Developing Agile Skills in IT Courses: Perspectives and Progress

Emre Erturk
Eastern Institute of Technology
eerturk@eit.ac.nz

Kathryn Mac Callum
Eastern Institute of Technology
kmaccallum@eit.ac.nz

ABSTRACT

Teaching at tertiary institutions implies a responsibility to provide an experience to the students that resembles and prepares them for the professional world. Agile software and systems development methods are increasingly prevalent in IT and other companies. Furthermore, modern teaching practices involve game like and collaborative activities in the classroom. This is also in line with the agile development philosophy as agile software projects have been early adopters of active learning, communication, and team work. This paper showcases agile practices and activities that were implemented earlier this year in a New Zealand tertiary institution in two distinct IT courses, by two different lecturers. The paper outlines the motivations and innovative ways that agile practices can be implemented within an IT programme.

Keywords: information technology education, agile development, agile methods, collaborative learning, group work, industry practices

1. INTRODUCTION

With more organizations adopting an agile approach to software development, it has become important that students are exposed to these practices within their IT programme. Agile methods also lend themselves nicely to supporting team work within an educational course. The concepts of group work, self-management, and recognizing individual abilities are all important qualities in a tertiary programme of study.

This paper describes two interesting but different ways to use agile practices in an IT degree programme at a New Zealand tertiary institution. The first case study describes how students were introduced to agile practices within a systems analysis course, while the second case study describes the adoption of the practices within a User Experience/User Interface course where agile practices were wrapped around a course assessment. The two cases had slightly different approaches as to how agile practices were adopted. The motivation in both cases was to start exposing students to practices that they may encounter in industry once they have completed their degree and to start embedding these practices into assessments to help support group work and encourage self-management.

2. LITERATURE REVIEW

The Agile approach to software development has been popularized since the Agile Manifesto of 2001. Agile and related so-called lightweight methods have been in contrast with the older and structured software development methodologies. Agile may not even be considered a methodology, since it does not prescribe development phases, long term planning, and specific types of documentation. Instead, Agile consists of general principles. Scrum, a popular framework associated with Agile, suggests roles that people can play in their project teams and activities they can perform together (Cockburn, 2013). As a result of this lack of prescriptive structure (and being more subtle and industry based), Agile is often not covered in depth in systems analysis and design courses at institutions of higher education. This is confirmed by Burns' survey of 172 systems analysis and design instructors (2011), where 93% of instructors covered either traditional waterfall or object oriented, or both, as opposed to only 7% who covered any of the other approaches including Agile. Furthermore, Scrum was covered by just 12% of the instructors. However, the concept of prototyping was taught by 80% of the instructors (Burns, 2011). Aside from prototyping, IT courses still need improvement in order to keep up with and cover agile approaches and frameworks.

Although a greater coverage of the agile approach is beneficial for IT students, it is also important to understand the type of project settings that are better suited to using this approach in industry, the potential limitations of this approach, and which particular agile industry methods and activities are relatively more important for teaching within courses. In an industry survey with 2,229 respondents, Doyle et al. (2014) found that agile projects are relatively common in web development as well as for developing in-house software solutions. Furthermore, approximately one half of the agile projects in the survey were conducted in small teams of a maximum of ten members, while teams of 11-25 and teams with more than 26 were less common. Doyle et al. (2014) also found that the agile survey participants generally felt more confident about being productive and being able to deliver functionality quicker and relatively less confident about being able to create greater economic value. Furthermore, popular agile practices included daily or standup meetings, and ability of individual members to negotiate and choose which tasks to work on. These points may also have certain implications for IT lecturers. For example, agile methods may be best learned within smaller projects. Collaborating in small teams is also suitable for an academic setting, where students can complete assignments in small groups, using agile activities. This is realistic and beneficial for teaching in the New Zealand context, where 97% of enterprises have fewer than 20 employees (http://www.med.govt.nz) whose IT projects are also compact.

In addition to preparing for jobs in the future, there are pedagogical and personal development reasons why agile practices in courses may be good for IT students. In their paper on agile software development methods in IT courses, Read et al. (2014) concluded that students had fun, and enjoyed a rich classroom experience. The main argument of the same paper is also very informative; agile activities help develop students' skills in entrepreneurship and innovation.

This quality assured paper appeared at the 6th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2015) and the 28th Annual Conference of the National Advisory Committee on Computing Qualifications, Queenstown, New Zealand, October 6-9, 2015. Michael Verhaart, Amit Sarkar, Rosemarie Tomlinson and Emre Erturk (Eds).

There are a number of reasons for this: increased confidence at the end of the course, receiving further mentoring through a scrum product owner, and improved problem solving skills. During scrum, students frequently broke down work into smaller tasks (i.e. user stories or requirements) and work on clearly defining the nature of each individual task, hence improving their ability to tackle and solve bigger problems (Read et al., 2014). Then Ezequiel Scott et al. (2014) went further and examined whether or not the particular learning styles of individual students influenced their initial experience and success with using scrum. For example, the so-called sequential students may have an advantage (especially in a developer role) because they are rather comfortable performing their work step by step and quickly, in the presence of incomplete and evolving information. Game-like activities are sometimes used during the software development process, particularly under the agile approach (Parsons, 2014). Likewise, game-like activities in the classroom contribute positively to student engagement and teamwork. These activities can also help develop useful high level skills such as analysis and estimation (Parsons, 2014).

3. IMPLEMENTATION IN COURSES

In 2015, student numbers increased significantly and this highlighted a need to approach IT courses in a more effective manner. This resulted in course work becoming more group focused and needing to cultivate students to become more self-managing. At the same time, it was important that these courses maintain their close alignment to current industry practices, such as the rise in agile approaches being embedded in many organisations. As a result, the User Interface Design (UID) and Systems Analysis and Design (SA&D) courses were redesigned to adopt agile software development methods where teamwork formed the core learning approach.

3.1 Agile Activities in SA&D

This section describes how three well-known agile activities were implemented as learning activities in a New Zealand tertiary institution in the first half of 2015, in a systems analysis and design course with more than 30 students.

Planning Poker: This is an agile estimating and planning technique, and an example of a game like activity. For each poker planning round, the lecturer played the role of the product owner introduced a different user story, requirement, or feature to the class. One of the purposes of the game is for team members to discuss and negotiate how much time each feature would take to develop. In this course, the students played the game in teams of three. For the estimations, the students used cards prepared and printed by the lecturer. These cards had values of 1, 3, 5, 8, and 13, in terms of so-called user points (agreed upon units of time). After discussing the user story, the estimators privately selected their cards and then revealed them at the same time. During the session, the results were collated by the lecturer on an ongoing basis on a spreadsheet. At the end, it was interesting to look at and discuss the results together as a class. The students also evaluated their own software development estimation skills and provided feedback as to what they thought of this activity. The feedback showed that this successful classroom activity can be repeated in the future.

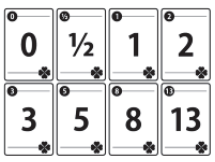

Figure 1: Agile Poker Cards
Reproduced under the Creative Commons License

Next, the Planning Game illustrates the concept of balancing velocity and quality over successive iterations. This involves a simple repetitive task. The participants need to track the number of units of work done for each of these iterations. As they go on, they compare the numbers, and try to improve performance in the next iteration(s), and also improve their ability to make better estimations as a result of better understanding how many units of work are feasible at any given level of quality. In this course, the lecturer created an authentic variant of this otherwise common agile game. In previous contexts in industry or literature, this game was played with physical objects such as paper hats or Legos as a fun way of teaching agile skills, outside of the software development process itself. In this course's variant, the game was completely electronic and did not require any physical objects or stationery.

The task was to make up sample database records, and each student worked independently, using a Google Sheets template provided by the lecturer. The types of records were customer, employee, or asset records, which were also relevant for the students' main assignment in the course. The students also worked in teams of three; and each team shared a Google Spreadsheet file, which had multiple tabs, one for each member. It must be noted here that using a Google Doc format made it very easy to replicate and create new files for each team. It was also very convenient to share and edit the same file in a live environment. These online files were then shared with the lecturer for reviewing and archiving.

Students had a time limit of 180 seconds (3 minutes) per iteration; the lecturer used the smartphone's stop watch for starting and stopping them. There was a break between each round, for discussions, clarifications, and comparing the results between the teams and against previous iterations. At the beginning of the game, the lecturer spent time to provide initial instructions to introduce the students to the game.

In the first iteration, students were asked to create as many as they could, without cautionary advice. As a result, some of the records were nonsensical, unrealistic, or of low quality. This was discussed with the students in a lighthearted way. In the second iteration, they were asked to do the same exercise but try to make the names, emails, and other sample data more realistic. Before the third iteration, the students were asked to write down an estimate in terms of how many they could create next time. In the third iteration, they were also hinted to use the data of people they know or of celebrities to increase the numbers, while keeping their sample records diverse and original. At the end of the game, the class made conclusions. The informal feedback from the students was positive, indicating that they enjoyed the session.

Daily Standup Meeting: This particular type of meeting was practiced several times on an ad hoc basis. During the course itself, the students worked in teams for their project assignment and already held regular meetings. However, having a conversation while standing up was a fresh experience for them. Although the standup meetings were limited to five minutes, the discussions were more active and to the point, as opposed to sit down meetings where students might stray off topic or become distracted by their laptops. Before doing a standup meeting in class, the students were shown a short video as an example of how to perform this. There are some general guidelines for a standup meeting; participants take turns to discuss what each member did yesterday, what they will do today, and what impediments they may be encountering. As they may encounter standup meetings in their future jobs, the benefit of this exercise is that students have been familiarized with the concept.

3.2 Agile User Interface Design

This section describes an assessment embedded approach taken to adopting agile practices with a course. User Interface (UI) Design is a second year course that covers principles and techniques of graphical interface design. It includes traditional Human-Computer Interaction (HCI) and usability techniques and concepts but also takes a wider perspective of supporting an effective user experience (UX), a relatively new and debated term (Law et al., 2006). The course focused on user-centred design (UCD) principles and incorporated UX qualities, e.g. aesthetics, affective and emotional aspects, and the nature of experience. The assessment took a project-based approach; students applied HCI and UX techniques and principles, in order to evaluate and design effective interfaces.

Within the UX industry, the adoption of agile software development approaches is becoming increasingly common. Students who wish to go into these sorts of industries are more likely to be involved within organisations that have adopted an agile approach. The principles of agile methods are strongly aligned with the aims of UX as they both have a focus on building quality software experiences (Da Silva et al., 2012). Despite this alignment of principles, the methods and practices of agile development can still be difficult to combine with UX tools and practices (Budwig et al., 2009). However, there are a number of studies that address how agile and UX can be successfully integrated (Da Silva et al., 2011).

Therefore, the lecturer used this course as an opportunity to introduce students to some of the concepts and approaches of agile development from the industry. The intent was not to fully emulate Agile UX/UI practices but rather give students a chance to see how these practices could be embedded in the development of a system. The main focus of the course was still on learning and experiencing UX/UI methods, while they also experienced how agile methods may support their work.

In addition to giving students an opportunity to experience scrum in a real life situation, the agile approach also enabled the lecturer to cope with increasing enrolments. Due to the class size, it was not feasible for students to work individually in the project. Instead, students had to work in self-managing groups. The agile framework helped to make this happen.

The assessment for this course was therefore redesigned into a group assessment where students were to work alongside an industry client to redevelop a website and develop a mobile application for a specific system. Rather than two distinct assessments (with the first one requiring students to evaluate an interface and the second to design/redesign an interface), the new single assessment this year comprised of five sprints. Students were placed into four self-managed teams, which consisted of 6 or 7 members. Each team investigated and developed a different smaller part of the same general system.

Each sprint in the assessment covered one of the core UX/UI techniques. A modified version of Nooder & Nielson's (2009) integrated life-cycle was adopted here, as illustrated below.

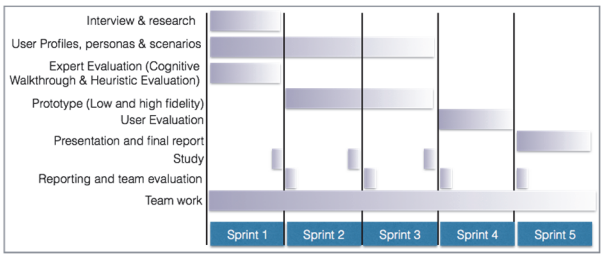

Figure 2: UCD/UX within the Agile Development Cycle Adapted from Nooder & Nielson (2009).

Sprint 1 (Research): The first assessment required teams to understand the users, user needs, and activities. User profiles, personas and scenarios were developed and used to evaluate the current system through cognitive walkthroughs and a heuristic evaluation to identify issues and propose improvements in the current system.

Sprint 2 (Low-fidelity Prototype): A wireframe prototype, using Balsamiq (https://balsamiq.com/), was created to start designing the new system. Students worked in sub groups in parallel design. If different versions were developed, then they would be compared to determine the best solution.

Sprint 3 (Hi-fidelity Prototype): Based on the initial wireframes, a hi-fidelity prototype was developed. The focus was not on the coding of the prototype, but rather the design aspects of the system. Prototyping software such as InVision and JustinMind were used to develop these prototypes.

Sprint 4 (User Evaluation): Students carried out user testing of the hi-fidelity prototype. The users in this case were students acting as potential users. Users were asked to carry out tasks and provide feedback on their experience through the think-aloud protocol and a survey.

Sprint 5 (Presentation and Final Report): Students wrote up their final report and presented their work design to the client.

Within each sprint agile practices were adopted. In particular, a Scrum framework was adopted. For the group assessment, the roles of product owner, development teams and scrum master were adopted. The product owner was undertaken by the lecturer of the course. This role involved setting the project vision, and supporting the students to determine and fulfil the stakeholders' interests. The product owner also maintained the product backlog, which outlined the requirements of each sprint.

The teams comprised of students with different skills and backgrounds. Within a team, the students could decide which tasks individuals members wish to undertake. After an agreement with others, these task assignments would be recorded in the sprint backlog. These tasks generally aligned with the individual students' abilities and interests. The teams were expected to be self-organising and managing, and to strongly collaborate throughout the sprint.

Each team appointed a different scrum master in each sprint. The role of the scrum master was to support and facilitate the scrum process. They would look after the sprint backlog and ensure that the other team members met the time allocations required of them.

The product owner provided the product backlog. This was defined at a high level, and formed from the assessment requirements. From this product backlog, specific tasks were identified as to what needed to be done. These were recorded in the sprint backlog. Each member would then select which tasks they wished to undertake. On the backlog, the individual team members would also indicate the proposed time/date for completing their tasks. As the project progressed, the backlog was updated to indicate the status of each task (not started, started, complete) and the actual time/date completed. At the start of every tutorial session, students were given time to hold a planning meeting to discuss and manage their backlog.

4. CONCLUSIONS

The implementations of agile methods in these two IT courses are not limited to what has been described so far. For example, functional prototyping has also been conducted in the systems analysis and design course over a two week sprint, using product backlogs, and task boards. Both courses

have involved innovative uses of technology and electronic templates, which have not been fully described in this paper. As all of the work has the potential to spawn multiple research outputs, this paper only carries out a specific aim, i.e. allowing the readers to compare and contrast two different approaches. Therefore, in the conclusions, it would be suitable to highlight the general benefits obtained from each approach.

From the perspective of constructive alignment, embedding the new agile related content into the assessment can ensure that students retain the information and perform these methods in order to succeed in the course. For broad perspective courses that familiarize students with various different approaches, game-like activities may be a gentle way to teach students the new content at any convenient time throughout the semester. Effective learning design involves the consideration of both the fun aspects and the academic assessment expectations. Thus, it is recommended that instructors consider both perspectives when introducing new and innovative concepts in their courses.

Recent studies, notably by Kropp et al. (2014), on similar implementations of teaching agile development practices have discovered that, although students start rather slowly, they become more productive and enjoy this approach over time. The same outcome was observed during the two courses in this paper. However, two aspects were different. First, the course observed by Kropp et al. (2014) in Switzerland was focused solely on agile software development, rather than general systems analysis and design or user interface design. Secondly, as a result of this, the Swiss students had to overcome two challenges at the same time, namely group formation and learning agile practices. In the two courses described here, the students had already formed their groups and had become relatively accustomed to working together. Therefore, in this paper, communication between team members has not been found to be a significant factor.

This paper's contribution to the literature is significant as existing publications on teaching agile methods are still recent and emerging. There are many interesting aspects that can be covered by researchers in the future. For example, a detailed coverage role of the software technology used by agile and scrum participants may also inform industry professionals. Finally, a critical evaluation of the deliverables produced during agile-oriented courses, can assist IT instructors in the continuous refinement of their assessments and projects.

5. ACKNOWLEDGMENTS

The authors would like to thank Ms. Sharon Chapman, the owner of ABC Software based in Hawke's Bay. She was the industry sponsor of the real-life project during User Interface, which is one of the two courses discussed in this paper.

6. REFERENCES

Budwig, M., Jeong, S., & Kelkar, K. (2009). When User Experience Met Agile: a Case Study. ACM *CHI'09 Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 3075-3084).

Da Silva, T. S., Martin, A., Maurer, F., & Silveira, M. S. (2011). User-Centered Design and Agile Methods: A Systematic Review. *Agile 2011 Conference Proceedings* (pp. 77-86).

Da Silva, T. S., Silveira, M. S., Maurer, F., & Hellmann, T. (2012). User Experience Design and Agile Development: from Theory to Practice. *Journal of Software Engineering and Applications, 5*(10), 743-751.

Doyle, M., Williams, L., Cohn, M., & Rubin, K. (2014). Agile Software Development in Practice. *Agile Processes in Software Engineering and Extreme Programming, Springer Link Lecture Notes in Business Information Processing, 179*, 32-45.

Kropp, M., Meier, A., Mateescu, M., & Zahn, C. (2014). Teaching and learning agile collaboration. Proceedings of the IEEE 27th Conference on Software Engineering Education and Training (CSEE&T) (pp. 139-148).

Law, E. L. C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009). Understanding, scoping and defining user experience: a survey approach. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems* (pp. 719-728).

Parsons, D. (2014). Creating game-like activities in agile software engineering education. *23rd Australasian Software Engineering Conference)*. Retrieved from http://www.massey.ac.nz/~dpparson/Creating%20game-like%20activities%20in%20agile%20software%20engineering%20education.pdf.

Read, A., Derrick, D., & Ligon, G. (2014). Developing Entrepreneurial Skills in IT Courses: The Role of Agile Software Development Practices in Producing Successful Student Initiated Products. *Proceedings of the IEEE 47th Hawaii International Conference on System Sciences* (pp. 201-209).

Scott, E., Rodriguez, G., Soria, A., & Campo, M. (2014). Are learning styles useful indicators to discover how students use Scrum for the first time? *Computers in Human Behavior, 36*, 56-64.