Experimenting with Realism in Software Engineering Team Projects: An Experience Report

Robbie Simpson and Tim Storer
School of Computing Science
University of Glasgow
18 Lilybank Gardens, Glasgow, G12 8QQ, United Kingdom
Email: r.simpson.3@research.gla.ac.uk, timothy.storer@glasgow.ac.uk

***Abstract*—Over Several years, we observed that our students were sceptical of Software Engineering practices, because we did not convey the experience and demands of production quality software development. Assessment focused on features delivered, rather than imposing responsibility for longer term 'technical debt'. Academics acting as 'uncertain' customers were rejected as malevolent and implausible. Student teams composed of novices lacked the benefits of leadership provided by more experienced engineers. To address these shortcomings, real customers were introduced, exposing students to real requirements uncertainty. Flipped classroom teaching was adopted, giving teams one day each week to work on their project in a redesigned laboratory. Software process and quality were emphasised in the course assessment, imposing technical debt. Finally, we introduced a leadership course for senior students, who acted as mentors to the project team students. This paper reports on the experience of these changes, from the perspective of different stakeholders.**

***Keywords*-Real world team projects; Flipped classroom; Agile methods; Mentoring**

I. Introduction

Many Computing Science programmes incorporate a Software Engineering Team Project course (TP), as this is a requirement of degree accreditation schemes, such as the British Computer Society [1]. At Glasgow, the course is undertaken in the third year of the four year BSc degree programme and runs through both semesters between October and March of the academic year. Students on the course are organised into teams of six students. The TP course itself does not incorporate taught material, but is delivered alongside a course covering Professional Software Development (PSD) principles, tools and methods. Over a number of years we had observed several aspects of the Software Engineering programme that were unsatisfactory. These observations arose from a number of sources, including discussions amongst staff and interview assessments of students during summer placements. Specifically, we have observed that:

- Students were unconvinced by the relevance of the material delivered in lectures, particularly with regards to quality assurance and change management. This scepticism arose because students knew that the projects specified for the course were largely 'artificial' for which they would have no responsibility beyond the end of the academic year. Students were therefore not convinced of the need for a more rigorous software process, because they were not exposed to a situation that demanded it. Further, the document-oriented approach to software process (RUP) created substantial overhead for the students, relative to the scale of the project tackled. This is not a critique of the RUP itself, but rather that it was not well suited to the short life-cycle, small team structure of the TP. This problem was further exacerbated by students returning from industry, where they were increasingly likely to experience the practice of agile methods.
- The typical structure of Computing Science assessments means that students are unfamiliar with situations in which requirements are uncertain and subject to change. Good practice dictates that students are provided with very clear specifications for coursework submissions, linked to intended learning outcomes (ILO). For example, an educator who wishes to check that students understand the mechanics of a linked list can specify coursework in which the students implement the data structure. Submissions are then assessed against the precise specification. In a TP, students are deliberately *not* presented with a precise specification of what to implement, since understanding the complexities of and methods for requirements elicitation, capture and negotiation is one of the key learning outcomes. In the TP at Glasgow, the course coordinators acted as customers and simulated requirements uncertainty with deliberate vagueness and contradiction. However, the students found this behaviour unconvincing and rejected the ILO. Students knew that the proposed project is not 'real' and are not convinced by the synthetic uncertainty of their 'customers'. We have, for example, found that students sometimes ascribe the intended vagueness in the specifications for projects to poor preparation by the course coordinators. In one instance of infuriated exasperation a student demanded that we 'just tell us what you want'.
- Assessment focused on product features delivered, rather

than product quality, or software process. Assessing a team's software process may be difficult, without ongoing monitoring of team activity, which may be time consuming. Instead, the assessment criteria assumes that teams which follow a rigorous software process will deliver better quality software in the final product. However, although assessment criteria refers to product quality as 'well-designed, functional, ..., maintainable,...', there is no further specific guidance on how to assess this. Consequently, assessment was based on the features that are readily identifiable in the delivered product. Students reasonably respond to this focus by delivering workable prototypes, however unmaintainable. This is unfortunate, because developing prototype quality software is not evidence of achievement of the course's ILOs. Student experiences in other courses, where the learning outcomes focus on demonstration of understanding of a concept within Computing Science, rather than the delivery of longer term maintainable code reinforces this bias. Consequently, we observed that students leave the programme lacking a sense of responsibility for the 'technical debt' in their code.

- There was a lack of time officially allocated to the team project course in the students' timetable. The TP and PSD courses between them accounted for one third of the credits available in the academic year, but only four hours per week were specifically allocated to them. Student teams were expected to arrange weekly meetings with an academic supervisor to review progress and agree objectives, as well as organise their own internal meetings and develop the project. However, no time was specifically allocated for this. In practice, many students tended to focus on smaller items of coursework with short term deadlines, neglecting the project itself.
- There is no opportunity to learn from mistakes, or benefit from the experience of others. We have found that there is a commonly held belief amongst colleagues that students learn the most from their failures; and that the existing design of the TP reflected this assumption. Students were given a theoretical introduction to a wide range of Software Engineering practices and tools, but little guidance on the practical selection and application of these tools, or incentive and time to practice them. Student teams were composed randomly from a population with little collective prior experience of Software Engineering (within a team or otherwise) and consequently lacked leadership or direction during the early phases of the work. We felt that this structure exacerbated the 'Storming' phase of team formation [2]. Students were in effect being set up to fail in the expectation that they would learn from this experience. However, we found that this meant many students became extremely anxious for their grades and resorted to 'thrashing', rather than taking the time to reflect on their failures and adapt their behaviours accordingly.

We anticipate that the above critique of the TP is recognisable in many Computing Science programmes. In this paper, we report on our experience of implementing a package of enhancements to the Software Engineering programme over several years to address these deficiencies. At their core, these changes are guided by the desire to create a more realistic environment for the practice of Software Engineering methods. Specifically, we sought to create project teams that work on real world problems with a real world customer; that work in a realistic Software Engineering environment, where they can benefit from the space and time to develop and practice modern agile methods and tools; and where novice student teams can benefit directly from the experiences of more experienced mentors.

A number of the changes implemented may be recognisable in other institutions and indeed, several are inspired by practices reported elsewhere, or from which we are aware by anecdote. However, all the changes made required customisation to fit within the constraints operating at Glasgow and we believe that when taken collectively, the package of measures represent significant innovation. Further, as Bull and Whittle [3] note, there is relatively little evidence of evaluation or even experience reports of innovations in Software Studios or Project Based Learning; and although we have found relevant literature discussing some aspects of the changes we have made, we have found little evidence of others. A further contribution of this paper, therefore, is to place the approach we adopted into the peer reviewed literature in a structured form to encourage further discussion.

This paper is structured as follows. Section II describes the changes made to the Software Engineering curriculum in the University since 2014. Section III presents the results of our interviews with different participants in the programme, including team project students, student mentors, industrial mentors and team project customers. Section IV discusses the implications of the changes made to the programme and reflects on links to related work. Finally, Section V assesses the wider implications and identifies the next steps for our programme of research and programme changes.

II. Changes Made to the Programme and Rationale

This section summarises the changes we have made to the delivery of the Software Engineering topics within the Computing Science Curriculum at the University of Glasgow. The changes comprise alterations to two courses: the Third Year Team Projects (TP) and an associated Professional Software Development (PSD) course; and the introduction of a new course in the Final Year, Advanced Software Engineering Practices (ASEP). The changes were introduced gradually since 2014 and, as discussed later in Section V, are part of a continuing reform of the curriculum.

A. Agile Software Project Structure

Students had previously been introduced to a wide range of software project models, including Waterfall, Rational and agile methods. A decision was made to focus primarily on agile software project structure and practices and encourage this through the assessment model for the course. Teaching materials were adjusted appropriately, emphasising topics selected from several agile methods, including agile software project planning, change management, user stories, test and behaviour driven development, software inspections, continuous integration and agile process improvement.

Practical alterations included structuring the projects into six one month iterations, with progress demonstrations and further requirements gathering from customers at the beginning of each month. Student teams are required to employ version control, issue tracking and continuous integration systems. Further, each team is required to undertake a retrospective at the end of each iteration to reflect on their software process and identify opportunities for improvement.

Rationale: Agile software development methods such as Extreme Programming (XP) and Scrum are recognised as being suitable for small teams of between 4 and 12 developers [4]. Further, agile methods, while requiring considerable discipline to apply well, are easier to introduce gradually into a team's practice [5] and are accompanied by less overhead, such as documentation maintenance. We anticipated that students would find these methods easier to adopt and practice, as well as recognise the benefits more quickly.

B. Flipped Classroom Delivery

We converted all the lectures in the PSD course into online materials, comprising short videos, slides, notes and links to supplementary reading. Students are required to cover this material in advance of taught contact time, which is spent in the laboratory working on project activities and in discussions with academic staff and demonstrators. Students are given a weekly online quiz with a small amount of credit each week to encourage engagement with the material. By consolidating all the time allocated into lectures, it was possible to allocate the whole of one working day for team project activities. A preexisting arrangement that allowed students 24 hour access to the laboratories was also retained, facilitating flexible working outwith the timetabled laboratory hours. We also re-designed the laboratory space for the course to be more suitable for co-located team work. Figure 1 shows the design of the new space, with students grouped into clusters with space for 6 students in each cluster. The space is equipped with overhead projectors on each wall, as well as extensive whiteboard wall space to facilitate design discussions and other project meetings. Each team table is also equipped with its own retractable whiteboard.

Rationale: Lectures are a time consuming and inefficient means of delivering information, and encourage rote memorisation of information over understanding of concepts [6].

Figure 1: The re-designed teaching space for the Team Project

In our own experience of delivering the course, we have found that a typical one hour lecture could be condensed into approximately 15 minutes of video footage, since time is not spent on setup problems, answering questions from the audience and emphasising key points. Eliminating lectures increased the amount of time that could be allocated for in-laboratory activities and allows students dedicated time to make progress on their software project. This reduces the risk that students will not themselves allocate sufficient time each week to the project, by making the time that should be used for it explicit in the timetable. Grouping this time together in a single day allows student teams the space to concentrate on developing the project and making progress with less distraction from other lectures or coursework.

C. Real world project customer for every team

The School of Computing Science has extensive links with external organisations. We advertised the opportunity for organisations to participate in the TP as customers, prioritising responses in order from charities, startup and SME businesses, the public sector, other university business units and lastly larger private organisations. Each customer was asked to create an initial project specification, with the advice that projects should address a real concern or challenge for their organisation. We advised that a good project specification should be relatively self-contained and not directly affect day-to-day operations at the organisation (i.e. project failure should not have critical consequences for the organisation). In the last iteration of the course, we selected 13 customers, with each customer working with two student teams.

Each customer was required to meet with the student team for six whole days during the course of the project. These *Customer Days* combined formal assessments of the teams' progress demonstrations and further requirements gathering with opportunities to informally discuss requirements with

customers. Beyond this, customers were free to structure their interactions with the project teams as they wished and we encouraged teams and customers to decide their own communication arrangements. Customers could also contact student teams more frequently or make arrangements for other activities such as site visits, if they wished.

Rationale: We anticipated that interaction with real world customers would enhance student team motivation to participate in the project. We have found in assessments of placements, that students often note the additional incentive of working on real world projects with real customers as a factor in ensuring project success. While we are unable to offer compensation to student teams for their project work, the real world customer ensures that student teams work on software where there is real demand. Further, by working with real world customers, students would be forced to recognise the necessity of engaging in negotiations over requirements, rather than depending on the course coordinators to eventually reveal them. This also means that students would need to monitor project costs and make adjustments to project schedules to ensure that the intended goals for the project remained realistic.

D. Emphasise technical debt in assessment

Several changes were made to the assessment model. First, every team received formative assessment of their software process at the end of each iteration, focusing on change management, quality assurance, project planning and process improvement. The assessment was informed by evidence gathered from their software project tools. Assessments were directed by practical questions concerning the maturity of the team's software process. For example, commit messages in the team's version control tool and tickets in the issue tracking tool were inspected for usefulness against a checklist. Only the final assessment of software process was included in the grade in the course, providing the student teams with the opportunity to improve gradually over the course of the project. Second, the team's interactions with their project customer were assessed, with marks available for the organisation and delivery of a progress demonstration and the negotiation of further requirements and priorities for the subsequent iteration. Each team was allocated 30 minutes to hold a formal meeting with their customer during a Customer Day. Marks were assigned for the professionalism of the demonstration, requiring the teams to describe progress against agreed objectives. Teams could be equally credited if they described a successful iteration or if they explained delays encountered and their causes. Credit was also available for negotiating and documenting new requirements. Third, the mark scheme for the final dissertation was redesigned as an experience report, requiring the team to reflect on the impact of the course on their thinking.

Rationale: Previous iterations of the course did not emphasise long term *responsibility* for the delivered software. By transferring the weight of the assessment to the demonstration of Software Engineering practices and reflection on this activity, we believed that we could compensate for the product centric thinking adopted by our students in their approach to coursework. As a consequence of these changes, only 20% of the total credit for the course was concerned with the final delivered software product and this included an assessment of software quality. Assessment of the teams' software process (25%), performance when demonstrating their progress and gathering requirements (30%), and for the final reflective dissertation (25%) accounted for the rest, incentivising students to practice more disciplined software development throughout the project.

E. Final Year Mentors for Student Teams

One challenge we have encountered in flipped classroom teaching with large classes is being able to deploy sufficient numbers of trained tutors to support students during the laboratory sessions. This is particularly challenging when laboratories extend across a full day. To address this, we began employing final year (FY) undergraduate students as tutors. One risk of this approach is that the FY students lacked the mentoring skills and confidence to engage with the third year teams and would be reluctant to intervene with teams or make recommendations, even when asked directly for advice.

To address this, we developed a FY mentoring and leadership course, Advanced Software Engineering Practices (ASEP). The course was developed in collaboration with an industry partner, the JP Morgan Glasgow Technology Centre. The course has two broad themes: an introduction to mentoring and leadership theory and skills, including, team psychology, process control theory and its application to agile project management; and a selection of current and advanced topics in Software Engineering, including Design Studios; Test Driven Development and Continuous Integration. Although some of these topics are delivered in the Third Year PSD course, there was a desire to spend more time revisiting them and extending the depth to which the topics are covered. The course is delivered in a series of focused workshops in which mentors from JP Morgan participated.

The coursework for the ASEP course is also designed to encourage the FY students to have the confidence to intervene with project teams. Mentors were required to monitor the student teams during their demonstrating hours in the laboratory, in order to identify emerging problems in the team software process. To facilitate this, the mentors were also tasked with providing the formative feedback discussed in Section II-D. The mentors then designed and implemented a software process improvement activity (PIA) to address the problem, in collaboration with the project team. Assessment focused on the students' planning, preparation, monitoring

and critical assessment of the process improvement, rather than whether it fundamentally succeeded or failed.

Rationale: The FY students had completed the TP course the previous year and undertaken summer or full year internships before starting their final year. Our intention was that the FY students would 'evangelise' the third years on the relevance of the Software Engineering practices that we taught, as well as deliver other topics based on their own experience, and be a more convincing emissary than academic staff. Providing the FY mentors with the skills and confidence needed to act as a mentors would enhance this experience and prepare the students for leadership roles in their future careers.

III. Interviews

Semi-structured interviews were conducted (Ethics Approval Number 300160104) with participants in all the different roles in the courses. This method was selected because we wanted to assess the qualitative impact of the changes implemented in the courses from the perspective of the participants. We decided against a quantitative pre- and post- assessment because the restructuring had occurred gradually over a number of years, and because the assessment model itself had been changed significantly. Open invitations were distributed via email requesting participation in the study in May 2017, approximately one month after the end of teaching in the 2016 academic year. Four categories of participants were interviewed: three Third Year TP students; five final year student mentors; five project customers and all the industry mentors from JP Morgan. We also considered interviewing the lecturers who coordinated the course. However, as two are co-authors of this paper and the others are close colleagues we felt this would be artificial. An initial set of questions were developed for each category of participant [7]. The questions focused on the perceived benefits and limitations of the course from the perspectives of the different participants, but also on how they perceived other participants. Students and project customers were all interviewed individually. The industry mentors from JP Morgan were interviewed collectively, as this was the most convenient arrangement. Summaries of the discussions with each category of interviewee are presented below.

A. Third Year Students

Three Third Year students were interviewed (L3A, L3B and L3C). L3A liked the structure of the course overall, describing the project as 'very realistic', and the style of development matched their expectations of commercial software. L3A remarked on the fast pace of the course, saying 'there was always something coming up' and referring to it as a 'permanent hackathon'. They stated that they found the combined workload of the PSD and TP courses burdensome and as a consequence prioritised the project. L3B observed that the part time nature of the project did reduce the realism. L3C stated that they learned about the practice of change management in larger teams and the importance of team communication and division/estimation of tasks.

Students reported spending a variable amount of time in the laboratory. L3A and L3C's teams met and worked collectively for most of the allocated laboratory day. L3A's team would also work sporadically on the project at other times during the week. L3C also reported using pair programming and standups in their team. L3B's team spent only 1-2 hours in the laboratory per week initially, but this increased in the second semester. L3B's team also worked more individually, often only meeting for 15 minutes a week as a team. They adopted a flexible team structure with interchangeable roles, and did work together on an ad-hoc basis. L3A and L3C reported that the laboratory space was busy, but that the layout and facilities were good for teamwork, when available. This meant that L3C's team often ended up working from home.

All three interviewed students said they developed a close working relationship with their customer. L3A's team had weekly customer meetings and used Slack at other times. L3B reported making visits to the customer's office and communicating via email. L3C's team used Google Hangouts for weekly meetings. L3A's project is now live with their customer. Both L3B and L3C reported that their customer was happy with the result of their project as a 'proof of concept', but it is not certain whether it will be used in production.

L3B and L3C stated that they enjoyed the flipped classroom model and made use of the videos to prepare for exams. However, L3B also liked conventional lectures as they provide an opportunity to ask questions, but would have also like more tailored guidance on how to apply the online material to the specifics of their project. All three liked the quizzes, but one (L3C) thought they were too easy. L3A noted that they would have preferred more information on team dynamics, such as people management or conflict resolution, rather purely on the technical side of project management. Also, some of the advanced topics covered in semester 2, such as aspect oriented engineering, were not obviously useful. L3A also noted that the explicit assessment model influenced their behaviour.

All three L3 students stated that their team did not make much use of the FY mentors, as they didn't feel the need for help. However, L3A did find the formative feedback provided by the mentors useful for identifying areas of improvement. L3C reported taking part in a PIA and felt that was a 'good experience'. The mentors lead them through a 'richer' retrospective adopting particular structures, which the team then used later.

B. Final Year Mentors

Five final year students were interviewed (FYA - FYE). All five FY students liked the course and particularly enjoyed the mentoring coursework aspects and the emphasis on

coursework in the assessment. FYB reported liking the lightweight nature of the assessment and found the diaries useful. FYD and FYE stated that it was interesting to see a software development project from a different point of view: as observers not participants.

All five students agreed that an internship was a useful experience before acting as a mentor. FYC believed that the internship was useful because they understood *why* good Software Engineering practices were needed and could better explain this to the students. FYC reported that students asked about his experience in industry and about internships. FYD and FYE learned a lot about team dynamics and agile methods during their internship, which was especially useful when designing PIA exercises.

FYA and FYB, FYD and FYE reported feeling of mixed use, whereas FYC felt that students generally accepted his advice. Sometimes it was difficult to tell if students needed help, so they felt like they might be intruding on their work. FYB noted that sometimes they couldn't help the teams with the particular technologies they were using. FYB, FYD and FYE reported feeling most useful during the middle of the course when the teams began asking questions concerning project management. FYD and FYE suggested that briefing sessions on PSD material would be useful each week to make them aware of key issues in the laboratory and likely questions. FYA stated that some students struggled to assimilate the advice they provided and sometimes wanted the mentors to solve the problem for them. FYB also reported difficulty in convincing L3 students of the merits of their advice, observing that the students often just wanted to start coding and 'hack it all together'. However, this improved gradually during the year. FYB, FYD and FYE suggested that the mentors should have more training and guidance upfront before they start working with the teams. FYA, FYD and FYE stated that the formative process assessment with a specific team was useful in helping to develop team-mentor relations. FYC suggested that being allocated to mentor a specific team throughout the course would have been preferable for this reason.

FYA conducted a PIA focused on change management. They felt that the student team was receptive to this, even though they had concerns that the L3 team wouldn't engage. They felt the coursework was 'well balanced' and 'open enough' to be interesting. FYB focused their PIA on mini-retrospectives, encouraging the team to perform a retrospective every week. This worked well at first, but the team eventually became bored by it. They learned that the retrospective should be done, but less frequently. In contrast, FYC did not implement a successful PIA (on using Kanban for project management), reporting that the student team were not motivated to engage. FYC noted that the team were all joint honours students and so were doing a proportionally smaller scale project compared to single honours teams. FYD and FYE found the process of writing a plan and getting feedback for a PIA helpful as it gave them the opportunity to improve. However, they would prefer more time before implementing the warm up exercise to identify problems more clearly.

C. Project Customers

Five customers were interviewed, CA, CB, CC from the 2016 iteration of the course, CD from the 2015 iteration and CE who participated in both iterations. Only CD and CE had prior experience of working with students on projects and all customers reported considerable uncertainty at the outset regarding the likely capabilities of the students, the expectations of themselves as customers and the likely outcomes of the collaboration. However, all the customers spoke positively about the experience.

CA now has a usable working application for managing lone workers, which they believe distinguishes them from other charities and described the experience as 'enlightening' and 'worthwhile'. CB did not adopt their projects into production, but have achieved a useful proof of concept. CB did, however, use the course to give their more junior staff some experience of project management. CB stated that they 'would have loved to' take some students on for internships, but were not able to do so because of timing. CC also used the projects to prototype ideas and were able to explore 'important research ideas' with one team that completely exceeded their expectations and could be quickly re-implemented. CC stated that they were keen to take part in the team projects again. Two of CD's teams delivered workable systems. One for data visualisation and the other various physical devices for monitoring engagement in exhibits. Both these projects are still in use, and have been updated with new data. CD reported that seeing other customer presentations was enjoyable and that the course had also initiated a collaboration with another of the customers. CE stated that they chose projects that were non-critical for the first iteration they participated in, in order to determine what the students were capable of. CE said they were 'pretty happy' with the experience, and had obtained 'two very useful products'.

All five customers spoke positively about the performance of the students overall, although they did recognise differences between the teams that were allocated to them. CA stated that they were impressed by the knowledge, ability and work ethic of the students and felt that the students were better than some professional tech companies, particularly in their ability to 'empathise' with the customers' requirements. Both CA and CD reported that one of their teams deviated from the requirements to focus on aspects of the project which interested them. CB stated that one of their teams was very good, while the other was less so. CB were surprised that students had no specific knowledge of Apple technologies and had also not been exposed to protocol concepts (this is taught during another third year course).

CA, CB and CC felt the students benefited from working with less technical clients, which required them to work on their communication, project management and requirements gathering skills. CB also noted that students initially lacked project management skills and learned about prioritisation and managing the customers expectations. CB suggested that there should be more focus in the taught course on communication skills and team dynamics, as this would help the team manage customer expectations. CC found that all of their students struggled with designing in the abstract. They felt they needed actual data to look at before system design could proceed. CE stated that due to the nature of his project, students learnt a lot about information governance and data protection, given the importance of data confidentiality.

All the customers reported that the commitment was found to be quite lightweight. CA and CC engaged in relatively little interaction with the students beyond the core requirements. CA had one extra meeting a small number of exchanges via email and reported that this was felt to be a very minimal commitment. CC also stated that interactions with the student teams was quite undemanding, having only two extra meetings at their offices and using slack and email at other times. CD felt that the monthly meetings were about right and stated that one of their teams stuck to the set meetings, whilst the other was far more engaged, making several trips to the customer's site. CB held meetings with teams every two weeks, stating 'we created our own structure'. CE tried using video-conferencing, but this never worked effectively due to technical problems.

Several customers would have liked more guidance on how to interact with the students and how to arrange projects. Some customers ran the same project with all their teams, while others had different projects. Both CC and CD reported that making the initial 'pitch' to the students intimidating. CD stated it was 'difficult but valuable'. This was not something they were used to and was not particularly enjoyable to do, but it forced them to think and express clearly the aims of their company. CC also asked for more guidance on intellectual property sharing. CD stated that one of their teams lacked confidence and was 'terrified" by meetings with them. Both CC and CD suggested that this might be improved if there had been more time at the start of the course for them to get to know the customer and build a rapport. CE stated that the formal customer meetings could feel a little rushed. They would prefer to have longer meetings (45 minutes to 1 hour) in a quieter and more private venue. CB and CE found the end of the project lacked organisation and subject to time constraints, meaning that handovers of projects were haphazard. CB suggested that it would be useful to have a final customer meeting after the presentations to wrap up or handover the project. CE suggested ending the development aspects of the projects slightly earlier to allow time for a handover. CD had acquired funding to take on one student over the summer to carry the project on, but this came through too late to take the student on.

D. Industry Mentors

The industry mentors had relatively little past experience in teaching before the course. A number of their expectations about the course were challenged. In particular, they were surprised by the decline in attendance in the lectures and workshops and discovered the students didn't know each other as well as they had expected. This made establishing a rapport with the students more difficult. The mentors also found that the students were familiar with the mechanics of agile methods and that this caused some overlap with material taught in the PSD course. However, they found they were not aware of the deeper reasons *why* agile methods are practised in industry. Overall, the mentors were positive about their personal benefits from taking part in the course. It was described as 'rewarding', and they felt that experience would benefit them in their future careers. They noted that students were prepared to challenge them about the content they were teaching, focusing them to think deeply about the techniques they were teaching.

There was some discussion of the 'cultural' issues regarding the course. For example, the industry mentors believed some students were deterred by the corporate background of the mentors, and did not like the idea of working in financial services. This meant that the students may not have been convinced by the experience of the mentors if they did not believe it was representative of the software industry. For example, the industry mentors reported that students were surprised by how little greenfield development work actually occurred in business and attributed this to the nature of the organisation's work, rather than the nature of the industry as a whole. Similarly, it was perceived that that the content of the course (focusing on techniques for large-scale teams) might not appeal to students hoping to work in smaller organisations. However, students eventually understood the motivation for these techniques, and hence engaged more effectively with the mentors.

The industry mentors reported on a number of practical problems with the structure of the ASEP course. The mentors had difficulty communicating with students outside of lectures due to corporate IT security policies. The mentors also recommended swapping the ordering of the week, so that the lecture happened first to prepare students, followed by the practical workshop. Like some of the FY students, the industry mentors were not sure that the skills and techniques taught in ASEP were linked to the students' roles as mentors . Further, the FY students did not have opportunities to apply all of the methods and practices shown to them during the ASEP course. One suggestion was to implement additional coursework to demonstrate these advanced concepts, such as a team project running for the length of the course.

IV. Discussion and Comparison with Related Work

The challenge of providing realistic experiences of Software Engineering has been addressed by a number of different authors. Dawson [8] explored the challenge of introducing realism through playing 'dirty tricks' on students. While we have independently experimented with these techniques, we have found that a limiting factor is plausibility. In our experience students simply find this behaviour unconvincing, believing it to be unrealistically malevolent, and are therefore resistant to the intended learning.

Many of the initial changes to the courses at Glasgow were inspired by the work on Software Engineering Studios by Bull et al. [9], which itself is based on older work on Design Studios in software development [10]. The size of the TP class at Glasgow (approximately 150 versus 20) and the need to use the allocated space for other laboratory work was a constraining factor in implementing a number of desirable aspects of the approach adopted at Lancaster. Desktop PCs were provided for each desk with large monitors, which hinders collaborative activities. The size of the class meant it was also not possible to allocate dedicated space to each team that they could reconfigure as desired. Students were encouraged to make use of other spaces around the University to conduct work, if desired, and several teams used other laboratories, or meeting rooms in the library and elsewhere, rather than spending all their time in the dedicated laboratory. Some student teams also grouped themselves into bays to facilitate collaboration between desks. Carter and Hundhausen [11] presented a review of several experiences of implementing studio based learning in Computing Science. Of the small sample studied, Carter and Hundhausen noted that like our own work, the principle motivation was to enhance student exposure to realistic industrial circumstances and practices.

Garlan et al. [12], Boehm et al. [13], Hayes [14],Buckley et al. [15] and Suri [16] all reported on their experiences running project courses with real world customers. Unlike in our own course, in most cases, students in the same cohort worked on the same project [12, 15, 16, 13]. However, the similarities with our own experiences are striking. All the authors found that students were motivated by the opportunity to work on projects that had real world impact and reported examples of projects providing substantial benefits to the customer. Boehm et al. [13] reported scalability as a significant challenge for the provision of continuous assessment. Our experience suggests that employing senior students as mentors can be useful in scaling formative assessment activities. Buckley et al. [15] also found that students left the course more convinced by the difficulties of requirements elicitation. Hayes [14] reported that their project led to the exploration of further development opportunities and grant applications.

Other experiments with Software Team Projects have been reported. Favela and Peña Mora [17] reported their experience in teaching global, distributed Software Engineering practices in a team project. Students from two different countries were organised into a project which required collaboration throughout the software life-cycle and were provided with coordination tools to assist with this process. Favela and Peña Mora found that team distribution exacerbated the challenges during team formation, with causes including language and cultural differences between the institutions. Coppit and Haddox-Schatz [18] took an alternative approach to increasing the realism of a Team Project, with all 30 students in a class working in a single project organisation. Coppit and Haddox-Schatz imposed some structure on the project by formally allocating roles such as project managers and technical leads, but otherwise allowed the students to self organise. As in our own experience, Coppit and Haddox-Schatz used more experienced students for leadership roles. Overall, the authors suggest the experiment was a success, with the scale of the project imposing little additional workload and enhancing the student's understanding of communication challenges across large distributed teams.

Several authors have emphasised the importance of reflection as an effective enhancement of learning during the practice of Software Engineering, based on Schon [19]'s ideas. Moore and Potts [20] observed that the goal of educators is not just to produce software professionals who know about techniques, but also 'have good judgement about when to apply them'. Many of the practices advocated by Bull and Whittle [21] were also implemented in our own approach, including the use of staff as mentors rather than teachers, continuous presentations and facilitation of group discussions and peer critique. We also went further, by explicitly introducing a formal team retrospective at the end of each iteration of the project. Many of our students found the conduct of the retrospective difficult, preferring to use it for project planning rather than identifying and remedying fundamental process problems. It was therefore encouraging that so many of the FY mentors recognised this and chose to guide the software teams through the conduct of a retrospective as their process improvement coursework.

Hazzan [22] discussed the application of reflective practice to software design and architecture reviews, relating this to the physical architecture studios. Several of the students in the ASEP course experimented with applying coaching techniques to design processes with student teams and these were often successful in solving the team's design problems. The case study dissertation was also designed to encourage students to reflect on their experience in designing and implementing software, although in hindsight, this reflection may come too late in the course process and could be enhanced through discussion of content with course coordinators and student mentors.

V. Conclusions

This paper has presented our experiences of implementing a substantial collection of changes in the Software Engineering programme at the University of Glasgow. The changes have resulted in a number of demonstrable successes. Team project students are motivated by the opportunity to engage in a project with real world customers and impact in a more realistic environment. Student mentors welcome the opportunity to pass on their experience to their more junior colleagues. Industry mentors enjoyed engaging with the final year students and passing on their own expertise. Customers reported that the projects delivered considerable value for relatively little cost. Finally, the course coordinators (ourselves and our colleagues) have enjoyed delivering the programme in its revised form and feel that our teaching is more effective because of it. Informally, the structure is also believed to reduce workload, since we are not committed to lecturing throughout the week and formative assessment load is shared with the student mentors.

The results of the interviews identified several areas where the programme requires improvement. The size of the cohort at Glasgow means that scalability of teaching methods remains a challenge. Bull et al. [9] identified the importance of inter-personal relationships and culture in creating an effective studio based learning environment, so to a certain extent, it is inevitable that the size of the class will inhibit this. However, there are a number of further steps that could be taken, for example, by assigning a set number of teams to each of the course coordinators and student mentors, as suggested by the interview participants. Explicit team building exercises may also be useful in building rapport between the different participants.

Fostering a sense of collective ownership of the laboratory space, even with the constraints of the cohort size, would also be desirable. For example, Bull et al. [9] noted that some advocates of studios discourage the presence of digital technology because it inhibits collaboration. In our own experience, the presence of desktop PCs in the laboratory inhibits conversations 'across the table' and discourages student interaction. Many of our students work predominantly on their own laptops, so removing desktop PCs could increase the room capacity. This could also encourage the use of the laboratory as a social space in which students are permitted to eat and drink [23].

Now that the course is established there is a need to implement mechanisms that develop longer term relationships with the external organisations who act as customers. Further research is needed to understand how the different customers structured their relationships with the student teams and how these approaches worked in practice. It would be desirable to be able to supply new customers with guidance and best practices for engaging with student teams without constraining the 'naturalness' of the relationship. Further, the handover process at the end of the project needs to be more formal and incorporated into the assessment model. As well as instilling the sense of responsibility for the project in students, this would also ensuring that customers properly receive the benefit of their engagement in the course. Alongside the meetings with students, time should also be allocated to raising awareness amongst the customers of mechanisms for continuing projects in the longer term, through student summer internships, follow on final year dissertation projects and collaborative funding applications.

We are also considering adopting the introduction of distributed Global Software Engineering experiences into the course, as described by Favela and Peña Mora [17]. The School of Computing Science has a cohort of students based at the Singapore Institute of Technology. These students follow an identical curriculum to the cohort based on Glasgow, but are taught by locally recruited staff. Creating larger, distributed Software Engineering teams of students across these two sites could have a number of potential benefits. Customers could present larger scale challenges for the teams to tackle. Students would experience the complexity of managing software projects across time zones and semi-autonomous organisational units. Working collaboratively on projects would also foster a greater sense of a single cohort between the Glasgow and Singapore students.

These new directions raise the question as to how far realism should go in Software Engineering education. If more realistic development projects are better for students, then why not dispense with the university environment entirely? There are a number of steps in this direction in the UK at least, such as the increasing emphasis and availability of paid summer and full year internships as part of degree programmes. The growing Graduate Level Apprenticeships scheme will also dramatically increase students' real world experience. It might be argued that universities should dispense with project based learning and instead develop programmes which allow students to spend more time in the software industry. However, our project customers welcomed the fact that students were familiar with the theory of Software Engineering methods, if not their practice. University environments also provide students with a 'safe' space in which they can experiment with different practices and experience failure without significant consequences. Further, university programmes provide the space and time for students to develop as reflective practitioners, and adopt mechanisms in coursework which encourage this reflection. The challenge then, for Software Engineering educators, is to develop environments in universities that are sufficiently real to be convincing to students, whilst also providing the space to learn and experiment with the principles and practices they will need in their careers.

References

[1] *Guidelines on course accreditation. Information for universities and colleges*, British Computer Society, June 2015.

[2] B. W. Tuckman, "Developmental sequences in small groups," *Psychological Bulletin*, vol. 63, no. 6, pp. 384–399, 1965.

[3] C. N. Bull and J. Whittle, "Observations of a software engineering studio: Reflecting with the studio framework," in *27th IEEE Conference on Software Engineering Education and Training, CSEE&T 2014, Klagenfurt, Austria, April 23-25, 2014*, A. Bollin, E. Hochmüller, R. T. Mittermeir, T. Cowling, and R. LeBlanc, Eds. IEEE, 2014, pp. 74–83.

[4] K. Beck and C. Andres, *Extreme Programming Explained*, 2nd ed., ser. XP Series. Addison Wesley/Pearson Education, February 2005.

[5] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*. Prentice Hall, 2001.

[6] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth, "Active learning increases student performance in science, engineering, and mathematics," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8410–8415, 2014.

[7] R. Simpson, "Survey instrument," Available on GitHub https://github.com/twsswt/Simpson_2017_Team_Based_Software_Engineering_Teaching/releases/tag/survey-instrument-v1.0, June 2017.

[8] R. Dawson, "Twenty dirty tricks to train software engineers," in *Proceedings of the 20th International Conference on Software Engineering, ICSE 2000*, C. Ghezzi, M. Jazayeri, and A. L. Wolf, Eds. Limerick, Ireland: ACM Press, 2000, pp. 209–218.

[9] C. N. Bull, J. Whittle, and L. Cruickshank, "Studios in software engineering education: Towards an evaluable model," in *Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA*. Piscataway, NJ, USA: IEEE Press, 2013, pp. 1063–1072.

[10] J. E. Tomayko, "Teaching software development in a studio environment," in *Proceedings of the Twenty-second SIGCSE Technical Symposium on Computer Science Education, San Antonio, Texas, USA*. New York, NY, USA: ACM, 1991, pp. 300–303.

[11] A. S. Carter and C. D. Hundhausen, "A review of studio-based learning in computer science," *Journal of Computer Science in Colleges*, vol. 27, no. 1, pp. 105–111, October 2011.

[12] D. Garlan, D. P. Gluch, and J. E. Tomayko, "Agents of change: Educating software engineering leaders," *IEEE Computer*, vol. 30, no. 11, pp. 59–65, November 1997.

[13] B. Boehm, A. Egyed, D. Port, A. Shah, J. Kwan, and R. Madachy, "A stakeholder win-win approach to software engineering education," *Annals of Software Engineering*, vol. 6, pp. 295–321, 1998.

[14] J. H. Hayes, "Energizing software engineering education through real-world projects as experimental studies," in *15th Conference on Software Engineering Education and Training (CSEET'02), 25-27 February 2002, Covington, Kentucky, USA*. IEEE Computer Society, 2002, pp. 192–206.

[15] M. Buckley, H. Kershner, K. Schindler, C. Alphonce, and J. Braswell, "Benefits of using socially-relevant projects in computer science and engineering education," in *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia, USA*. New York, NY, USA: ACM, 2004, pp. 482–486.

[16] D. Suri, "Providing "real-world" software engineering experience in an academic setting," in *37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, Oct 2007, pp. S4E–15–20.

[17] J. Favela and F. Peña Mora, "An experience in collaborative software engineering education," *IEEE Software*, vol. 18, no. 2, pp. 47–53, March/April 2001.

[18] D. Coppit and J. M. Haddox-Schatz, "Large team projects in software engineering courses," in *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2005, St. Louis, Missouri, USA, February 23-27, 2005*, W. Dann, T. L. Naps, P. T. Tymann, and D. Baldwin, Eds. ACM, 2005, pp. 137–141.

[19] D. Schon, *The Reflective Practitioner: How Professionals Think In Action*. Basic Books, September 1984.

[20] M. M. Moore and C. Potts, "Learning by doing: Goals & experience of two software engineering project courses," in *Software Engineering Education, 7th SEI CSEE Conference, San Antonio, Texas, USA, January 5-7, 1994, Proceedings*, ser. Lecture Notes in Computer Science, J. L. Díaz-Herrera, Ed., vol. 750. Springer, 1994, pp. 151–164.

[21] C. N. Bull and J. Whittle, "Supporting reflective practice in software engineering education through a studio-based approach," *IEEE Software*, vol. 31, no. 4, pp. 44–50, 2014.

[22] O. Hazzan, "The reflective practitioner perspective in software engineering education," *The Journal of Systems and Software*, vol. 63, no. 3, pp. 161–171, September 2002.

[23] D. R. Herrick, "Food and drink in computer labs: Why not?" in *Proceedings of the 40th Annual ACM SIGUCCS Conference on User Services, Memphis, Tennessee, USA*. New York, NY, USA: ACM, 2012, pp. 161–164.