Adopting an Agile Approach for Reflective Learning and Teaching

Eleanor Leist
E.Leist@uea.ac.uk
University of East Anglia
Norwich, Norfolk, UK

Jaejoon Lee
jaejoon.lee@uea.ac.uk
University of East Anglia
Norwich, Norfolk, UK

ABSTRACT

Software engineering is concerned with how best to create software in ways that promote sustainable development and maximise quality. We have been largely successful at transferring software engineering knowledge into the industry, however, many challenges in software engineering training remain. A key amongst these is how best to teach practical engineering approaches along with the theoretical concepts behind them.

This paper describes our experience of adopting an agile approach for reflective learning and teaching within the context of our Software Systems Engineering module, aimed at addressing challenges identified with previous efforts to promote reflective practice. Our study attempts to strengthen the use of reflective learning approaches for our current cohort, as well as introducing reflective teaching practices, whereby we examine our teaching approach in order to improve its efficiency and effectiveness. Our analysis of student response to the module shows that it was very well-received by the students, and we were able to collect ample evidence from feedback to support this. Most of our approaches resulted in positive feedback and contributed to improvements in teaching quality, however, we also identified some key aspects in our method that could still benefit from refinement, such as the need for explicit links between learning outcomes and workshop activities, and intuitive design of feedback questions, along with feedback collection frequency. We plan to incorporate these additional updates into the revision of the module for the next academic year, and to continue collecting and analysing feedback data for further enhancement.

CCS CONCEPTS

• **Software and its engineering** → *Software development process management*; • **Social and professional topics** → **Software engineering education**; **Model curricula**.

KEYWORDS

Reflection, Learning, Teaching, Agile, Feedback, Gamification

ACM Reference Format:
Eleanor Leist and Jaejoon Lee. 2024. Adopting an Agile Approach for Reflective Learning and Teaching. In *ICSE'24: International Conference on Software Engineering, April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

ICSE'24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Today, software engineering is concerned with how best to create software in ways that promote sustainable development and maximise quality. There is a general consensus that we have been largely successful at transferring software engineering knowledge into the industry through various forms of education [14, 20]. However, many challenges in software engineering training still remain. One key challenge is how to best teach practical engineering approaches along with the theoretical concepts behind them.

Lab- and workshop-based approaches, aiming for learning-by-doing, have been widely adopted for the teaching of both practical and engineering disciplines [5]. Likewise, software engineering teaching often incorporates lab and workshop sessions. Whilst such interactions with students can give them hands-on experience, understanding their level of learning attainment can be challenging. One possible approach for assessing students' achievements of learning outcomes is reflection [2], where students are asked to review the work they did in order to re-digest the core ideas (e.g., techniques, theories, etc.).

This paper centres around a Software Systems Engineering [1] module where, in recent years, we have incorporated reflective learning into our teaching practices, albeit in an implicit manner. For instance, we have routinely revisited the main content of lectures or labs at the end of the sessions, offering students a reminder of the learning outcomes. In other instances, we have organised additional revision sessions for the students to prepare for exams and/or coursework submissions. These approaches are in keeping with reflective learning practices, however, the actual reflection component was mostly left for students to consider by themselves, without the proper mechanisms to support the reflection happening. Consequently, the potential benefits of reflective learning were not fully recognised by the students.

In addition to reflective learning, we also wanted the module to incorporate reflective teaching, by examining our teaching approaches in order to improve their efficiency and effectiveness. Prior to the module changes described in this paper, the reflective teaching process centred around the collection of student feedback at the end of each semester. However, the time frame of this feedback implies that practical improvement happens annually, and the long interval between data collection (i.e., receiving student feedback) and implementation (i.e., revision of teaching methods) hinders the accurate adjustment of teaching approaches for improvement

[1] The current module name at our University is 'Systems Engineering' to distinguish it from the Software Engineering module, where the focus is on software system implementation. In this paper, we refer to the Systems Engineering module as 'Software Systems Engineering' to avoid any confusion with hardware centred modules.

and enhancement. Moreover, due to the varying characteristics of each cohort on the module (e.g., number of registered students, student composition with respect to their degree courses (e.g., BSc Computing Sciences, BSc Business Information Systems, etc.), and the background knowledge of the students (e.g., whether or not they have taken some optional modules, and the differences in their compulsory modules)), makes it difficult to understand whether the potential improvements would practically translate to the following academic year.

This paper describes our experience of adopting an agile approach for reflective learning and teaching within the context of our Software Systems Engineering module, aimed at addressing the challenges identified above. Our paper presents results from a one year study, including individual aspects of the reflective learning and teaching experience, such as how it fosters reflective practice, the student response to our methodology, and what the practical difficulties are for implementing our approach. In this paper we provide a comprehensive description of how our approach was designed and put into practice. The key contribution of the paper, therefore, is a concrete description of our agile reflective learning and teaching practices, which others, thinking of taking a reflective approach, can use in designing and developing their own modules.

The rest of this paper is organised as follows: Section 2 reviews related work and explains the motivations for our teaching approach. Section 3 describes our approach in detail, including the module structure and an illustrative process model. In Section 4, we discuss the student feedback collected during the process, and in Section 5 we summarise the key lessons learned from this process. We conclude our paper in Section 6 with future work.

2 BACKGROUND AND MOTIVATION

2.1 Related Work

2.1.1 Reflection. The use of reflection for teaching and learning in higher education is a well established concept, with many studies exploring the benefits it can offer. From a student perspective, reflective learning is "a process that leads to reflection on all sources of knowledge, including personal sources and experiences which may contribute to understanding a situation" [6]. It can help students to "turn experience into learning" [1], by analysing and responding to an experience, such as a lecture, then processing this thinking to formulate evaluations and decisions about what has been learnt. The inclusion of reflection as part of the teaching process allows a dialogue to form between the teacher and learner, so that the educator becomes a 'facilitator of learning', focusing on how learners are interacting with the material, rather than how it is being transmitted [3]. Boud et al. [1] state that such reflective activities should take place very frequently, even on a daily basis.

Reflection can be an equally useful tool from a teaching perspective. Schön [18, 19] presents the idea of the 'Reflective Practitioner', whereby education professionals apply 'reflection-in-action' techniques, with foundations in design, to improve their own practice. Many of the decisions educators make regarding their teaching approach is based on assumptions they have formed, and critical reflection of these decisions helps educators to check if these assumptions are valid [4].

More recent studies have applied the idea of reflective practice within a computer science context, particularly when exploring the human aspects of software engineering [12]. This study describes a framework for applying design studio techniques introduced by Schön [19], in order to encourage tutor-student interaction and collaboration. In a wider software engineering context, reflection can be highly useful for informing decision making and improving current processes [9], and thus is a valuable skill for students of this discipline to learn.

2.1.2 Agile teaching. Reflection and agile development are closely linked concepts. A key aspect of the most widely used agile development approach, Scrum, is the 'Sprint Retrospective', whereby members of the Scrum team discuss what went well and what could be improved about the sprint.

Krehbiel et al.[15] proposed the idea of an Agile Manifesto for Teaching and Learning , with two of the key values being particularly relevant. The first of these values is 'Student-driven Inquiry Over Classroom Lecturing', which promotes student empowerment through active learning and encourages them to use their own voices. The second, 'Continuous Improvement Over the Maintenance of Current Practices', encourages educators to frequently evaluate their teaching practices and to adopt a trial and error approach to exploring new ideas. An example of applying these values in practice includes the use of retrospectives, offering the students the opportunity to reflect on previous courses [15].

Other contexts in which agile principles have been applied to learning and teaching include the use of Scrum techniques when teaching agile project management in a business school [7], and the proposal of 'Extreme Pedagogy', where four core values based on Extreme Programming are used to enhance collaboration and facilitate learning by continuous doing [8].

2.2 Motivation

The undergraduate software engineering related modules are structured over three years[2]. The Software Systems Engineering module, which is the main topic of this paper, builds on the knowledge and experience gained in Years 1 and 2 of study that introduce students to:

- **Year 1**: Programming and testing, software life cycle models
- **Year 2**: Requirements engineering, software design and design patterns

In Year 3, the focus is on the Software Systems Engineering module.

The Software Systems Engineering module draws together a wide range of material and considers it in the context of developing modern large-scale computer systems. Topics such as Systems Thinking, Outsourcing, Quality, Risk Management, Measurement, Project Management, Software Process Improvement, Configuration Management, Maintainability, Testing, and Peopleware are covered in the module. This module is supported by well-documented case studies and includes guest speakers from the industry.

The teaching method is based on lectures and workshop sessions. The major interaction with students is organised through the workshop sessions: prior to our changes to the teaching approach,

[2]In the UK, the undergraduate study for a Bachelor's degree spans three years and, therefore, Year 3 is the final year before the student graduates from university.

students were expected to read specified papers each week, before discussing the content using some key topics or discussion points shown at the beginning of the workshop session. This approach has been used for the last 10 years of the module and was generally well-perceived by students. In particular, the student and teaching staff interactions during the workshop sessions were appreciated by the students as evidenced in the end of semester student feedback.

Despite the general positive feedback, there were some challenges we recognised in terms of using student feedback for improving their learning experience. They are:

- Students' low motivation for providing end-of-semester feedback: It does not offer any tangible benefit to the students who provided it, and they may have difficulty recollecting what happened during the early stages of the semester.
- Low number of responses to school-led feedback: Low motivation, as highlighted in the previous item, and the low engagement (attendance) toward the end of semester due to various reasons (e.g., multiple coursework deadlines, exam preparation, etc.).
- Standard set of questions without specifics about each module: A question, for example, about promptness of coursework score return for a module with only one piece of coursework at the end of semester is not relevant.

More importantly, these questions are designed to collect feedback on the quality of teaching, not for students to reflect on their learning outcomes. Such reflection to improve student attainment should be embedded into the module and customised to fit for each module's learning outcomes. Based on these observations, we have revised the module to include reflective learning practices with a rapid feedback cycle. The details of our new module design and its results are described in the following section.

3 TEACHING METHOD DESIGN

We first describe the process model of our approach, and the detailed module structure is also illustrated in this section.

3.1 Agile Process for Reflective Learning and Teaching

The main idea that we aimed to incorporate into the module design was reflective practice techniques [1, 2, 16], such as coaching and critiquing. In addition, the reflection should happen within a short time-frame so that the students could reinforce their learning when each topic was covered, rather than revising them at the end of the module. It is therefore important to define the agile reflective process to enable such practice in the first place. Figure 1 depicts the process model we followed throughout the module. The activities involved are explained below.

The process begins with refining learning outcomes for the upcoming week. We do have a set of pre-defined general learning outcomes, but we should refine them for each week based on the cohort specific information (e.g., cohort size, students' background, etc.) and inputs from previous week, if available. For instance, a general description of 'demonstrate knowledge of feature modelling' is refined as 'demonstrate knowledge of feature identification within a mobile game domain' to reflect the scenario of that week's session.

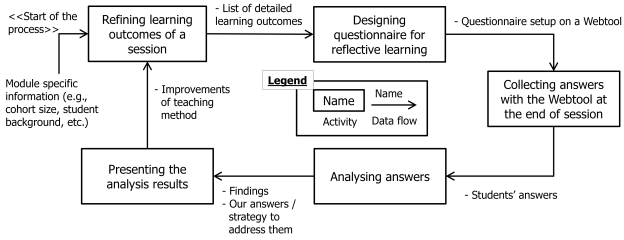

Figure 1: Agile process for reflective learning and teaching

Once we have the refined learning outcomes, questionnaires for the week are designed which encourage the students to revisit and reflect on the main topics. In the case of feature identification, for example, the questions may include 1) 'Which activity was most useful for identifying features?' and 2) 'Which features were most difficult to identify?'

Based on the refined learning outcomes, we revise lecture slides and also adopt reflective practice techniques into the workshop sessions, whenever applicable, in addition to the reflective questionnaire practice. After we teach the students through lectures and workshops, the students are the offered the chance to answer these questions via Padlet [3]. Padlet was chosen as the feedback mechanism due to it being free to use and relatively easy to interact with. Virtual walls can be created, with a reasonable amount of freedom in terms of design choices. It offers anonymity, or the option to respond with a name, and students can access it by simply scanning a QR code or following a short URL. Padlet has been used successfully for motivating and engaging students in a number of other studies [10, 13, 17].

Moreover, using the feedback mechanism, we also collect student feedback on the teaching methods used that week. For example, we may include questions about the quality of workshop-based teaching (e.g., 'How easy was it to understand the workshop organisation?', 'Which workshop topics need be revisited for next week?', etc.). We can then analyse the collected feedback to see the level of student attainment (i.e., quality of students' answers related to learning outcomes) and to understand the points for improvement in our teaching methods.

The findings from the analysis are summarised and presented in the next lecture or workshop session (see the 'Presenting the analysis results' activity in Figure 1.). During this activity, students may ask further questions and/or provide additional feedback. At the same time, we also make improvements to our teaching approach for the next session, and check the effectiveness of this through the questions of the subsequent session.

Fundamentally, we put considerable thought into getting students to reflect on their practice by embedding 'reflective practice' [11] into the curriculum. This was done through a variety of means – peer assessment, constant critique from teaching staff, coaching, and individual feedback collection. This was challenging because it required a change of culture in the students, who were not used to reflecting on their practice in this way.

[3] www.padlet.com

3.2 Module Structure and Topics Covered

The module runs for one semester over twelve weeks and its exam is scheduled separately during the university's summer assessment period. For each week, there is a two-hour lecture and a one-hour workshop session. Where possible, we also invite external guest speakers to share their industry experiences relevant to the topic of that week with the students. The weekly topics and brief descriptions of lectures, guest talks and workshops are explained in the following:

- **Week 1**: Introduction to the Module and Systems Thinking
  - Lecture: The module structure, assessments and our teaching methods are explained. Also, the basic concept of systems thinking is introduced.
  - Workshop: Activity in a small group on analysing a given topic using a Causal Loop Diagram.
  - Reflective practice: Questionnaire.
  - Outcomes: Students familiarise themselves with initial concepts for software systems engineering and understand how to use a Causal Loop Diagram.
- **Week 2**: Software Process Models and Feature Modeling
  - Lecture: Various software process models are introduced along with their pros/cons. As a tool for initial domain understanding, the feature modelling technique is also introduced.
  - Workshop: Feature modelling exercise within a mobile game domain.
  - Reflective practices: Questionnaire and coaching.
  - Outcomes: Students can select a software process model for a project and use a feature model to understand the project domain.
- **Week 3**: Project Management and Effort Estimation
  - Lecture: Concepts and practical techniques for project management are introduced with a focus on effort estimation.
  - Workshop: Effort estimation exercise for developing test cases with a mobile game domain.
  - Reflective practices: Questionnaire and peer assessment.
  - Outcomes: Students understand the core activities required for project management and can perform effort estimation systematically.
- **Week 4**: Software Quality: Risk Management and Measurements
  - Lecture: Concepts and practical techniques for risk management are introduced with a focus on measurements.
  - Guest lecture: 'Quality in Agile'
  - Workshop: Risk analysis exercise for risk planning with a coursework preparation scenario.
  - Reflective practices: Questionnaire, acting on a scenario and coaching.
  - Outcomes: Students understand the core activities required for risk management and can perform risk planning systematically.
- **Week 5**: Software Quality: Software Architecture
  - Lecture: Concepts and practical techniques for software architecture design.
  - Workshop: Architecture design and its quality implication exercise with a paper airplane development scenario.
  - Reflective practices: Questionnaire and gamification.
  - Outcomes: Students understand the critical implication of architecture design to achieve required quality.
- **Week 6**: Testing
  - Lecture: Concepts and practical techniques for testing with a focus on test case development.
  - Guest lecture: 'Software testing in practice'
  - Workshop: Test case development exercise by comparing/contrasting black box/white box approaches.
  - Reflective practices: Pair development and peer assessment. General mid-module feedback collected.
  - Outcomes: Students understand how to develop test cases with different approaches.
- **Week 7**: Software Process Improvement
  - Lecture: Concepts and practical techniques for software process improvements with an agile process.
  - Workshop: A ball-passing game is arranged to simulate an agile process improvement approach.
  - Reflective practices: Questionnaire, peer assessment and gamification.
  - Outcomes: Students understand the core factors for process improvement and how to put them into practice.
- **Week 8**: Software Maintenance and Configuration Management
  - Lecture: A trend of lengthening software maintenance phase is explained with case studies and practical techniques for configuration management to cope with the trend are discussed.
  - Guest lecture: 'Software Maintenance Challenges'.
  - Workshop: A system maintenance case study discussion is organised.
  - Reflective practices: Questionnaire and self assessment.
  - Outcomes: Students understand the significance of software maintenance and how to plan software maintenance.
- **Week 9**: Software Reuse/Software Product Line Engineering
  - Lecture: Concepts and practical techniques for software reuse with the focus on software product line engineering.
  - Guest lecture: 'Software product line engineering in a large organisation'.
  - Workshop: Software architecture trade-off analysis workshop for software reuse is organised.
  - Reflective practices: Questionnaire, acting on a scenario and gamification.
  - Outcomes: Students can understand the importance of software reuse and the critical role of software architecture.
- **Week 10**: Peopleware

  - Lecture: Concepts and practical techniques for managing a team and team dynamics.
  - Guest lecture: 'How companies keep employees happy and productive'.
  - Workshop: Technical writing and verbal feedback on the abstraction of student coursework are organised.
  - Reflective practices: Questionnaire and individual coaching.
  - Outcomes: Students can understand how to manage the team dynamics. Also, they can improve their writing skills and their coursework through the workshop session.
- **Week 11**: Outsourcing and Re-engineering
  - Lecture: Concepts and practical techniques for outsourcing and re-engineering.
  - Workshop: Continued from previous week: technical writing and verbal feedback on the abstraction of student coursework are organised.
  - Reflective practices: Questionnaire and individual coaching. General end-of-module feedback collected.
  - Outcomes: Students can understand how the outsourcing works and when/how to re-engineering their software assets.
- **Week 12**: Revision lecture for exam preparation
  - Lecture: Key lecture contents are revisited in preparation for the exam.
  - Workshop: Continued from previous week: technical writing and verbal feedback on the abstraction of student coursework are organised.
  - Reflective practices: Questionnaire and individual coaching. General end-of-module feedback collected.
  - Outcomes: Students can understand the important part of the module contents for the exam.

Assessment: Marks are awarded for three different pieces of individual assessment. They are: 1) Conference paper (i.e., a short technical paper on a given topic), worth 40% of the overall mark. 2) End of semester examination during the summer exam period, worth 50% of the overall mark. 3) Workshop attendance and participation, worth 10% of the overall mark.

Workshops for coursework and exam: The workshops in weeks 10, 11 and 12 were organised to help the students prepare their coursework and exam. As the coursework submission deadline of Friday Week 12 approached, we observed that the students were quite distracted and found it difficult to focus on workshop activities. As such, we deliberately designed the workshops to give individual feedback on their abstracts for the conference paper coursework. The reflection on the lecture content of these weeks were covered by the weekly reflective questions.

Registered student composition: We had a total of 80 students registered in the module for this year. Of these, there were 69 undergraduate students, with most of these students taking a degree in Computing Science (50) or Business Information Systems (16), and a further three students on other undergraduate courses. Also we had 11 postgraduate master's students, all enrolled on a Master's in Computing Science. As a prerequisite, they all had experience of software system implementation and were familiar with software systems.

Number of students in each session: We could host all 80 students for lectures, but we had to divide them into three groups for workshop sessions, meaning we were running three workshop sessions after each lecture. We allocated 25-30 students to each workshop and found this number of students was manageable for all the workshop activities we had planned.

Teaching team: We had two academics in the teaching team (i.e., the two authors of this paper) and did not have any teaching assistants. Both academics were present for all workshop sessions.

In this section, we provided the details of the module structure and its running context. We focus on the feedback question design and the analysis results of the collected feedback data in next section.

4 STUDENT FEEDBACK

The following subsections discuss the types of questions presented to students using the feedback mechanism, and present the findings from both this feedback, and the school-organised feedback collected at the middle and end of the module.

4.1 Feedback Questions

A range of questions were presented to students each week, using the Padlet feedback platform, which allowed the teaching team to facilitate both reflective learning and reflective teaching. The feedback format each week generally consisted of two to three questions requiring written responses or comments, followed by a number of learning outcomes for the workshop session, with the option to 'upvote' or 'downvote' on whether the students felt that had achieved that learning outcome.

Examples of typical questions presented to the students can be found in Table 1. As seen in the first column, questions designed to encourage reflective teaching had a focus on the teaching method used, such as the effectiveness of examples given to students, and the usefulness of activities utilised during workshops. It was hoped that responses given to this type of question would help the teaching team to assess whether appropriate choices had been made when developing methods for delivering course content.

Questions aimed at fostering a culture of reflective learning instead focused more on the students' response to the content and activities. These questions typically encouraged students to reflect on, and to critique, their own thoughts and feelings about what they have gained from the workshops and lectures.

4.2 Weekly Reflective Feedback

Reflective student feedback was collected by the teaching team using the Padlet platform for each week of teaching on the module. In weeks 6, 11 and 12 of the module, more general mid- and end-of-module feedback was collected, alongside the formal feedback process implemented by the School, and results of these weeks are therefore not included in the totals.

The questions presented to students prompted responses including free-text comments, ratings, and up or down votes. A typical example of a Padlet wall can be seen in Figure 2. The Padlet settings were altered such that students had the ability to add comments to,

Table 1: Examples of questions presented to students via the feedback platform

Examples of reflective teaching questions	Examples of reflective learning questions
What could be improved about the abstract feedback sessions?	How confident would you be applying effort estimation to your own project? (1-5 scale, 1=not confident at all, 5=very confident)
Which item/activity did you find most helpful for understanding the concept of feature modelling? (e.g. lecture slides, car example)	Which part of the risk management exercise was most challenging? (e.g. identifying, deciding on a strategy)
How interesting did you find the guest talk this week? (1-5, 1=not interesting, 5=very interesting)	Considering providing and receiving feedback from others, which of these did you find most helpful?
What was the most useful element for understanding 'Architecture'? (e.g. Vasa ship, ATAM, BMS, airplane exercise)	"Structure decides quality". Do you agree with this statement? Why?
Did you find the ball game activity helpful for understanding process improvement? (1-5, 1=not at all helpful, 5=very helpful)	ChatGPT: Friend or foe? What are your thoughts on the use of AI tools?
Have the sessions today helped you to feel more confident about the conference paper assessment?	Which of the 3 stakeholders (PM, marketer, user) did you side with after the debate exercise? Why?

or 'upvote' or 'downvote' existing posts. Although the reflective feedback process was presented as an optional task each week, many students chose to respond and engage with the process, with a minimum of 10 responses for each week of the module.

The teaching team manually categorised student responses according to whether they were valid and on-topic, with subcategories of 'positive invalid', 'neutral invalid' and 'negative invalid' for those that were deemed off-topic for a given question. As an example, when asked "Which aspects of the conference paper would you like further help with?", comments would be categorised as follows:

Valid: *"Structure, structure and analysis!!"*
Positive invalid: *"Very fun lecture"*
Neutral invalid: *"var check"*
Negative invalid: *"We lost so I'm mad."*

Table 2 shows the average number of valid and invalid responses given by students for each feedback question, and Table 3 provides a breakdown of the total responses for each week by response type.

It was noted that more invalid comments were received when students took part in a practical activity, whereas sessions involving a paper-based activity tended to receive a larger share of valid comments. This was particularly evident in week 7 (a ball game activity), which received the highest number of invalid positive responses, and week 9 (battlefield debate activity), which received the highest number of invalid neutral responses (see Figure 3). The results indicated high levels of engagement and enjoyment, but with potentially reduced focus on the learning outcomes. The decline in the number of positive comments towards the end of the module may be attributed to students experiencing a general decrease in

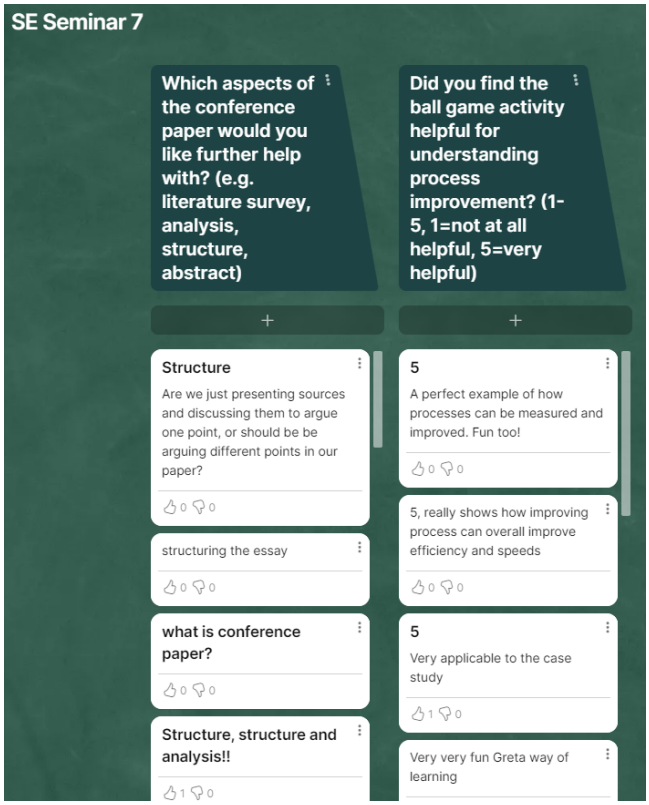

Figure 2: An example of a Padlet wall used to collect student feedback during a workshop

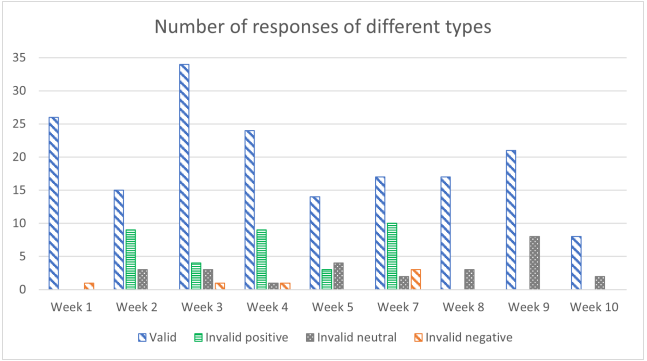

Figure 3: Total responses categorised by type

mood as they approach course deadlines, amongst other factors. An overall decline in total responses was observed over the duration of the module, perhaps indicating diminishing interest in participation over time (see Figure 4).

Questions requiring a simple answer, such as a rating or choice between limited options, frequently received higher numbers of responses than those requiring more thought. Similarly, questions which were not directly linked to the workshop activity, and instead required students to think more deeply about a software

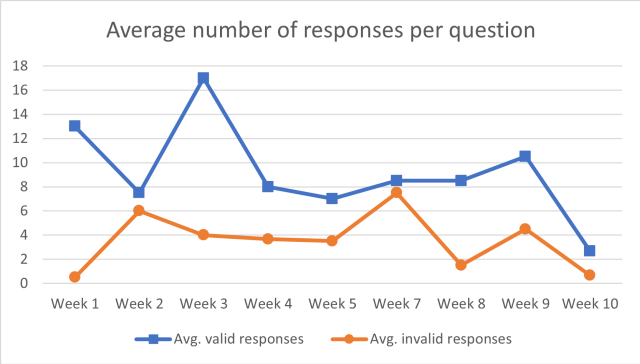

Figure 4: Average number of responses per question

Table 2: Average number of valid and invalid responses per feedback question

Week	Task type	Average number of valid responses per question	Average number of invalid responses per question
1	Paper-based	13	0.5
2	Practical	7.5	6
3	Practical	17	4
4	Paper-based	8	3.67
5	Practical	7	3.5
7	Practical	8.5	7.5
8	Paper-based	8.5	1.5
9	Practical	10.5	4.5
10	Paper-bsed	2.67	0.67

Table 3: Response totals by type

Week	Task type	Total responses	Total valid responses	Total invalid positive responses	Total invalid neutral responses	Total invalid negative responses
1	Paper-based	27	26	0	0	1
2	Practical	27	15	9	3	0
3	Practical	42	34	4	3	1
4	Paper-based	35	24	9	1	1
5	Practical	21	14	3	4	0
7	Practical	32	17	10	2	3
8	Paper-based	20	17	0	3	0
9	Practical	30	21	0	8	0
10	Paper-based	10	8	0	2	0

engineering concept, gained the fewest responses. The response rate dropped significantly if there were more than two questions requiring a written answer on any given week.

With regard to the feedback items requiring an 'upvote' or 'downvote' on achievement of a learning outcome, although only one downvote was received across all learning outcomes listed, very little meaningful feedback was gathered. There were low levels of interaction with these items, with no comments being left to gain any further insight into the thought behind the response being given. This observation highlights the importance of careful consideration when choosing the appropriate format for directly measuring attainment of learning outcomes.

When asked to give their opinion on the use of Padlet as a reflective feedback mechanism, all responses from the students indicated that they found the tool to be useful and the feedback process was beneficial to them.

Table 4: Mid-module feedback questions and ratings

Question	Mean
1. I find the lectures useful for learning the material	3.9
2. I find the labs/seminars useful for learning the material	4.2
3. I think the module is well organised and runs smoothly	4.1
4. Overall, I am satisfied with the quality of this module	4.1

Table 5: End-of-module feedback questions and ratings

Question	Mean
1. I found the module to be intellectually stimulating	4.4
2. I felt this module challenged me to achieve my best work	4.2
3. I think this module was well organised and ran smoothly	4.5
4. The criteria used during marking was clear in advance	3.9
5. I felt that the marking and assessment was fair	3.8
6. Feedback on my work was timely	4.0
7. I was able to contact staff when I needed to	4.5
8. Any changes in the module or teaching were communicated effectively	4.5
9. The TAs were effective in supporting labs classes	4.5

4.3 School-Organised Module Feedback

Mid- and end-of-module feedback was collected from all students on Computing Sciences courses using a formal feedback process, involving standardised questions (scale 1-5, with 1 being 'strongly disagree' and 5 being 'strongly agree'), with the opportunity for additional free-text responses. 9 responses were collected during mid-module feedback (see Table 4), and 15 responses were collected during end-of-module feedback (see Table 5).

The module performed well at both stages of the school-led feedback, and received a positive mean rating across all questions presented to students. The workshop sessions, where the majority of the reflective process took place, received notably high scores (a mean rating of 4.2 out of 5) on mid-module feedback, with students commenting that they *"help to reinforce the concepts taught in the lectures by providing practical exercises"*, *" are engaging and help me to understand how the theory can be applied to practical tasks"*, and *"were one of the best in my entire bachelors degree"*.

In end-of-module feedback, the module also scored highly on aspects including 'the module was organised and ran smoothly', 'I was able to contact staff when I needed to', and 'changes in the module were communicated effectively', with each of these aspects receiving a mean rating of 4.5 out of 5.

5 LESSONS LEARNED AND POINTS FOR IMPROVEMENTS

Through our analysis of feedback data and drawing upon our experiences, we were able to identify four broad categories that our key aspects of learning could be classified into. These key aspects of learning could then be revisited and explored to suggest further improvements that could be made to the approach, which would be helpful when applying it to other modules. The four categories are explained in the following subsections with accompanying points for improvement.

5.1 Valid vs. invalid answers – question design

- Lessons learned: As we identified in Section 4.2, some answers from the students were invalid, meaning the answers did not have appropriate contents to be useful for making improvements. Occasionally, it was the case that some students did not take the reflection task seriously. It was also noted that some questions were difficult to understand in a short period of time and this led the to students providing invalid answers. This is clear evidence that the quality of the questionnaire really matters for the success of the reflective approach, though the process is well established and followed.
- To improve: Clarifying a motivation for each question: as a simple analogy, the questions should be designed as if an exam question is set. Like the way we set exam questions to test learning outcomes, each question for reflection should also have a clear goal. As a general guideline, we may consider the following:
  - Simulate possible answers: as one member of the teaching team sets questions, another member tries to answer them independently. We noticed that sometimes the member's answers were irrelevant or the question wording made it difficult to understand in order to be able to answer correctly.
  - Consider the usefulness of possible answers: In addition to considering the expected answers, their usefulness for improving the teaching approach must also be taken into account. For example, an answer to the question of 'What improvements would you make to the lecture?' might be 'want to have more examples'. This is a perfectly valid and expected answer, but we may have already included as many examples in the slides of the lecture and workshop as we are able to cover, given the limited time available. Therefore, we could be more specific about the improvements we want to focus on - for example, 'Amongst the examples covered today, which one was the most effective for helping you understand the core idea of software architecture, and why?' would create more useful information to reflect and improve our teaching approach. We would be able to discover that the students were in favour of 'video-based case studies relevant to the topic' rather than static slide-only examples.

5.2 Learning vs. fun – adoption of gamification

- Lessons learned: During the workshop sessions, we tried to include more interesting and interactive tasks rather than static paper-based discussions. For example, the ball game activity was the most dynamic session, as the students were competing with each other in teams and the winning team would win a prize. We believe we were very successful in gamifying the teaching content (e.g., for the ball game, the main learning outcome was 'achieving product quality improvement through rapid iterations of an agile approach'). However, as discussed in section 4, the feedback data for game-based workshops had the highest levels of invalid answers. Conversely, the static and individual activity of test case development in week 3 gained the most valid feedback. This shows the importance of balance between the learning and the 'fun' elements. Through some informal conversation with the students, many informed us that the ball game was the most memorable, but did not recall much about the learning outcomes from this session. Too much focus on 'fun' can detract from the real message of the activity: the learning outcomes.
- To improve: Based on our experience, we would like to improve our approach as follows:
  - Too much fun might hinder learning attainments, but it is equally inadvisable to only use activities which may potentially be perceived by students as being 'boring'. As such, we would continue our current teaching approach with games, but we would also incorporate a 'cool down' period and explicit reflection time for the students to understand the meaning of the games.
  - Injecting the learning outcomes during the game activities. Instead of reflecting at the end of the game session, we may link the intermediate game result (e.g., the productivity improvements between the initial and second games in the ball game case) to the learning outcomes so that the students would understand the meaning gradually.

5.3 Short term vs. long term feedback cycle – adoption of agile approach

- Lessons learned: We initially believed that a short weekly feedback cycle would be most beneficial, and at the beginning of the module, this appeared to be true. From around week 5, we noticed a decline in the number of responses given by the students. We observed that the frequent feedback collection might have caused 'feedback burnout' and the students were significantly less interested in providing feedback than they were in the beginning of the module.
- To improve: After we noticed such fatigue in answering questions, we revised the type of questions being presented, and were able to improve the overall response rate (with the exception of the last week of the module, where obvious reasons such as low attendance due to preparing the coursework submission had an effect). We suggest improving the feedback rate with the following guidelines:
  - Weekly reflective questions: should focus on assessment of knowledge and measuring the level of understanding of the teaching material.
  - Questions presented at 3–4-week intervals: should collect information on how the interaction with the teaching team was perceived and how useful the teaching team's reaction was based on the weekly student feedback.
  - Questions presented at longer term intervals: should revisit the benefits and drawbacks of teaching methods and give an opportunity to compare with other modules.

5.4 Reflective learning vs. reflective teaching – distinction between types of reflection

- Lessons learned: The two concepts of learning and teaching are tightly coupled, and the improvement of learning usually implies the improvement of teaching and vice versa. However, in some cases, we noticed that good feedback on one does not necessary mean that the other was also good. One clear example was the case of gamification - the students acknowledged and appreciated the new idea of active teaching, but the resulting level of learning appeared to be lower when compared to other weeks.
- To improve: We suggest that the teaching team could enhance the teaching method and content by treating these two concepts separately during the design process. We would keep the following guidelines in mind for future module design.
  - For reflective learning, we should focus more on the teaching content and how to confirm the content is actually delivered to the students when we use a certain teaching method. When we use gamification, for example, we should more closely match the learning outcomes to the game activities.
  - Reflective teaching should consider a holistic approach by using available teaching techniques. For instance, a gamified workshop activity could also include peer assessment to improve the student learning. Note that the reflection on teaching should happen at the teaching team side based on student feedback.

These lessons learned and points for improvements will be incorporated into part of module revision plan for next academic year. We conclude our paper in the next section.

6 CONCLUSION

A successful software engineer should be equipped with 1) software implementation techniques and 2) problem analysis, optimal solution selection with rationale, and software project management skills. In our curriculum, the Software Systems Engineering module described in this paper has an important role to play, covering the latter part of the content for students. As such, we have placed much emphasis on improving the quality of teaching and learning on this module. This paper has described the most recent improvements made, and how they were perceived by the cohort of students, based on student feedback provided. For the improvements made, our main objective was to embed reflective approaches to enhance student learning as well as teaching, with an agile rapid feedback cycle.

Our post-analysis shows that the module was well-received by the students and we were able to collect sufficient evidence from the student feedback to support this. Most of our approaches resulted in positive feedback and contributed to improvements in teaching quality, however, we also identified some key aspects in our method where there were still areas for improvement, as discussed in Section 5. Some examples include the making links more explicit between learning outcomes and workshop activities, and careful design of feedback questions, along with feedback collection frequency. We feel that integrating the additional updates into the module for future years will be invaluable, and we intend to continue collecting and analysing feedback data in order to make further improvements.

6.1 Future Work

If a similar approach was to be implemented on a larger module, manually categorising student feedback may become burdensome for the teaching team. To alleviate this challenge, there may be potential to implement natural language processing to automate part of this process. Techniques such as sentiment analysis or topic modelling could be employed in this context, in order to more easily build a picture of general consensus amongst the cohort.

7 DATA AVAILABILITY

We are unable to make the data collected as part of this study available as our approved research ethics application stipulates that access to the feedback data is restricted exclusively to the principal investigator and co-applicant.

ACKNOWLEDGMENTS

We extend our gratitude to our students for their active participation in the lectures and workshops, as well as providing us with invaluable feedback.

REFERENCES

[1] David Boud, Rosemary Keogh, and David Walker. 2013. *Reflection: Turning experience into learning*. Routledge.
[2] Evelyn M Boyd and Ann W Fales. 1983. Reflective learning: Key to learning from experience. *Journal of humanistic psychology* 23, 2 (1983), 99–117.
[3] Anne Brockbank and Ian McGill. 2007. *Facilitating reflective learning in higher education*. McGraw-Hill Education (UK).
[4] Stephen D Brookfield. 2017. *Becoming a critically reflective teacher*. John Wiley & Sons.
[5] Lawrence E Carlson and Jacquelyn F Sullivan. 1999. Hands-on engineering: learning by doing in the integrated teaching and learning program. *International Journal of Engineering Education* 15, 1 (1999), 20–31.
[6] Jordi Colomer, Laura Serra, Dolors Cañabate, and Teresa Serra. 2018. Evaluating knowledge and assessment-centered reflective-based learning approaches. *Sustainability* 10, 9 (2018), 3122.
[7] Marija Cubric. 2013. An agile method for teaching agile in business schools. *The International Journal of Management Education* 11, 3 (2013), 119–131.
[8] Manoj Joseph D'Souza and Paul Rodrigues. 2015. Extreme pedagogy: An agile teaching-learning methodology for engineering education. *Indian Journal of Science and Technology* 8, 9 (2015), 828.
[9] Tore Dybå, Neil Maiden, and Robert Glass. 2014. The reflective software engineer: reflective practice. *IEEE software* 31, 4 (2014), 32–36.
[10] David Ellis. 2015. Using Padlet to increase student engagement in lectures. In *14th european conference on e-learning: ECEl2015*. 195–198.
[11] Marina Harvey, Debra Coulson, and Anne McMaugh. 2016. Towards a theory of the ecology of reflection: Reflective practice for experiential learning in higher education. *Journal of University Teaching & Learning Practice* 13, 2 (2016), 2.
[12] Orit Hazzan. 2002. The reflective practitioner perspective in software engineering education. *Journal of Systems and Software* 63, 3 (2002), 161–171.
[13] Masami Kimura. 2018. ICT, a motivating tool: A case study with Padlet. *Motivation, Identity and Autonomy in Foreign Language Education* (2018), 122–128.
[14] Gerald Kotonya and Jaejoon Lee. 2014. Teaching reuse-driven software engineering through innovative role playing. In *Companion Proceedings of the 36th International Conference on Software Engineering*. 276–282.
[15] Timothy C Krehbiel, Peter A Salzarulo, Michelle L Cosmah, John Forren, Gerald Gannod, Douglas Havelka, Andrea R Hulshult, and Jeffrey Merhout. 2017. Agile Manifesto for Teaching and Learning. *Journal of Effective Teaching* 17, 2 (2017), 90–111.
[16] Jaejoon Lee, Gerald Kotonya, Jon Whittle, and Christopher Bull. 2015. Software design studio: a practical example. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 389–397.
[17] Nahla Helmy Nadeem. 2021. Students' perceptions about the impact of using Padlet on class engagement: An exploratory case study. In *Research anthology on developing effective online learning courses*. IGI Global, 1919–1939.

[18] Donald A Schön. 1983. *The reflective practicioner: How professionals think in action.* Basic Books New York.

[19] Donald A Schön. 1987. *Educating the reflective practitioner: Toward a new design for teaching and learning in the professions.* Jossey-Bass.

[20] Ian Sommerville. 2016. Software Engineering. *Pearson Education Limited* (2016).