Journal of Information Systems Education

Volume 29
Issue 2
Spring 2018

Scrum-Based Learning Environment: Fostering Self-Regulated Learning

Tanya Linden

Recommended Citation: Linden, T. (2018). Scrum-Based Learning Environment: Fostering Self-Regulated Learning. *Journal of Information Systems Education*, 29(2), pp. 65-74.

Article Link: http://jise.org/Volume29/n2/JISEv29n2p65.html

Initial Submission:	16 August 2017
Accepted:	30 January 2018
Abstract Posted Online:	21 March 2018
Published:	13 June 2018

ISSN: 2574-3872 (Online) 1055-3096 (Print)

Scrum-Based Learning Environment: Fostering Self-Regulated Learning

Tanya Linden
Department of Business Technology and Entrepreneurship
Swinburne Business School, Faculty of Business and Law
Swinburne University of Technology
Hawthorn, Victoria 3122, Australia
tlinden@swin.edu.au

ABSTRACT

Academics teaching software development courses are experimenting with teaching methods aiming to improve students' learning experience and learning outcomes. Since Agile software development is gaining popularity in industry due to positive effects on managing projects, academics implement similar Agile approaches in student-centered learning environments. In this paper, we discuss teaching introductory programming based on Scrum. Our learning environment, supported by the Doubtfire learning management system, fosters perceived autonomy and perceived competence by providing tools and opportunities for self-regulated learners to adjust their learning strategies. Evaluation of the learning environment revealed that students want to be in control of their learning.

Keywords: Agile, Scrum, Self-regulated learning, Introductory programming, Programming

1. INTRODUCTION

Knowledge of programming concepts is perceived as being important for information systems (IS) and information technology (IT) courses as it facilitates development of problem solving and reasoning skills, as well as provides a foundation for learning other subjects in IS and IT. There is anecdotal evidence that introductory programming subjects are often the reason for high dropout rates and high failure rates. So far, there have only been two formal studies on the failure rates in programming. According to Bennedsen and Caspersen (2007), failure rates are on average around 30%; however, they indicate that the study was limited by the low number of respondents and data coming mainly from U.S.-based institutions. The same levels of failure were reported by the second study conducted by Watson and Li (2014). The authors of this paper found similar results in introductory programming subjects in IS courses. Therefore, it is not surprising that the research literature widely suggests that introductory programming is among the most difficult subjects for students and explores reasons behind these difficulties (Gomes and Mendes, 2007; Jenkins, 2002; Ma et al., 2011). Interestingly, Watson and Li (2014) came to the conclusion that the choice of the first programming language did not have an effect on pass rates. Thus, the investigation of issues affecting pass/failure rates in introductory programming continues.

In many institutions, introductory programming is taught in a traditional way with lectures, labs, and assessments containing lab exercises, assignments, and a final exam. According to Bennedsen and Caspersen (2007), in many colleges and universities, the final grade in the first-year programming course is affected by a large assessment such as a final exam, an assignment, or a project. Swinburne University of Technology chose a different approach. This approach is based on the adaptation of Scrum to teaching and learning in the context of the self-regulated learning framework (Young, 2005). Although the university is using Blackboard as its main learning management system (LMS), it was decided that Blackboard is rather inflexible in supporting the Agile approach to students' learning, and its user interface lacks the necessary features to support frequent resubmission of work and student-tutor communications, the views also supported by previous research (Carvalho, Areal, and Silva, 2011; Kim and Booth, 2015). An LMS named Doubtfire has been developed, and although it is treated as a work in progress by its developers, it has been successfully used to support IT and IS students in their learning explorations of programming concepts. This study explores the effects of a non-traditional approach to teaching programming concepts on students' self-regulated behavior as well as pass/failure rates in this category of subjects.

2. PARALLELS BETWEEN SOFTWARE DEVELOPMENT METHODOLOGIES AND TEACHING AND LEARNING METHODOLOGIES

Academics teaching software development subjects often see similarities between IT projects and teaching IT subjects (Alfonso and Botía, 2005; Chun, 2004). Although there exist multiple definitions of the term "project," most of them are based on the definition of the Project Management Institute

defining a project as a "temporary endeavor undertaken to achieve a unique product, service, or result" (Project Management Institute, 2017). Reiss (2007) adds a time dimension to this definition by emphasizing that a project is "a human activity that achieves a clear objective against a time scale" (p. 12). For a software development team, the goal is to produce a product that will be accepted by the customer, whereas for a student, learning a subject is a project with the clear objective to pass the subject.

Academic staff who observed these similarities examined software development methodologies hoping to learn from best practices in one industry and apply those practices in another. Initial reports on introducing Agile software development practices in software engineering subjects describe students working through software development projects using, for example, eXtreme programming (XP) methods (Johnson and Caristi, 2002; Reichlmayr, 2003; Williams and Upchurch, 2001); however, these reports focus on the adoption of software development practices in educational settings only to teach students this method of software development. They do not consider approaches of adopting the metaphor of XP or another Agile approach to projects of teaching and student learning.

In contrast with those reports, Chun (2004) examined Agile software engineering in order to apply its best practices to teaching and learning. He proposed the Agile Teaching/Learning Methodology (ATLM) which was later adopted by other academics in a variety of tertiary education settings. The ATLM facilitates self-learning and promotes knowledge sharing through an ATLM e-learning platform.

Alfonso and Botía (2005) combined the Agile Rational Unified Process (RUP) with selected features of XP and Scrum development approaches and applied them to pedagogical tasks from building knowledge to assessing it in a group-work environment where students learn through active participation in software engineering projects. The teaching staff act as development team managers.

D'Souza and Rodrigues (2015) also based their work on XP features. They developed Extreme Pedagogy by identifying correlations between fundamentals of the software process (product, customer, developer) with the cornerstones of the pedagogical processes (learning, student, and teachers as "developers of learning") (p. 830). The main characteristics of Extreme Pedagogy are "learning by continuous doing," "learning by continuous collaboration," and "learning by continuous testing." All these features are known as facilitating active student learning.

Some academics examining the literature on Agile approaches to teaching (e.g., Grossman et al., 2011; Tengberg, 2015) note the lack of studies exploring Agile approaches to teaching and learning. The studies that exist report on the success of adopting Agile practices, however further exploration is required. Consequently, this study develops these ideas by exploring aspects of adopting Scrum as an Agile approach to teaching and learning.

3. ADOPTING BEST SOFTWARE DEVELOPMENT PRACTICES FOR EDUCATION

As observed by Chun (2004), students' learning needs are affected by many variables and therefore educators should consider Agile teaching approaches. This is similar to software developers discovering that the most well-known and popular waterfall model was gradually losing its suitability for modern projects. The waterfall model is too rigid to allow software developers to easily adjust based on changing project needs and customer requirements (Balaji and Murugaiyan, 2012). Therefore, alternative approaches have been developed, tried, and adopted. The Agile movement proposes approaches to deal with the unpredictability of projects. The movement proposed 12 software development principles, but they are also relevant for the actual learning process undertaken by students. Table 1 lists the principles of Agile development (Beck et al., 2001) and discusses adoption of the principles in the context of the students' learning process. In this interpretation, we treat a student as a developer being in charge of studying a subject through the semester. This interpretation is in line with the student-centered teaching and learning concepts which focus on what the student does (O'Neil and McMahon, 2005). In this analogy to software development, a teaching staff member becomes a customer providing useful feedback on a student's deliverables, which is different to previous research where teaching staff are considered to be managers of student development teams.

One of the popular Agile approaches to software development is Scrum, which is defined as an iterative development method where the product is being developed in increments (Beedle et al., 1999). The development happens within short time intervals called *sprints* which are mapped onto the development stages. Within each sprint, developers work on specified tasks, called *backlog*. Each sprint ends with a deliverable, a *demo* to the customer. After a demo, re-prioritization happens which involves creating a new backlog (based on leftovers from the previous sprint and new tasks planned for the current sprint). A *burndown chart* graphically represents the progress of the project by comparing the ideal or targeted progress line with the depiction of the real progress (Karlesky and Voord, 2008). Although Scrum often assumes teamwork, its techniques are equally applicable to single developers (Blom, 2010). The Scrum approach, if applied to an educational setting, could foster self-regulated learning, especially if supported by an appropriate learning environment.

4. SELF-REGULATED LEARNING

For centuries, educators have been searching for ideal teaching approaches. For more than 20 years, we have been observing a strong shift from traditional teaching methods where the focus is on the content fed to a learner to learner-centered approaches focusing on the "needs, skills, and interests of the learner" (Norman and Spohrer, 1996, p. 26). This shift has been supported and facilitated by rapid development and adaptation of educational technologies. Technological advances fostered the creation of engaging learning environments that support individual learning styles, self-paced learning, and interactivity that significantly complement if not replace "traditional" teaching (Hannafin and Land, 1997). Such environments are supposed to foster students taking control of their learning and becoming self-regulated learners. Self-regulated learners are active participants in the knowledge construction process who "set goals for their learning and then attempt to monitor, regulate, and control their cognition, motivation, and behavior

Agile Manifesto Principles (Beck et al., 2001)	**Our Interpretation of Principles in the Student-Centered Learning Environment**
"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."	Students continuously complete tasks and provide deliverables for assessment by teaching staff.
"Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."	A student may need to adjust his/her learning style to achieve the subject learning outcomes within the expected timeline.
"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."	Due to a semester being a strictly 12 weeks period, students' deliverables should happen from every week to every fortnight.
"Business people and developers must work together daily throughout the project."	A student and teaching staff need to work together for the student to achieve the subject learning outcomes.
"Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."	Teaching staff expect students to be motivated to learn and should provide a challenging and exciting learning environment which supports student-centered learning.
"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."	Although a variety of methods are used for modern communications, there should be face-to-face communication opportunities where students support each other as well as get necessary instruction and help from teaching staff during formal and informal sessions.
"Working software is the primary measure of progress."	Students' learning is judged by the quality of the deliverables.
"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."	The subject should be designed to maintain a constant learning pace, with regular deliverables. Instead many subjects seem easy in the first 3-4 weeks with an unexpected jump in difficulty levels closer to the middle – end of the semester rapidly increasing workload and amount of stress for students (as well as marking workload for teaching staff).
"Continuous attention to technical excellence and good design enhances agility."	Teaching staff should provide valuable feedback on improvement opportunities and students should be given an opportunity to implement and deliver improved versions of their work as evidence of their learning.
"Simplicity – the art of maximizing the amount of work not done – is essential."	Students should be able to see how much more they could learn beyond the constraints of the subject.
"The best architectures, requirements, and designs emerge from self-organizing teams."	Students being in charge of their learning approaches often find a "study buddy" to maximize their learning, they decide on getting support based on their individual needs.
"At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."	At the end of each semester students should be encouraged to reflect on their learning approaches to ensure that they learn from the mistakes and adjust their learning style and time management for the next subject.

Table 1. Agile Manifesto Principles and their Application to Student-Centered Learning Environment

guided and constrained by their goals and the contextual features in the environment" (Pintrich, 2000, p. 453).

Young (2005) proposed a social cognitive framework for self-regulated learning. Empirical work evaluating the framework was conducted with students learning core marketing subjects (see Figure 1). The framework depicts four areas for self-regulation identified by previous studies (Pintrich, 2000; Zimmerman, 1995): cognition, motivation, behavior, and context. Self-regulated behavior is characterized by learning strategies that students employ to achieve quality learning outcomes (Pintrich and Groot, 1990). Young (2005) established that self-regulated behavior is affected to various degrees by self-regulated motivation (intrinsic or extrinsic), self-regulated cognition, and classroom environment. Intrinsic motivation has been shown to be driven by satisfaction with the learning and interest in the subject matter, whereas extrinsic motivation is more rewards- or grades-oriented. The model confirmed the link between intrinsic motivational value and self-regulated behavior and cognitive strategies which is in line with previous studies (e.g., Pintrich and Groot, 1990; Zimmerman, 1990).

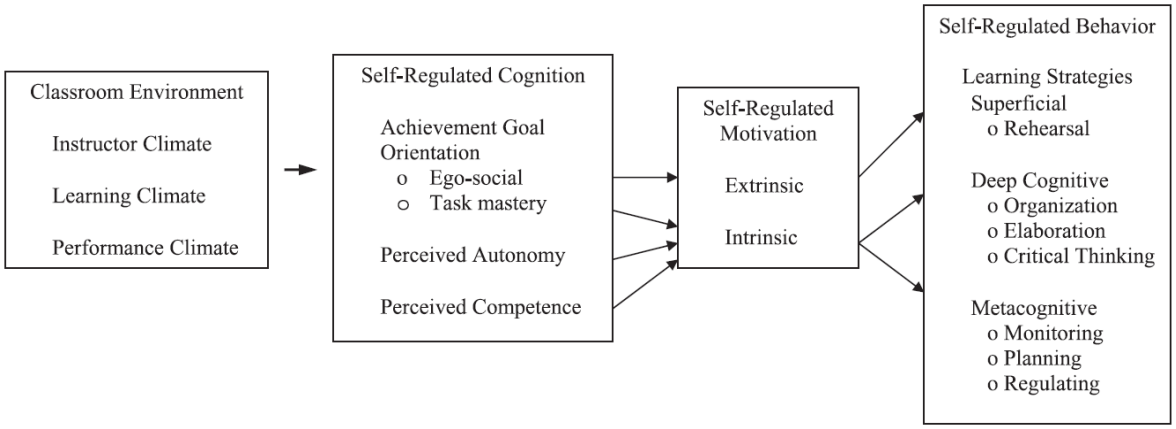

Figure 1. A Social Cognitive Framework for Self-Regulated Learning (Young, 2005)

Self-regulation of cognition depends on achievement goal orientation, perceived autonomy, and perceived competence. Young (2005) comes to the conclusion that students with a high sense of perceived competence and autonomy are intrinsically motivated. He interprets perceived autonomy as students' control over the factors influencing their outcomes in learning the course and perceived competence as their ability to complete tasks and perform through the course. Achievement goal orientation is a personal set of beliefs motivating a person to undertake learning and can be mastery-oriented, i.e., the goal is learning and self-development, or ego-socially oriented with the focus on rewards, such as grades and social standing (Urdan et al., 1998; Young, 2005).

The parameter "classroom environment" of the framework was assessed using three general subcategories: the instructor climate, the learning climate, and the performance climate. Each sub-category in turn was assessed using specific criteria. For the instructor climate, the criteria affecting students' learning are student-instructor personal interaction, informative supportive feedback, and enthusiasm of the instructor. Learning climate is comprised either of traditional teaching including lectures and readings or active/interactive teaching with hands-on learning activities and real-world applications. Performance climate is characterized by clear course goals and expectations, emphasis on learning, and contrast between grades being determined by individual versus group performance. The results of the framework evaluation emphasize the importance of instructor-student personal interactions including formative supportive feedback and their positive effect on students' intrinsic motivation and self-regulated behavior.

Educators recognize two types of feedback: summative feedback where students' achievements are represented by their scores and grades and formative feedback which is feedback provided with the aim to support a student throughout the learning process (Yorke, 2003). Formative evaluation of student learning has been widely discussed in the research literature (Black and Wiliam, 2003; Bloom, 1971; Shute, 2008). Nicol and Macfarlane-Dick (2006) evaluate formative assessment and feedback in the context of self-regulated learning and discuss seven principles that address the three aspects of self-regulated learning, i.e. behavior, motivation, and cognition.

Since the classroom environment plays an important role in increasing intrinsic motivation and subsequently self-regulated learning strategies, we investigate the application of the Scrum approach to create a learning environment supported by an LMS that would facilitate self-regulated learning.

5. SCRUM APPROACH FOR SELF-REGULATED LEARNERS

At Swinburne University of Technology, introductory programming subjects in computer science and information systems are taught using the Doubtfire system developed at this university. In our application of Scrum, students play the role of developers whereas teaching staff become customers. At the beginning of the semester, students are given a list of tasks for development which they are supposed to deliver on a regular basis within a sprint (e.g., weekly or biweekly). Students/developers work through a backlog within each sprint (a weekly set of tasks), whereas teaching staff/customers provide regular feedback on submissions, either accepting a deliverable or sending it back for improvement. The formative feedback reflects on bugs in the code, user-friendliness, as well as suggesting the ways to optimize the code. Collaboration is optional and students only resolve to use it when they need help in completing the tasks.

Burndown charts are used to show students' progress through tasks (Woodward et al., 2013). In Doubtfire, burndown charts depict target completion, actual completion, and projected completion (see Figure 2). The target completion line is based on task due dates as set in Doubtfire. Actual completion reflects dates of students' submissions, and projected completion is the result of calculating the end date of submissions based on the velocity of actual submissions.

Students do not get marks for submitted deliverables, but instead they get detailed feedback and an opportunity to resubmit the program code until it is working and efficient. That is when the deliverable is marked as complete. The goal of the student is to get all submissions marked as complete by the end of the teaching period.

Grading occurs based on the difficulty of tasks. The tasks are evaluated as pass (P), credit (C), distinction (D), and high distinction (HD) level. Each student sets a goal for themselves, and Doubtfire gives each student access to the tasks based on their set goal. For example, a student who decided to go for a Pass will be shown pass level tasks only. A student aiming at

Figure 2. Student's Dashboard in Doubtfire

Credit will be shown Pass and Credit level tasks (as depicted in Figure 2). For control purposes, all students must complete two closed-book tests to ensure their understanding of important programming concepts.

By the end of the teaching period, students generate a portfolio of completed tasks. Based on the quality of the portfolio and test results, each student is evaluated against his or her goal and to what extent it is reached (e.g., for a student aiming at distinction, we have a range of marks 70-79 so the evaluation is conducted on whether this student reached the distinction level and, if yes, what their mark would be within the distinction range).

Table 2 maps the features of our non-traditional classroom environment onto the four aspects of the social cognitive framework for self-regulated learning proposed and validated by Young (2005). Our classroom environment is comprised of the Doubtfire LMS where the subject convener sets up the tasks to be completed as sprints with set deadlines and where teaching staff can provide formative feedback for students.

The environment created for teaching introductory programming fosters perceived autonomy and perceived competence. In the first week of the semester, students decide on the grade for which they wish to aim. By giving students an opportunity to make this choice, we encourage students to be in control of their learning from the very beginning.

Most students start with aiming at high distinction which reflects on their goal-orientation behavior and their perceived competence. Those who are mastery-oriented usually keep this goal throughout the semester and take action to achieve it. If they scale back, it is usually to distinction level. Students selecting high distinction for ego-social reasons often do not achieve this level when they discover that the learning curve is steep and the tasks are getting more difficult from one week to another and require constant efforts and regular submissions and resubmissions to achieve the required quality. These students try to wear down staff by re-submitting the work with little changes and show a lack of interest in gaining knowledge (Woodward et al., 2013).

Self-regulated learning and active learning environments are linked by the bi-directional relationship (Boekaerts, 1999). Engaging classroom environments promote self-regulated behavior and facilitate improvement of self-regulatory skills. At the same time, a student needs self-regulatory skills to take full advantage of resources offered by a non-traditional learning environment. Throughout the semester, we observed improvement in self-regulatory skills through students'

Classroom environment	**Self-regulated cognition**	**Self-regulated motivation**	**Self-regulated behavior (learning strategies)**
▪ Doubtfire (LMS) ▪ Sprints with pre-set tasks (by difficulty level) ▪ Teaching staff ▪ Formative feedback ▪ Burndown charts	▪ Setting a goal (choice of grade) ▪ Ability to change the goal grade	*Intrinsic*: ▪ setting the goal and sticking to it throughout the semester ▪ accepting feedback positively *Extrinsic*: ▪ superficial approach, resubmissions just to get a "complete" mark ▪ sometimes negative reaction to formative feedback requiring resubmission	▪ Re-submitting tasks ▪ Follow-up on provided feedback ▪ Selecting communication methods with staff ▪ Adjusting order of tasks in the backlog ▪ Dealing with deadlines ▪ Opting to work alone or with classmates

Table 2. Scrum Approach as Facilitator of Self-Regulated Learning

adjusting their learning strategies. For example, some students opt for sending longer messages in response to staff formative feedback via email or they opt for discussing the feedback face-to-face asking for clarification, explanations, and examples, therefore demonstrating a desire for deep understanding of concepts and genuine learning. Students also decided on the regularity of attending face-to-face sessions such as labs and consultations.

Strategies on adjusting the order of submitted tasks is often related to the issues of dealing with submission deadlines. Some students observe weekly deadlines with minimal deviations whereas others focus on Pass and Credit tasks as a safety net and start working and submitting distinction and high distinction tasks only closer to the end of the teaching period. Less confident students start with Pass tasks and need some encouragement from teaching staff to convince them to attempt Credit level tasks. As reported by Cain, Woodward, and Pace (2013), our learning environment gives an opportunity to students with the slow start (e.g., initially struggling with concepts) to catch up by applying their own learning strategies. Many of these students demonstrate sufficient mastery resulting in good learning outcomes and good final grade for the subject.

6. EMPIRICAL WORK

Cain, Woodward, and Pace (2013) discussed themes and patterns in students' progress while learning introductory programming within our non-traditional learning environment. Their work was based on data extracted from burndown charts. This study explores two additional questions:

1. What is students' acceptance of our non-traditional approach using Scrum to facilitate the acquisition of self-regulated learning skills?
2. Does our non-traditional, Scrum-based approach improve students' pass rates in the introductory programming subject?

6.1 Data Collection and Survey

To answer the first question, the main features of our approach were identified and a questionnaire of five questions was developed (see Table 3). To foster perceived autonomy, we allow students to set their target grade and change it if they want to. The tasks allocated to them are based on the target grade. To achieve the target grade, students are allowed to keep resubmitting the tasks until the marker assesses the submission as meeting the requirements. We wanted to know what students think about these features.

Question 1 asked students to mark their preference by choosing between a traditional assessment approach (e.g., assignment, test, exam) and an alternative approach using sprints and Doubtfire for submissions. Questions 2-5 required Agree/Disagree answers, and a textbox was provided for further elaboration on the student's answer.

1.	Which assessment approach do you prefer: Traditional or Alternative (using Doubtfire)?
2.	A student should be able to decide upfront what grade (Pass, Credit, Distinction, High Distinction) he/she wants to achieve in the subject.
3.	A student should be able to change the grade as a goal for the subject during the semester.
4.	Students aspiring for a different grade level (Pass, Credit, Distinction, High Distinction) should be given assessment tasks varying in the difficulty levels.
5.	Students should be given an opportunity to resubmit their work to achieve a grade they are aspiring to.

Table 3. Questionnaire

For sample selection we targeted students who are currently learning programming concepts in the information systems undergraduate degrees at Swinburne University of Technology. These students are using our non-traditional learning environment based on Scrum and supported by Doubtfire. Eighty eight students were sent a request to complete the questionnaire. The invitation to participate in the survey was advertised in the last three weeks of the teaching period on Blackboard. Students were notified that participation is voluntary with the implied consent, i.e., students who opened the survey had an option to read the questions and after that make their decision on participation. Thirty five students (40%) provided their responses. Students who stopped attending classes and/or submitting their deliverables were unlikely to see the invitation to participate in the survey.

6.2 Students' Acceptance of Our Non-Traditional Scrum-Based Approach

The distribution of students' responses to the survey questions is shown on Figure 3. Some students also provided explanations of their choices. Open coding was used to identify themes emerging from students' answers. Some elaborations simply confirmed the choice of the answer whereas others provided interesting insights including students commenting on Doubtfire features, as well as their self-regulated learning skills and expectations.

In response to Question 1, 88.6% preferred the non-traditional approach used in the programming concepts. Elaborating on this choice, many students commented on Doubtfire features providing the ability to track their progress in the subject:

> Doubtfire is the easiest way of keeping track of work that has been completed and marked or the work which is due

> More frequent feedback, motivation to keep up with content and feedback/tasks etc. are all in one place for revision

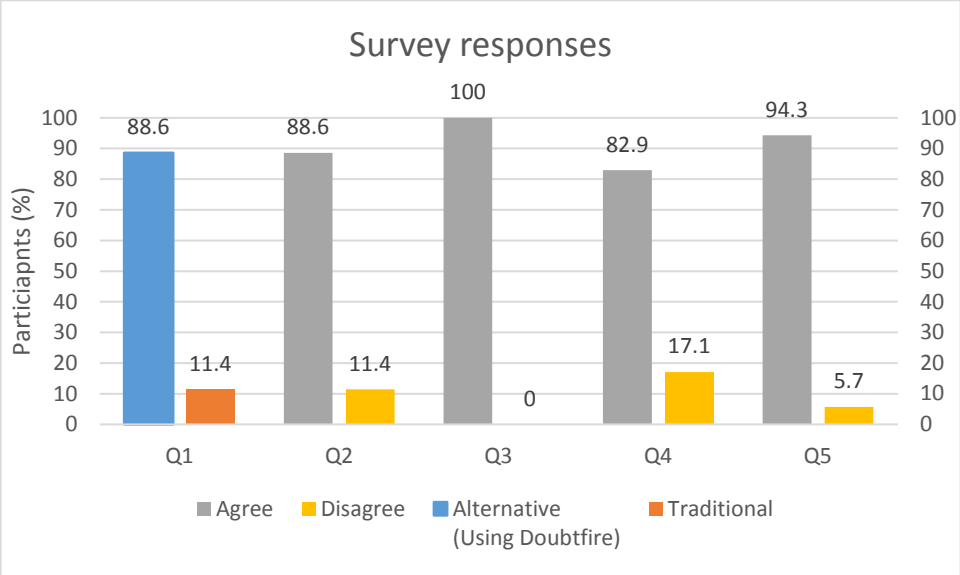

Figure 2. Survey Responses (%)

> I love the use of Doubtfire as it gives me a weekly gauge as to what level I am at and what I should have learnt by now. In other subjects that do not use Doubtfire, I have to wait for a topic test or something similar in order to gauge my understanding of the subject

In response to Question 2, 88.6% of students agreed that they should be able to decide upfront on the grade to be achieved. Many students feel that setting a grade creates a clear goal for them. They also commented on the system feature that by setting the grade they can see upfront which tasks they need to complete to achieve the goal:

> The destination is clearer than the traditional marking scheme.
>
> Set the goal and then try to achieve it.
>
> See what you have to do to achieve that grade.
>
> I think this is a very clever, transparent way of grading students at university. 'If you complete all of these tasks to a suitable standard, you are guaranteed a grade of X.'

Of the 11.4% (4 students) who selected 'Disagree,' one anecdotal response indicated preference of the traditional assessment system:

> I personally think all students should be required to attempt all tasks – if they are unsuccessful or provide poor content then that should lower their grade.

For Question 3, all respondents (100%) voted in favor of being able to change their goal grade. The comments show students' interpretation of the grade as a goal they set to achieve, subject difficulty, and their ability to catch up with work:

> You may start with high expectations for yourself but soon realize that these may not be realistic.
>
> Things change throughout the semester, the unit could be harder than expected so you should be able to change grades during the semester.
>
> If a student begins aiming for a credit, but finds this relatively easy, they should be able to rectify what they've missed and still achieve a distinction.

Although the point of the workload was not in the question (it was addressed in Question 4), some students linked the grade with workload, clearly showing that they agree with the arrangement 'the higher the grade the more work a student will have to do:'

> Students may find the workload easier or harder as they progress through the semester and should therefore be able to modify their goals accordingly.
>
> At the start of the semester most students aren't aware of exactly how much work a particular unit might need. So if they start to struggle to keep up they can make the decision which units they are less likely to do very well in and re-adjust those unit's workload.
>
> In the realization of other subjects piling up, it may be difficult to complete tasks if you are unable to change grades during the semester.

Some comments (the last one being a good example) also show the link between grades and time management in students' minds.

Question 4 asked students' opinions on whether students aspiring for a different grade should have to complete assessment tasks varying in the difficulty levels with 17.1% of respondents not in favor of this approach. For example, a

comment below notes a weakness of such an approach in the case of students under-estimating their ability to reach a higher grade:

> Although someone may be aiming to just pass, they may have the ability to surpass this. By providing them with a task of an alternative difficulty you are limiting their ability to perform and therefore grading in an unfair way. It makes the most sense to provide everybody with the same, flat and even assessment and then mark students based on this with their goals in mind.

The second comment against this approach was:

> Everyone should be given the same tasks, whether or not you want to complete it is up to you.

All other comments supported the idea that students aspiring for different grades should have different sets of tasks to complete, e.g.:

> Credit should be harder than a Pass? Yes, they should be showing a deeper knowledge of the subject not just more of the same.

> If a student wants a HD it is expected that they will have to complete more difficult tasks in order to achieve it.

However, one student felt that there should be no overlap in tasks:

> It seems that the harder tasks should be given to the people who want higher marks, but I think if you complete the harder Credit tasks, you should not need to complete the Pass tasks.

Finally, in responding to Question 5 on the opportunity to resubmit work to achieve the desired grade, only two students (5.7%) were against such an option. In their comments, students linked the resubmission not to grades but to learning:

> It is a good way to learn and improve throughout the semester while still getting the grade you want.

> Only way to learn.

> Feedback is key to improvement and everyone should be given a second chance to succeed.

> Resubmission helps gain a deeper understanding of the tasks especially if the student is struggling.

The above results demonstrate that the majority of our students are in favor of the environment that allows them to work using a Scrum approach and supports self-regulated learning. Their responses and comments show their satisfaction with the ability to work in short sprints, submitting incremental deliverables, and having a way to keep track of their progress. These responses also illustrate the importance of perceived autonomy and perceived competence. They show appreciation of feedback and the ability to learn from it. One student directly referred to feedback in a workplace environment:

> In a job, it would be necessary to get feedback.

Responses to this question also demonstrate students' learning strategies, such as taking advantage of formative feedback to improve the quality of deliverables, being able to learn from formative feedback, and demonstrating implementation of learned concepts through resubmission. One student commented on his strategy as:

> I personally keep a file with all of the major mistakes I have made so that I don't make them again.

Responses to Questions 2 and 3 demonstrate that many students want to be in control of their learning. Although these questions ask about setting goals as grades to be achieved, students interpret setting such a goal as setting an amount of work they will have to accomplish. Responses to Question 5 reflect on students' views on learning. Although none of our questions referred to learning as an objective of studies, students turned their comments from grades to learning outcomes. The responses not only identified the importance of formative feedback for the learning process, but also an opportunity to act on feedback and demonstrate learning from the feedback. Something that teaching staff always knew and previous research confirmed now becomes conventional wisdom for our students who use formative feedback and resubmission opportunities as their self-regulated learning strategies.

6.3 Does Scrum Approach Improve Pass Rates?

The short answer to the second question under investigation is "no." We examined results of the last three semesters of the subject "Introduction to Programming" taught at the Faculty of Science, Engineering and Technology, Swinburne University of Technology (Table 4). Only students who attempted the subject were considered (i.e., we did not count those who enrolled but did not submit any deliverables).

These results are in line with the discussion of students' progress reported by Cain, Woodward, and Pace (2013). In the sample of the burndown charts of students' progress Cain, Woodward, and Pace examined, 38% completed less than 75%

Number of Students	Semester 2, 2015	Semester 1, 2016	Semester 2, 2016
Pass or higher	207	349	158
Fail	65	97	87
Total	272	446	245
Failure rate	23.9%	21.7%	35.5%

Table 3. Failure Rates in Introductory Programming

of tasks, and according to their findings, these students demonstrated a superficial approach to their studies and a lack of interest in learning. So, although the Scrum approach is beneficial for students who are motivated and are receptive of facilitation of self-regulated learning, it is not helpful for students who are not interested in learning.

7. CONCLUSION

Introductory programming subjects are known for having disappointingly high failure rates. For years, educators have been examining variables affecting students' learning. Academics teaching IT and IS courses noticed similarities in teaching IT subjects and running IT projects, and as a result, there have been several approaches in applying Agile development to teaching IT subjects. In this paper, we reflected on the implementation of Scrum to create a teaching and learning environment that fosters self-regulated learning. We explored students' views on this non-traditional approach to teaching programming concepts and found that the majority of students were in favor of being in control of their learning. They showed support of the option of setting the goal for themselves and being able to develop learning strategies to reach that goal, as well as the chance to adjust the goal based on their perceived competence. Students also expressed appreciation of having formative feedback and the opportunity to act on it, to learn from mistakes, and to be able to demonstrate their learning through resubmission. However, although this non-traditional approach benefited self-regulated learners, it did not improve motivation of disinterested students and did not have any positive effect on the ratio of pass/failure rates. So, it is not enough to provide a computer-based learning environment to support self-regulation. Further research is needed on which tools affect students' motivation and how we can improve self-regulation skills. Future studies may consider triangulated data, i.e., data from larger student cohorts and data from teaching staff to provide a better understanding of variables influencing self-regulation.

8. REFERENCES

Alfonso, M. I. & Botía, A. (2005). An Iterative and Agile Process Model for Teaching Software Engineering. *Proceedings of the 18th Conference on Software Engineering Education & Training*, IEEE.

Balaji, S. & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A Comparative Study on SDLC. *International Journal of Information Technology and Business Management*, 2(1), 26-30.

Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). *Principles Behind the Agile Manifesto*. Retrieved from http://agilemanifesto.org/principles.html.

Beedle, M., Devos, M., Sharon, Y., Schwaber, K., & Sutherland, J. (1999). SCRUM: An Extension Pattern Language for Hyperproductive Software Development. In B. Foote, H. Rohnert, & N. Harrison (Eds.), *Pattern Languages of Program Design* (Vol. 4, pp. 637-651). Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Bennedsen, J. & Caspersen, M. E. (2007). Failure Rates in Introductory Programming. *ACM SIGCSE Bulletin*, 39(2), 32-36.

Black, P. & Wiliam, D. (2003). 'In Praise of Educational Research:' Formative Assessment. *British Educational Research Journal*, 29(5), 623-637.

Blom, M. (2010). Is Scrum and XP Suitable for CSE Development? *Procedia Computer Science*, 1(1), 1511-1517.

Bloom, B. S. (1971). *Handbook on Formative and Summative Evaluation of Student Learning*. New York: McGraw-Hill.

Boekaerts, M. (1999). Self-Regulated Learning: Where We are Today. *International Journal of Educational Research*, 31(6), 445-457.

Cain, A., Woodward, C. J., & Pace, S. (2013). Examining Student Progress in Portfolio Assessed Introductory Programming. *Proceedings of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Kuta, Indonesia.

Carvalho, A., Areal, N., & Silva, J. (2011). Students' Perceptions of Blackboard and Moodle in a Portuguese University. *British Journal of Educational Technology*, 42(5), 824-841.

Chun, A. H. W. (2004). The Agile Teaching/Learning Methodology and Its E-Learning Platform. *Proceedings of the International Conference on Web-Based Learning (ICWL 2004). Lecture Notes in Computer Science*.

D'Souza, M. J. & Rodrigues, P. (2015). Extreme Pedagogy: An Agile Teaching-Learning Methodology for Engineering Education. *Indian Journal of Science and Technology*, 8(9), 828-833.

Gomes, A. & Mendes, A. J. (2007). Learning to Program – Difficulties and Solutions. *Proceedings of the International Conference on Engineering Education – ICEE 2007*, Coimbra, Portugal.

Grossman, F., Tappert, C., Bergin, J., & Merritt, S. M. (2011). A Research Doctorate for Computing Professionals – A Ten Year Experience. *Communications of the ACM*, 54(4), 133-141.

Hannafin, M. J. & Land, S. M. (1997). The Foundations and Assumptions of Technology-Enhanced Student-Centered Learning Environments. *Instructional Science*, 25(3), 167-202.

Jenkins, T. (2002). On the Difficulty of Learning to Program. *Proceedings of the 3rd Annual Conference of LTSN*, Centre for Information and Computer Science, Loughborough, UK.

Johnson, D. H. & Caristi, J. (2002). Using Extreme Programming in the Software Design Course. *Computer Science Education*, 12(3), 223-234.

Karlesky, M. & Voord, M. V. (2008). Agile Project Management (or, Burning Your Gantt Charts). *Proceedings of the Embedded Systems Conference Boston*, Boston, MA.

Kim, J. & Booth, A. (2015). Rethinking Blackboard: Teaching Models for Interactive Learning. *Proceedings of the 13th APacCHRIE Conference*, Auckland, New Zealand.

Ma, L., Ferguson, J., Roper, M., & Wood, M. (2011). Investigating and Improving the Models of Programming Concepts Held by Novice Programmers. *Computer Science Education*, 21(1), 57-80.

Nicol, D. J. & Macfarlane-Dick, D. (2006). Formative Assessment and Self-Regulated Learning: A Model and Seven Principles of Good Feedback Practice. *Studies in Higher Education*, 31(2), 199-218.

Norman, D. A. & Spohrer, J. C. (1996). Learner-Centered Education. *Communications of the ACM*, 39(4), 24-27.

O'Neil, G. & McMahon, T. (2005). Student–Centred Learning: What Does It Mean for Students and Lecturers? In G. O'Neill, S. Moore, & B. McMullin (Eds.), *Emerging Issues in the Practice of University Learning and Teaching* (27-36). Dublin, Ireland: All Ireland Society for Higher Education (AISHE).

Pintrich, P. R. (2000). The Role of Goal Orientation in Self-Regulated Learning. In M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of Self-regulation* (451-529). San Diego, CA: Academic Press.

Pintrich, P. R. & Groot, E. V. D. (1990). Motivational and Self-Regulated Learning Components of Classroom Academic Performance. *Journal of Educational Psychology*, 82(1), 33-40.

Project Management Institute. (2017). *What is Project Management?* Retrieved from http://www.pmi.org/about/learn-about-pmi/what-is-project-management.

Reichlmayr, T. (2003). The Agile Approach in an Undergraduate Software Engineering Course Project. *Proceedings of the 33rd Annual Conference Frontiers in Education*, Westminster, CO.

Reiss, G. (2007). *Project Management Demystified* (3 ed.). London, UK: Taylor & Francis Ltd.

Shute, V. J. (2008). Focus on Formative Feedback. *Review of Educational Research*, 78(1), 153-189.

Tengberg, L. G. W. (2015). The Agile Approach with Doctoral Dissertation Supervision. *International Education Studies*, 8(11), 139-147.

Urdan, T., Anderman, L. H., Anderman, E., & Roeser, R. (1998). The Development and Validation of Scales Assessing Students' Achievement Goal Orientations. *Contemporary Educational Psychology*, 23(2), 113-131.

Watson, C. & Li, F. W. B. (2014). Failure Rates in Introductory Programming Revisited. *Proceedings of the Conference on Innovation Technology in Computer Science Education (ITiCSE '14)*.

Williams, L. & Upchurch, R. (2001). Extreme Programming for Software Engineering Education? *Proceedings of the 31st Annual Conference Frontiers in Education*, Reno, NV.

Woodward, C. J., Cain, A., Pace, S., Jones, A., & Kupper, J. F. (2013). Helping Students Track Learning Progress Using Burn Down Charts. *Proceedings of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Kuta, Indonesia.

Yorke, M. (2003). Formative Assessment in Higher Education: Moves towards Theory and the Enhancement of Pedagogic Practice. *Higher Education*, 45(4), 477-501.

Young, M. R. (2005). The Motivational Effects of the Classroom Environment in Facilitating Self-Regulated Learning. *Journal of Marketing Education*, 27(1), 25-40.

Zimmerman, B. J. (1990). Self-Regulated Learning and Academic Achievement: An Overview. *Educational Psychologist*, 25(1), 3-17.

Zimmerman, B. J. (1995). Self-Efficacy and Educational Development. In A. Bandura (Ed.), *Self-Efficacy and Changing Societies* (202-231). New York: Cambridge University Press.

AUTHOR BIOGRAPHIES

Tanya Linden is a Lecturer in Information Systems at the School of Business, Swinburne University of Technology. She has a diverse educational background with qualifications and expertise that cover developing the technical side of information systems, web design, and HCI, as well as teaching and learning in higher education. She is the recipient of numerous teaching awards from the universities where she has worked. Her research interests are in e-learning, educational technologies, and multimedia development practices. She is well-published in all these areas and regularly presents her work throughout North America, Europe and Australasia.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 2574-3872