Capstone 1 and Capstone 2, respectively.

[8] Students may withdraw over the time-frame of the teaching period for many different reasons.

4.3 Teaching period timeline

Figure 2 summarizes the key aspects of the teaching period timeline. For our purposes, we have split a teaching period into four × 3-week time periods (or *iterations*) with a "preparation" week before the first and some "buffer time" after the last iteration: this can be adjusted if a teaching period has a different duration.

Prior to the start of the teaching period, the Product Owners and Tribe Leaders review the deliverables from the previous teaching period, discuss the vision of the respective project and define, as well as prioritize, work packages to be completed based on the anticipated number of squads. The Unit Chairs allocate Project Supervisors to the relevant squads and discuss teaching period timelines, assessment tasks etc. with the supervisors and the technical support staff. If need be, introductions are made between all stakeholders in order for communication during the teaching period to be as smooth as possible.

During Week 1 of the teaching period, the Tribe leaders get together with all returning *Senior* students of their respective Product Tribe, discuss the identified work packages, and with assistance of the Unit Chairs, allocate the Senior students to the relevant squads. For the remainder of the first three weeks, the (yet incomplete) squads review and refine their respective work packages, seek clarification from the Tribe Leader (if necessary), devise any necessary roles within the squad, and set up the "infrastructure" (communication tools, repositories, development tools etc.) to be used. The students further identify any project-specific skills gaps within each squad such that these can, ideally, be closed by *Junior* students assigned to the squad. Finally, at a high-level, each squad plans the activities for the remainder of the teaching period and define milestones for the three iterations to come.

In parallel, the Junior students are being *inducted* into the entire capstone project set-up, participate in refresher workshops on various project-relevant topics (*e.g.*, project management, teamwork, iteration planning and reviewing, repository systems), receive an overview of all the product tribes, and complete a skills survey. Although the Junior students can express preferences for which Tribes they would like to work in, the allocation to squads also takes the results of the skills survey, identified skills gaps by the Senior students as well as availability constraints into consideration. Ideally, all Junior students are allocated to a squad and introduced to their peers, project supervisor and the squad's *modus operandi* by the end of Week 3 so that everything is "work ready" by the start of Week 4.

During the first iteration (Weeks 4 to 6), the Junior students are inducted into their squad and Product Tribe, identify what knowledge and skills they lack in order to become a productive squad member, and are tasked to close their knowledge and skill gaps as much as possible. The Senior students, on the other hand, execute the plan they devised during the first three weeks and mentor the Junior students across a variety of project-specific issues. A weekly "stand-up" meeting with the Project Supervisor provides

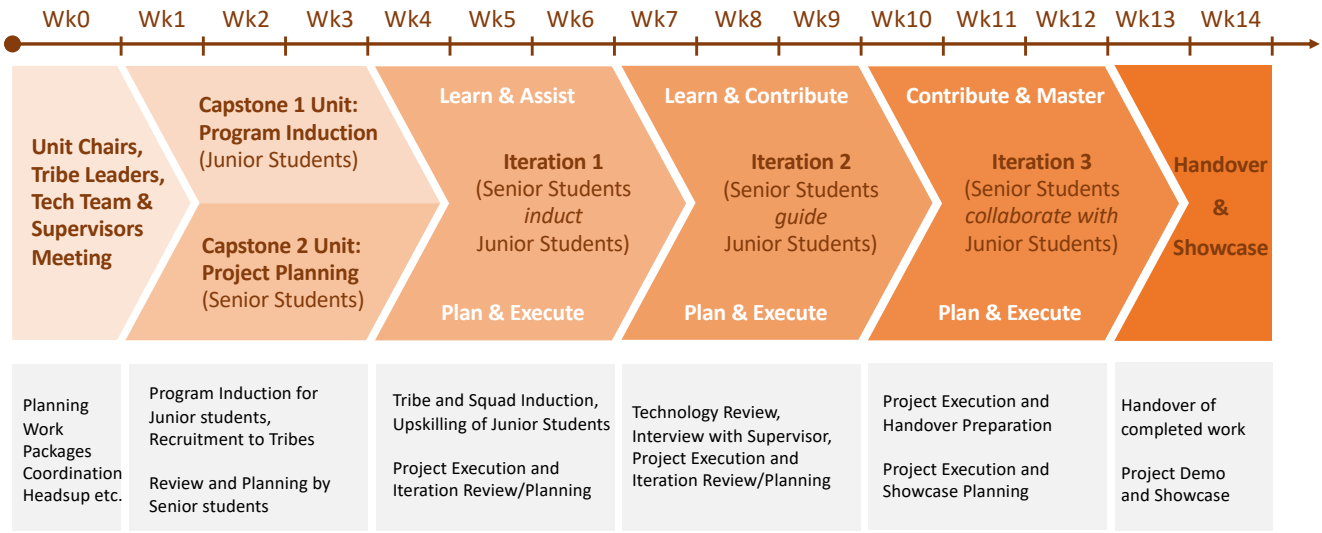

Figure 2: Teaching period timeline of proposed capstone model.

the opportunity for project updates, feedback and discussion on key challenges.

At the end of the iteration, each squad reviews the work completed as well as the effectiveness of their internal processes and practices, discusses their findings with the Project Supervisor, and devises a plan for the next iteration. In addition, each student (individually) compiles a brief report on their achievements and their plan forward (in line with the squad's plan). The individual retrospectives are then used to give students feedback on how they are "travelling". Dedicated 1-to-1 discussions are held in Week 7 in order to allow the students to discuss their progress with their Project Supervisor.

During the second iteration, Junior students are expected to conclude their up-skilling activities and gradually start to contribute towards deliverables. They may still need support from the Senior students, but are asked to become increasingly more independent. There is also an expectation that they become fully familiar with the context their Product Tribe and the scope of what their squad has been tasked to achieve. The Senior students are still mainly responsible for project execution, but they are instructed to reduce the level of support they are giving to the Junior students, and to expect them to contribute more. Similar to the first iteration, both squad-based and individual retrospectives are compiled and Project Supervisors provide feedback on these retrospectives.

During the final iteration, each squad is expected to complete the work packages they have been assigned. Both Junior and Senior students are expected to fully collaborate on project work and the level of mentoring by Senior students is reduced to a minimum. The aim is that at the end of this iteration, all Junior students have a sufficient knowledge about their Product Tribe to confidently graduate to the Senior student role when they come back for their second capstone project unit.

In order to finalize the squad's work, the Junior students are to compile handover documentation, including the relevant work products, and these will be passed to the Product Owner for their review. The handover artefacts will also serve as the basis for new squads' work packages in the following teaching period. The Senior students, on the other hand, prepare a show-case presentation where they demonstrate the main outcomes of the squad's work (the combined work of *both* the Junior and Senior students) to a wider audience, their peers, other Faculty and externals partners. We generally organize a show-case event at the conclusion of each teaching period where these presentations are made, judging is held and prizes awarded.

Once all project activities are completed, the students are tasked to compile a *learning portfolio*, providing a summary of all activities they were involved in during the teaching period, a reflection on key learnings as well as a discussion how they achieve the Unit Learning Outcomes for their respective capstone project unit.

4.4 Project Environment

One of the observations made in prior teaching periods was that not all Project Supervisors had the necessary in-depth knowledge in a project's technology-stack and supporting tools to confidently and competently provide guidance and support to their students. We also noticed that it was very challenging (if not impossible) for students to find suitable work spaces on campus, for meetings and collaborative work sessions. As a consequence, we introduced (i) a *physical collaboration work-space* (equipped with a mix of meeting rooms, wall-to-wall whiteboards, large screens, workstations etc.) with exclusive access for the capstone project students and (ii) a co-located help-hub with *Technical Advisors* available during office hours Monday to Friday. All meeting rooms are equipped with video conferencing facilities in order for "off-campus" students to participate in squad meetings.

The Technical Advisor's role is to provide all squads with assistance with any technology-related questions they have, hence

mitigating the need to have all Project Supervisors expert with all the technology involved, and also organize drop-in workshops on topics relevant across a number of Product Tribes. We instructed Project Supervisors to have their weekly "stand-up" meetings in the collaboration space and strongly encouraged the students to spend some time before/after their supervisor meeting collaboratively working on their projects.

In order to facilitate the burden of managing a large number of squads within each teaching period, and avoiding the problem of each squad having their own set of tools (with associated learning challenges), we introduced a uniform set of collaborative support tools across all squads: Microsoft Teams, Atlassian Bitbucket and Trello. One of the compelling reasons for using Microsoft Teams over other groupware solutions is the ease with which it allowed the unit administrators to pre-populate the squads in the continuing tribes, and to directly create the Teams groups from the Global Address List (GAL) in MS Exchange Server. Changes in squad composition, that involve additions and deletions, are also easily maintained using MS Teams, so this platform was selected for group-based conversations and documentation sharing. Thanks also to an Atlassian site-license and deployment on a local server, we conducted workshops on the use of Bitbucket, so that students had access to a web-based version control repository hosted on campus. Another important resource for web-based Kanban-style list-making is Trello[9] and this was also available to our students for collaboration, project management and planning, respectively. If squads wanted to use different tools for collaboration and file-sharing, they needed explicit approval from their Project Supervisors.

Although the chosen collaborative support tools may not be the ultimate solution to the needs to run and manage a large number of squads, they provided an acceptable compromise with regards to industry-relevance and adoption, usability, accessibility (at no extra costs to students), availability across multiple platforms, and administrative support, respectively.

4.5 Addressing the Motivations

Our model is designed to address the six motivations discussed in Section 2. The structure of the model supports longer-term continuous "live" products to be worked on, allowing the tackling of more complex problems (Motivation 1: Industry relevant experience). The on-boarding and mentoring of Junior students as well as an agile methodology is also in line with industry expectations and practice. The model provides a structure for stakeholders at different stages of knowledge of the project to interact (Motivation 2: Authentic learning experience). As the model is explicitly designed to support longer projects with multiple student squads, students can have breaks in between different capstone units, and return to the same Tribe, thus reducing the need to learn a new context (Motivation 3: Continuity of learning experience).

With the embedded ability to have multiple student squads working on a project in parallel, and for the projects to be worked on in a continuous way over a longer time-period, there is an increased chance for a client (*i.e.* the Product owner) to be satisfied and, therefore, continue to want to be involved (Motivation 4: A successful client experience). This is further facilitated by Tribe leaders shielding clients from "unnecessary" interactions with students and translating the project vision into work packages.

As well as supporting more continuous projects, the model also explicitly supports multiple simultaneous squads within each tribe. With this approach it is feasible for potentially hundreds of students to work on the same long-term product at any given point in time (Motivation 5: Scalability of approach).

The model is about the structuring of the capstone learning experience, but like any teaching, will require effective support for all students. As well as the embedded student-student support mechanisms, the model introduces a consistent approach and uniform tool-set, which is easier to support and maintain (Motivation 6: Effective support of students).

How well the model realizes these goals is very much dependent on the deployment for a particular capstone experience. The following section provides an evaluation of the model when used for the first time in university capstone units.

5 INITIAL EVALUATION

5.1 Setting for case study

All ICT-degree programs at Deakin have a mandatory capstone component consisting of two capstone project units, each requiring at least 120 hours of effort across a teaching period. All students must complete these units prior to graduation. The number of students across all our ICT-degrees has increased from approx. 1,500 in 2015 to approx. 4,000 in 2019, with anticipation of continuing growth in the coming years. Given the current trend, we expect in the order of 1,000 students enrolling in capstone projects each teaching period in the not distant future.

The proposed capstone project model as presented in the previous section was first introduced to two PG capstone project units in 2019 with the aim to roll-out the model to UG capstone project units in 2020. Across the two PG capstone project units we ran during the second teaching period, a total of 302 students, allocated to 29 squads (each with a unique Project supervisor) composed of 9 to 12 students,[10] worked on a variety of different projects.

5.2 Survey description

As part of the ongoing evaluation of the capstone units we routinely survey students, project supervisors, Tribe leaders, capstone unit coordinators, and the unit administrative team. We also sought feedback from external client organisations. Aside from assessment satisfaction (dealt with in a companion paper [42]), we are interested in perceptions around organisation of the squads, the success (or otherwise) of roles played by the actors in the model in Figure 1, and an evaluation of usefulness of the resources that support the capstone units, including the physical collaboration space, the help hub, technical advisors and up-skilling workshops. The focus of the survey is to seek evidence-based feedback that will further drive innovation in subsequent iterations. Of particular interest is

[9] www.trello.com

[10] Due to late enrollments and withdrawals, not all squads had the same number of students.

the organisation of the mixed squads, Junior and Senior capstone students, and whether this results in positive learning experiences and outcomes for both student cohorts.

To this end, a survey was designed for students with 26 ordinal choice questions and 2 open-ended textual questions. The open-ended questions called for two aspects of the assessment that worked well, and two aspects that could be improved, 5 of the 26 ordinal choice questions concerned assessment (dealt with in an accompanying paper [42]). Project supervisors were asked 18 ordinal choice questions and one free text question "What would you change about the unit? What works and what does not?" Throughout the remainder of this section, we will report on the key findings from the surveys.

5.3 Survey Results

93 students responded to the survey (51 Junior; 42 Senior), a response rate of 31% of the total enrollment. More than half of the respondents claimed that they had had industry-based group-work experience prior to the start of the capstone units (32 Junior; 17 Senior)[11] Of the 49 students who said they had prior industry-based group-work experience, 7 disagreed (2 of these strongly so) with the statement that the capstone experience created an environment similar to what is expected in industry, 9 were neutral to this statement and 33 agreed (12 of these strongly agreed) that the capstone experience created an environment similar to what is expected in industry. This suggests that students with prior industry experience were convinced that the capstone provided an authentic industry-like experience. On the other hand, only one of the students without prior industry-based group-work experience disagreed with the statement, but the total agreement rate was slightly lower compared to the students with prior industry experience (64% versus 67%).

The project supervisors were surveyed along the same lines. 12 responded, 2 disagreed, 1 was neutral, and 9 agreed (1 strongly agreed) with the statement the capstone experience created an environment similar to what is expected in industry. Therefore, there is doubt in some quarters that the capstone experience created an environment similar to what is expected in industry. One explanation lies in the observation that a couple of the tribes are inwardly focused, with a product owner internal to the organisation.

A main feature of the implementation of the model described in Figure 1 is that a squad contains both Junior and Senior students. This feature is intended to encourage mentoring between students at different stages through the capstone units. How well does mentoring (and communication) work in the mixed Junior and Senior squads?

71% of all Senior students responding to the survey felt positively about them bringing the Junior students up to speed with the project (*cf.* Figure 3) with only 3% disagreeing (the remaining 26% having a neutral view). On the other hand, as indicated in Figure 4, 69% of all Junior students agreed that the Senior students helped them to come up to speed with the project (with 22% strongly agreeing).

[11] An explanation for the difference between senior and junior students self-reporting industry experience is that the senior students have, after already completing the first capstone unit, reconsidered the definition of industry experience.

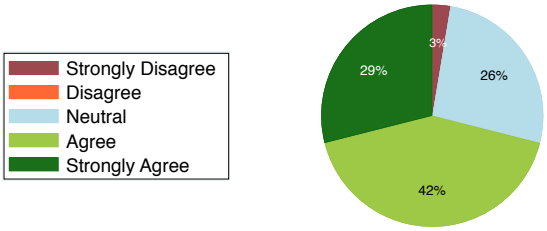

Figure 3: "As a Senior student, Junior students we responsive to my help coming up to speed with the project."

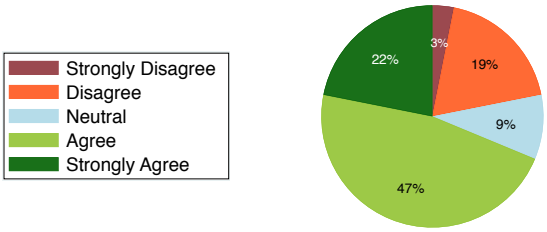

Figure 4: "As a Junior student, Senior students helped me come up to speed with the project."

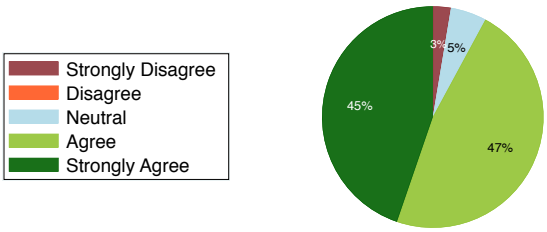

Figure 5: "As a Senior student, I was prepared and able to mentor Junior students."

9% of Junior students were neutral to this proposition with 22% disagreeing (3% strongly). Therefore, there is strong evidence that mentoring by Senior students is taking place and that it is positively delivered, but it seems somewhat less positively received by the Junior students, indicating they had not received sufficient help. This is something we need to investigate further and address in future.

On the question to Senior students, as to whether they were prepared and able to mentor the Junior students, the vast majority of Senior students agreed (92% agreement with 45% Strong Agree) with only 3% disagreeing (*cf.* Figure 5). When it comes to the question whether the communication between squad members was useful and productive, 81% of students responded positively to this question (with 35% strongly agreeing) with only 3% disagreement. However, more Senior students disagreed with the usefulness of the communication than Junior students (7% vs. 3%), albeit at a very low level.

We surveyed the students about the supporting project tools (*cf.* Section 4.4) and asked if Microsoft Teams, Bitbucket, and Trello were useful for facilitating group-work. 87% of students agreed

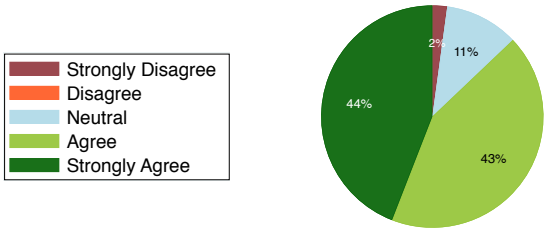

Figure 6: "The supporting project tools (MS Teams, Bitbucket, Trello) were useful for facilitating teamwork."

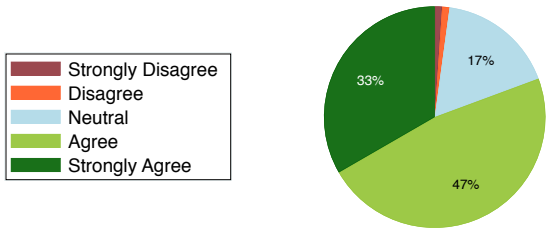

Figure 7: "The 3 Iterations (duration of 3 weeks each) were a good model for project work."

that they were, and only 2% disagreed, noting that they preferred alternatives (*cf.* Figure 6). In contrast, only 2 of the project supervisors were neutral about the collaboration tools, the remainder were positive.

Although 68% of the respondents found the dedicated collaborative space as useful (28% neutral responses), 36 students (*i.e.* 39%) indicated that they used the space less than five times during the teaching period. Looking at the responses to these two questions in more detail, only one student that used the collaborative space once a week (or more) disagreed with the space being useful to their project. Most students that used the space regularly found it useful, and not surprisingly, the students that did not frequent the space did not find it useful. Although from an academic perspective we think such a collaborative space is useful and should be regularly used, the data shows that we need to do more to encourage our students to make an effort and benefit from such a dedicated work-space.

88% of students survey agreed (of these 40% strongly agreed) with the statement that "The capstone project unit(s) allowed me to improve my IT skills in order to get IT-related employment in industry." Interestingly, we have a slightly higher agreement to this question by students with prior industry-based group-work experience versus students without such an experience (90% vs. 86%). However, more Junior students agree with this question than Senior students (92% overall agree versus 83% agree). We need to find out more why it is that 17% of Senior students did not feel as if their IT-related employment skills were improved by the second capstone project unit, and how we can address this in future.

In other results from the survey, not surprisingly, project supervisors were less enthusiastic than the students about certain student-facing services and facilities: the help hub, the technical advisors and the physical collaboration space, some project supervisors complained that the technical advisors had contradicted their advice. All the project supervisors recorded that interactions between the project supervisor and the squad were useful and productive, and while only 6 students disagreed with this statement, 79% of students recorded positive interactions with their project supervisor. Some project supervisors reported confusion among the students by the large number of different roles: project supervisor, unit chair, tribe leader, client, and technical advisors. Students also reported that it surprised them how much management and communication skills influenced the project's success.

In terms of what aspects of the unit worked well, students reported teamwork, communication, the groupware tools, the 3-week iteration cycles (*cf.* Figure 7), the end of teaching period showcase, and the weekly project supervisor meetings. On what aspects of the unit could be improved, students mentioned more workshops about tools, better interaction with other squads, so that experience could be shared, and better matching of supervisor's expertise to projects. Many students wanted better and more access to the Tribe Leader and/or external Product Owner.

6 DISCUSSION

In running our model, we identified a set of issues and challenges that we plan to address in coming teaching periods, including the following seven points.

1. Closer and better pairing of supervisors skills to projects. History has shown that one of the key factors for successful project outcomes are engaged project supervisors. We find that supervisors assigned to projects in their areas of interest are more engaged, supportive and better able to steer the project, as well as provide technical advice. The future plan is, when possible, to allow supervisors to bid on projects they are interested in supervising and to get academic staff earmarked for supervision involved early, when new Product Tribes are introduced.

2. Offer more up-skilling workshops. A significant number of students demonstrated limited skill-sets when they commenced the capstone units and so it is difficult to assign them, in any strategic way, based on the skills survey. We identify two types of workshops to organise: (i) one or more agile practice workshops to ensure that students understand and know how to refine user stories, manage project backlogs, set priorities, etc. and; (ii) technical workshops to ensure students are in a position to take ownership of their assigned tasks. This is especially critical for Junior students that need further skills in design thinking, *i.e.* Unity, web stack, Ethereum, and other technologies and approaches, respectively.

3. We need to minimise internal-facing tribes by building out our external industry network. In past years, external clients were lost because projects and students could not adequately respond to requirements and/or did not deliver on promises. To date we are going through what can be described as a model validation phase, a phase where we want to ensure the model actually works, that proper workplaces and resources are available, before exposing our model externally. Our approach to external clients has therefore

been conservative. Now that the model is stable and producing fruitful outcomes, our capstone management team are talking with more external partners with the aim of having an increased number of industry-based Product owners (currently around 20).

4. We should implement the Spotify chapter concept, a horizontal collection of students with common interests across squad boundaries. A useful feature of the Spotify agile model is to consider horizontal tiers (cohorts) across different Tribes and Squads as "chapters". We see this as a way to organise a response to the demands for additional up-skilling workshops mentioned earlier, as well as organising clubs around the roles or tools, e.g., around UX/UI, design thinking, MongoDB, React.js, jQuery etc. This is an opportunity both to share knowledge between students, who share the same role or are using the same tools, and also as a platform for Senior students to coach Junior students outside the bounds of their respective squads.

5. Better clarity on delegations to minimize interference between the advice offered by supervisor, tribe leader and technical advisors. With many stakeholders and roles in the project ecosystem, the suitability of communication is difficult to fully manage. For example, we have experienced students asking business (scope) related questions to technical advisors. These conversations led to confusion in some cases regarding project priorities. On the other hand, dialogue between roles is often necessary and productive, for instance some user stories or requirements planned by project supervisors or clients are not practical and need to be either broken-down or re-prioritised as a result of technical limitations or constraints. We are developing detailed role descriptions to help all stakeholders better understand their responsibilities in the project ecosystem.

6. It probably does not matter what group-ware tools you select. We have changed some of the group-ware tools used in the capstones from previous teaching periods, *i.e.* switching from Slack[12] to Microsoft Teams because of its integration with our infrastructure. From an operations perspective it is easier to add and remove groups and students via scripts compared to Slack. These changes turned out to be useful to reduce the management overhead, but did not affect collaboration (at least as evidenced in our survey), mostly because the majority of features required to run a successful project are available across all groupware platforms. When we re-evaluate the choice of group-ware tools in the future, we will use this more pragmatic approach.

7. Junior & Senior students benefit from collaboration in Squads. There are many useful features resulting from blending Junior and Senior students in a single squad. For Junior students, it allows a smooth transition to the project, up-skilling on relevant skills given the stability of the projects, and acquiring the necessary orientation from Senior students. We also noticed that Senior students are more enthusiastic about mentoring Junior students. They become mature when they get to play the role of a senior member in the team. Such an arrangement seems to push Senior students to try to find answers to questions from Junior students, both in relation to the project requirements and to the technology stack. Our initial observations indicate that blended squads work surprisingly well and we are encouraged to roll out blended squads in other project units where possible.

[12]https://slack.com

7 CONCLUSIONS AND FUTURE WORK

This paper reports the adoption of a hybrid agile organisational model based on Spotify's Tribes and Squads for the purpose of teaching delivery of capstone units containing large number of students. The design is motivated by a set of features that we want to satisfy: (i) industry relevant experience; (ii) authentic learning experience; (iii) continuity of learning experience; (iv) a successful client experience; (v) scalability of approach; and (vi) effective support of students.

Our initial evaluation of the proposed model shows positive indicators across the majority of these features. On the last point in particular, we have found evidence that student mentoring is occurring in blended squads and that it is positively given and received, without any perceived communication loss compared to squads that are composed of only students at the same level.

The only feature we do not have sufficient data to draw any conclusions on is feature (iv) – a successful client experience. As part of our future investigations, we intend to survey and closely work with the various Product Owners to identify their views and measure their experience.

The model we presented is evolving incrementally based on evidence-based feedback from its stakeholders: students, technical advisors, supervisors, Tribe Leaders, Product Owners, unit chairs, and the capstone project management team. The research team plans to conduct a longitudinal study that will allow the presentation of comparative results based on student performance in addition to measuring stakeholder satisfaction. Because of the size of the student cohort, organisational innovation can be introduced to a subset of that cohort, and student performance and stakeholder satisfaction tested against this baseline model.

ACKNOWLEDGMENTS

The authors would like to thank the members of the School of Information Technology Capstone Projects Team at Deakin University, especially Kristiina Tukk and Jesse Mcmeikan, for their support of this work. We would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Malek Al-Zewairi, Mariam Biltawi, Wael Etaiwi, and Adnan Shaout. 2017. Agile software development methodologies: survey of surveys. *Journal of Computer and Communications* 5, 05 (2017), 74–97.

[2] Mashal Alqudah and Rozilawati Razali. 2016. A Review of Scaling Agile Methods in Large Software Development. *International Journal on Advanced Science, Engineering and Information Technology* 6, 6 (2016), 828–837.

[3] Kent Beck. 1999. *Extreme Programming Explained: Embrace Change.* Addison-Wesley.

[4] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. The Agile Manifesto. (2001). http://users.jyu.fi/~mieijala/kandimateriaali/Agile-Manifesto.pdf. Accessed on 30-Oct-2019.

[5] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software Engineering Project Courses with Industrial Clients. *ACM Transactions on Computing Education* 15, 4 (2015), 17:1–17:31.

[6] Alistair Cockburn. 2004. *Crystal clear: a human-powered methodology for small teams.* Pearson Education.

[7] Sharon Coyle, Kieran Conboy, and Tom Acton. 2015. An exploration of the relationship between contribution behaviours and the decision making process in agile teams. In *Proceedings of the 36th International Conference on Information Systems.* Fort Worth, Texas, USA, 1–15.

[8] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. 2012. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software* 85, 6 (2012), 1213–1221.

[9] Xiaocong Fan. 2018. Seven Principles of Undergraduate Capstone Project Management. In *Proceedings of the International Conference on Software Engineering Research and Practice*. Las Vegas, Nevada, USA, 106–112.

[10] Görel Hedin, Lars Bendix, and Boris Magnusson. 2003. Introducing Software Engineering by means of Extreme Programming. In *Proceedings of 25th International Conference on Software Engineering*. Portland, Oregon, USA, 586–593.

[11] Rashina Hoda, James Noble, and Stuart Marshall. 2011. The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology* 53, 5 (2011), 521–534.

[12] Eric Kisling. 2019. Transitioning from Waterfall to Agile: Shifting Student Thinking and Doing from Milestones to Sprints. In *Proceedings of Southern Association for Information Systems*. 1–2.

[13] Henrik Kniberg and Anders Ivarsson. 2012. Scaling agile@ spotify. (2012). https://creativeheldstab.com/wp-content/uploads/2014/09/scaling-agile-spotify-11.pdf. Accessed on 30-Oct-2019.

[14] Dean Knudson and John Grundy. 2016. International Capstone Exchange – the SUT and NDSU Experience. In *Proceedings of 2016 Capstone Design Conference*. Columbus, Ohio. https://doi.org/10.13140/RG.2.1.1181.9764

[15] Dean Knudson and Alex Radermacher. 2011. Updating CS capstone projects to incorporate new agile methodologies used in industry. In *Processings of the 24th IEEE-CS Conference on Software Engineering Education and Training*. Honolulu, HI, USA, 444–448.

[16] Stefan Koch. 2005. Evolution of Open Source Software Systems–a Large-scale Investigation. In *Proceedings of the 1st International Conference on Open Source Systems*. Genoa, Italy, 148–153.

[17] Lowell Lindstrom and Ron Jeffries. 2004. Extreme Programming and Agile Software Development Methodologies. *Information System Management* 21, 3 (2004), 41–52.

[18] Baochuan Lu and Tim DeClue. 2011. Teaching Agile Methodology in a Software Engineering Capstone Course. *Journal of Computing Sciences in Colleges* 26, 5 (2011), 293–299.

[19] Vijan Mahnic. 2012. A Capstone Course on Agile Software Development Using Scrum. *IEEE Transactions on Education* 55, 1 (2012), 99–106.

[20] Angela Martin, Craig Anslow, and David Johnson. 2017. Teaching Agile Methods to Software Engineering Professionals: 10 Years, 1000 Release Plans. In *Proceedings of the 18th International Conference on Agile Software Development*. Cologne, Germany, 151–166.

[21] Andreas Meier, Martin Kropp, and Gerald Perellano. 2016. Experience Report of Teaching Agile Collaboration and Values: Agile Software Development in Large Student Teams. In *Proceedings of the 29th International Conference on Software Engineering Education and Training*. Dallas, Texas, USA, 76–80.

[22] Antonio F Mondragon-Torres. 2013. An agile embedded systems capstone course. In *Proceedings of 2013 IEEE Frontiers in Education Conference*. Oklahoma City, OK, USA, 127–133.

[23] Antonio F Mondragon-Torres, Alexander Kozitsky, Clifford Bundick, Edward McKenna, Eric Alley, Matthew Lloyd, Peter Stanley, and Roger Lane. 2011. Work in progress – An agile embedded systems design capstone course. In *2011 Frontiers in Education Conference*. Rapid City, SD, USA.

[24] Bahram Nassersharif and Linda Ann Riley. 2012. Some best practices in industry-sponsored capstone design projects. In *2012 Capstone Design Conference*. Champaign-Urbana, Illinois, USA.

[25] Andres Neyem, Jose I Benedetto, and Andres F Chacon. 2014. Improving software engineering education through an empirical approach: lessons learned from capstone teaching experiences. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. Atlanta, Georgia, USA, 391–396.

[26] Steve R Palmer and Mac Felsing. 2001. *A practical guide to feature-driven development*. Pearson Education.

[27] Ian Parberry, Timothy Roden, and Max B Kazemzadeh. 2005. Experience with an industry-driven capstone course on game programming. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. St. Louis, Missouri, USA, 91–95.

[28] Marko Poženel. 2013. Assessing teamwork in a software engineering capstone course. *World Transactions on Engineering and Technology Education* 11, 1 (2013), 6–12.

[29] Michael J. Prince and Richard M. Felder. 2006. Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases. *Journal of Engineering Education* 95, 2 (2006), 123–138.

[30] David F. Rico and Hasan H. Sayani. 2009. Use of agile methods in software engineering education. In *Proceedings of 2009 Agile Conference*. Chicago, IL, USA, 174–179.

[31] Diane Rover, Curtis Ullerich, Ryan Scheel, Julie Wegter, and Cameron Whipple. 2014. Advantages of agile methodologies for software and product development in a capstone design project. In *Proceedings of 2014 IEEE Frontiers in Education Conference*. Madrid, Spain, 1–9.

[32] Jan Schilling and Ralf Klamma. 2010. The difficult bridge between university and industry: a case study in computer science teaching. *Assessment & Evaluation in Higher Education* 35, 4 (2010), 367–380.

[33] Jean-Guy Schneider and Lorraine Johnston. 2003. eXtreme Programming at Universities – An Educational Perspective. In *Proceedings ICSE 2003*. IEEE Computer Society Press, Portland, Oregon, 594–599.

[34] Jean-Guy Schneider and Lorraine Johnston. 2005. eXtreme Programming – Helpful or Harmful in Educating Undergraduates? *Journal of Systems and Software* 74, 2 (Jan. 2005), 121–132.

[35] Jean-Guy Schneider and Rajesh Vasa. 2006. Agile Practices in Software Development – Experiences from Student Projects. In *Proceedings of the 17th Australian Software Engineering Conference*. Sydney, Australia, 401–410.

[36] Ken Schwaber and Mike Beedle. 2002. *Agile Software Development with Scrum*. Prentice Hall Upper Saddle River.

[37] Tucker Smith, Kendra M.L. Cooper, and C. Shaun Longstreet. 2011. Software engineering senior design course: experiences with agile game development in a capstone project. In *Proceedings of the 1st International Workshop on Games and Software Engineering*. Honolulu, HI, USA, 9–12.

[38] Richard Stansbury, Massood Towhidnejad, Jayson F Clifford, and Michael P Dop. 2011. Agile methodologies for hardware/software teams for a capstone design course: lessons learned. In *Proceedings of American Society for Engineering Education*. Russellville, Arkansas, USA, 1–13.

[39] Jennifer Stapleton. 1997. *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press.

[40] Chuan-Hoo Tan, Wee-Kek Tan, and Hock-Hai Teo. 2008. Training students to be agile information systems developers: A pedagogical approach. In *Proceedings of the 2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral Consortium and Research*. Charlottesville, VA, USA, 88–96.

[41] Robert H Todd and Spencer P Magleby. 2005. Elements of a successful capstone course considering the needs of stakeholders. *European Journal of Engineering Education* 30, 2 (2005), 203–214.

[42] Laura Tubino, Andrew Cain, Jean-Guy Schneider, Dhananjay Thiruvady, and Niroshinie Fernando. 2020. Authentic Individual Assessment for Team-based Software Engineering Projects. In *Proceedings of IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. ACM, Seoul, South Korea. to appear.

[43] Laurie Williams and Richard Upchurch. 2001. Extreme Programming in Software Engineering Education?. In *Proceedings of the 31st Annual Frontiers in Education Conference*. Reno, Nevada, USA, 1–6.