Agile practices taught online at a distance

Leonor Barroca
The Open University, UK

leonor.barroca@open.ac.uk

Karen Kear
The Open University, UK

karen.kear@open.ac.uk

Abstract
Agile software development has become, in the last twenty years, a popular approach to software development. It puts people and working software at the forefront of the development process. The emphasis on people stresses development carried out by teams of developers and stakeholders working together for the regular delivery of solutions.

There is long experience of teaching software development at a distance at the Open University, UK; however, teaching agile software development at a distance creates new challenges due to its heavy reliance on face-to-face communication. Developing these skills with students who are at a distance, studying online, and mostly part-time adds significant difficulties.

This paper reports a case study of the use of an online communication tool to help students develop agile communication practices, and to help tutors assess these skills. The tool is based on a studio-based approach to learning, where students work independently and in groups, learning from each other in an apprenticeship model. Using this tool, students share their mental models of problems and solutions, as they would in an agile development team.

A survey was carried out with students and tutors on the perceived impact of the use of this tool in learning; the results are discussed, in particular the importance of supporting students in using the peer feedback they receive in a reflective way in order to improve their work. The lessons learned for improving the quality of the collaboration and the richness of the learning experience are discussed.

Keywords: agile development, online distance education, studio-based learning

1. Introduction

Agile software development is an umbrella term used to describe a variety of methods and practices that encourage simpler, more lightweight, faster and nimbler software development. These practices enable software developers to adapt to the inevitable changes in customer requirements. This continual realignment of software development goals to the needs and expectations of the customer should result in software that better serves its purpose.

As seen in the agile manifesto[1], agile development is an approach to software development that puts people and working software at the forefront of the development process. The emphasis on people translates, in agile, to development carried out by (usually) small teams of developers and stakeholders working together for the regular delivery of solutions.

Teaching agile software development is now common in the software engineering curriculum; research has identified some of the problems, and made suggestions for how to address them (Devedžić et al., 2011; Maher, 2009). However, research on how to teach agile development and related skills at a distance is still scarce.

In this paper, we present a case study of the use of a studio-based approach to distance teaching of some of the skills required for agile software development. Studio-based approaches to learning can be used to develop and promote the skills that are in demand for agile development: collaborative working, quick feedback, learning with and from peers, and problem solving.

The case study is in the context of a software engineering undergraduate module at the Open University, UK. An online collaboration tool was used to enable students to share visual models that they had created, and to give feedback on these models. In an evaluation of the approach, students and tutors completed surveys which explored the impact of this tool on their learning and teaching; we discuss the findings of the two surveys.

Section 2 discusses research on teaching agile software development, particularly in a non-colocated context; section 3 introduces the studio approach to learning; section 4 presents the case study; the findings are discussed in section 5; section 6 concludes the paper.

2. Teaching agile software engineering at a distance

The research on the teaching of agile skills in a distance learning context is sparse. Rodriguez et al. (2015) propose a virtual world environment that simulates the context of an agile team, but they only use it in face-to-face teaching. Brocke (2011) appropriates the term 'agile' from the software engineering domain to talk about 'agile communication skills'. She discusses an eLearning environment to support the development of these skills in a university setting. Although she applies this environment in a context where students need to communicate with other non-colocated students, her context is not that of software engineering education, but rather of cultural studies, languages and social sciences.

At the Open University, UK, there is experience of around 30 years of teaching software engineering at a distance, both at undergraduate and postgraduate level (Quinn et al., 2006; Wermelinger et al., 2015). Technology has been used to support the teaching, and also the development of skills associated with the subject. At undergraduate level, for example, students experience teamwork mediated by technology within a module which teaches project and service management (Oldfield & Morse, 2007). The diversity of tools to support student collaboration and communication allows for the development, at a distance, of skills that are typically taught only in a face-to-face context (Kear et al., 2014).

There is extensive literature on how agile teams that are not colocated work (Holmstrom et al., 2006; Hummel et al., 2012; Jalali & Wohlin, 2010; Paasivaara et al., 2008); this suggests that it should be possible also to teach agile development at a distance, and that the distance or online context should not therefore

[1] http://www.agilemanifesto.org/

be a barrier. However, many issues need to be taken into account, mainly in the way technology is used and the constraints experienced by students.

3. The studio approach to learning

The studio approach to learning has its origins in the teaching of architecture, but has also been applied in computing education. It views learning as happening in a collaborative manner through the design and development of artefacts that are constructed iteratively, reviewed and refined by learners (Hundhausen et al., 2008).

The two main activities of this approach are the construction of visual artefacts and the sharing of feedback on these artefacts by peers and educators. Studio-based learning is an apprenticeship model that moves the learning from a focus on knowledge of concepts and techniques to an appreciation that communication, design and problem-solving are of key importance. These are especially relevant skills in the teaching and learning of software engineering, and agile development in particular; they are also skills that are increasingly in demand by employers (Shadbolt 2016).

In distance education, online tools can support the creation and presentation of artefacts by learners, and their active engagement in collaborative tasks involving feedback and problem-solving. Literature discussing studio-based learning in online environments is limited, although it has started being addressed in the generic area of STEM[2] subjects (Cennamo et al., 2011; Thomas et al., 2016).

The Open University, UK, has developed an online studio environment called OpenStudio. It was initially developed for a photography module in 2007, for students to upload their photographs and get feedback on them from peers; it soon became very popular and started being adopted by other modules for different purposes (Jones & Lloyd, 2013). In OpenStudio students can upload artefacts, view the artefacts uploaded by other students, and comment on them. Tutors also have access to their students' artefacts and comments.

4. Case study: developing agile skills at a distance

The Open University (OU) is the UK's largest university, with more than 260,000 students, most of whom are enrolled part-time in undergraduate degrees. All our undergraduate students are at a distance, and they fit their studies around their professional and family commitments. They study printed or online learning materials provided by the OU, and they work towards fixed-date assessment points, both during a module (which typically lasts about eight months) and at the end.

Since 2011 we have been revising our curriculum for level 3 in Computing and Communications to take into account a major UK government review (Browne 2010) and to ensure that our qualifications are engaging, relevant, and will equip students for their current job roles and future careers. This work took place against a background of a changing landscape in computing: where desk top computing has been overtaken by ubiquitous computing, where web is the norm, and where outsourcing and globalization mean that technical skills need to be at a high level.

[2] STEM stands for Science, Technology, Engineering and Maths

The case study discussed here is of *TM354 Software Engineering*, a module which is part of a theme relating to the way large technological systems are designed, built, used and maintained within organisations. TM354 teaches the principles, patterns, techniques and practices associated with requirements engineering, analysis, software architecture and design, as well as the principles and techniques of implementing and testing a software system. Students get a sound understanding of the quality issues involved in software products and processes. Many of these concepts are reasonably stable and have been taught at a distance for many years. The practices of software engineering have, however, been evolving, in particular with the surge of developers adopting agile.

4.1 The design

TM354 was designed with the intention of developing skills of abstraction, and promoting critical reflection. As a level 3 module, it also prepares students for a final year project in software engineering, developing skills such as conducting searches and reviewing found material. To promote an understanding of the professional context, the module includes (regularly updated) current topics on software engineering practice. This challenges students to consider what they are learning in the context of what practitioners do.

As part of the review of curriculum for TM354 the following elements were considered in its design:

- teaching agile approaches to software development;
- simulating some agile practices, such as the daily stand-up meeting;
- strengthening skills in sharing, discussing, giving feedback and reflecting on feedback received.

The last two elements are discussed here, as they were achieved with the support of OpenStudio.

Many agile teams use a whiteboard where sketches are kept, progress is shown, and collaborative modelling is displayed. Yu and Petter (2014) carried out a study of agile practice using shared mental models theory. This theory explains how teams develop a knowledge base that allows them to take decisions. In agile development, evolving a shared understanding among members of a team is a common activity.

The daily stand-up meeting is an integral part of agile practice. Practitioners spend no more than 15 minutes reviewing what has been achieved and what needs to be done; the meetings also promote conversations in order to reach shared understandings of problems and solutions. Yu and Petter (2014) talk about two types of mental models: taskwork and teamwork; the former is related to what is to be achieved and the latter related to the team interactions, communication and roles. The assessment developed for TM354 covers both these aspects: the task and the team.

TM354 has three assessments which are marked by tutors. The OpenStudio activities account for between 15% and 20% of each tutor-marked assessment (TMA). Students are asked to develop their own artefact, which is either a model to understand a domain problem, or a model to work towards a software solution to a domain problem. They upload their model to OpenStudio for other students to see. Students are also required to comment on other students' models, reflect on the peer feedback received on their own models, and improve their models accordingly. With the support of OpenStudio, students are encouraged to be creative and to work with others, sharing their artefacts and reflecting on feedback from colleagues. We decided against requiring any kind of synchronous (real-time) activity, as our students are mostly in full time employment, and find it very difficult to depend on others and align their disparate timetables. However, the way the activities were designed promoted the continuation of conversations and the reaching and sharing of an understanding.

In an agile approach, modelling should only be carried out as long as it is useful. Modelling can be used as part of a documentation of the process followed, but in agile development, models are not perfect artefacts. This is sometimes a difficult concept for students to accept, as they tend to focus on the correctness of details of the models rather than on the understanding that the model conveys. Students often apply modelling techniques mechanically, and find it difficult to stop until the model is 'perfect'. The assessment activities were intended to convey the message that conversation around the artefact, and reflecting on the shared meaning, is more important than the perfect drawing of the model. OpenStudio was used to promote these conversations and to strengthen the giving and receiving of feedback.

Figure 1 is a snapshot of the work uploaded by students for the first of their assessment (TMA01).

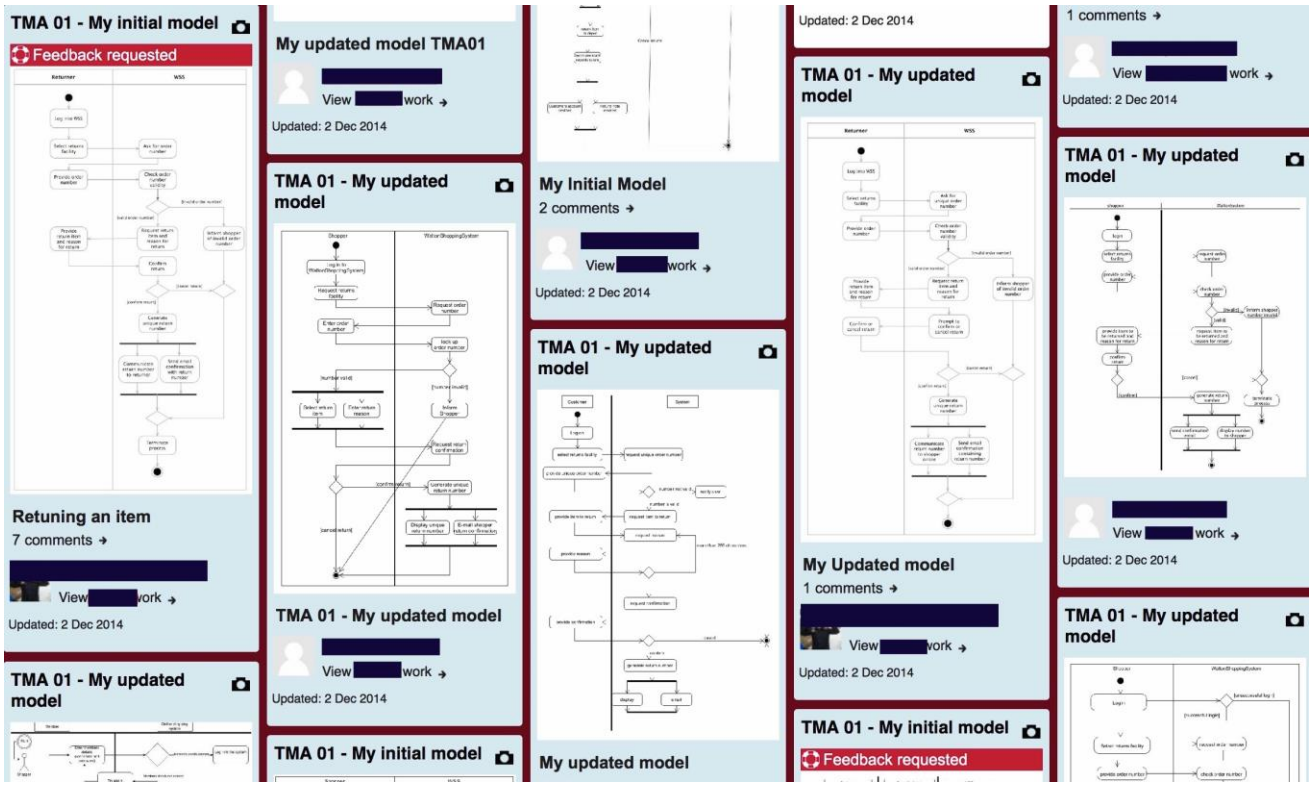

Figure 1: OpenStudio as used in TM354

Students were offered the opportunity of an icebreaker activity to familiarise themselves with the OpenStudio environment before they started work on their first assessment. They also performed a preparatory activity to help them with giving feedback.

4.2 The student survey

A survey was sent to 300 students and had a response rate of 13.6% (n = 42); this response rate is not untypical in the OU's distance learning context. The survey consisted of 11 questions to find out:

- whether students had uploaded their work; the clarity of instructions given; how well prepared they felt for giving feedback to their peers;
- whether they had viewed the work of other students; whether they had commented on other students' work; whether they had received comments from fellow students;

- whether their work had improved as a result of looking at other students' models; whether they became aware of the range of possible solutions; whether their work improved as a result of receiving comments from peers;
- whether they gained a sense of what it means to share an understanding of a model with peers; and whether they enjoyed carrying out the activities.

Students were also asked to add any further comments.

The results of the survey revealed that:

- 98% of the students had uploaded their work; 93% agreed that the instructions were clear; 85% felt adequately prepared to give feedback.
- 93% viewed the work of other students; 98% commented on the work of other students; 92% received comments from other students. (The high percentages of participation are not surprising, as the OpenStudio activities were a compulsory part of the assessment, although only 15 to 20% of the total marks.)
- 63% felt their work improved with the feedback; 67% agreed they had learned from feedback.
- 85% agreed they saw a range of possible solutions; 77% agreed they had shared an understanding.

A qualitative analysis was also carried out on the comments made by respondents, which helped to illuminate the findings from the quantitative data.

Students agreed that viewing others' work helped them realize that there are *'multiple possible solutions for the same problem, helping me to improve my work'*, and that '*It was interesting to see the variety of approaches to the problem'.*

Some students felt insecure about reviewing others' work: "*I did look but I wasn't sure if their work was up to standard and I had my own opinion so only by looking at other peoples work didn't learn anything new*"

Commenting was seen as a positive activity: "*[I] guess my comments help other students. My comments are based in my experience and knowledge trying to be constructive*".

However, some students thought that it would be better if comments came from a tutor: "*As someone who was still studying the material myself, I was not fully convinced as to whether all my comments were accurate and I'm sure a qualified lecturer could have given a much more useful response*".

The final step of the OpenStudio activity required reflection by each individual student upon the comments received, and action to make changes to their model, if appropriate and justified: "*It was good see other people arriving at a similar solution to me and I could take on board ideas for improvement by comparing my work to theirs; I found the comments very useful and was able to make changes accordingly*".

However, some students resisted the idea of collaboration, as it imposes unwanted dependencies on others: "*Was done because it had to be done, not valued or taken on board. I didn't study through the OU to be working with others, I have enough work colleagues and work in the SDLC [software development life cycle] so teams are big with good communication*".

Figure 2 illustrates the richness of feedback comments that some students made to peers within OpenStudio.

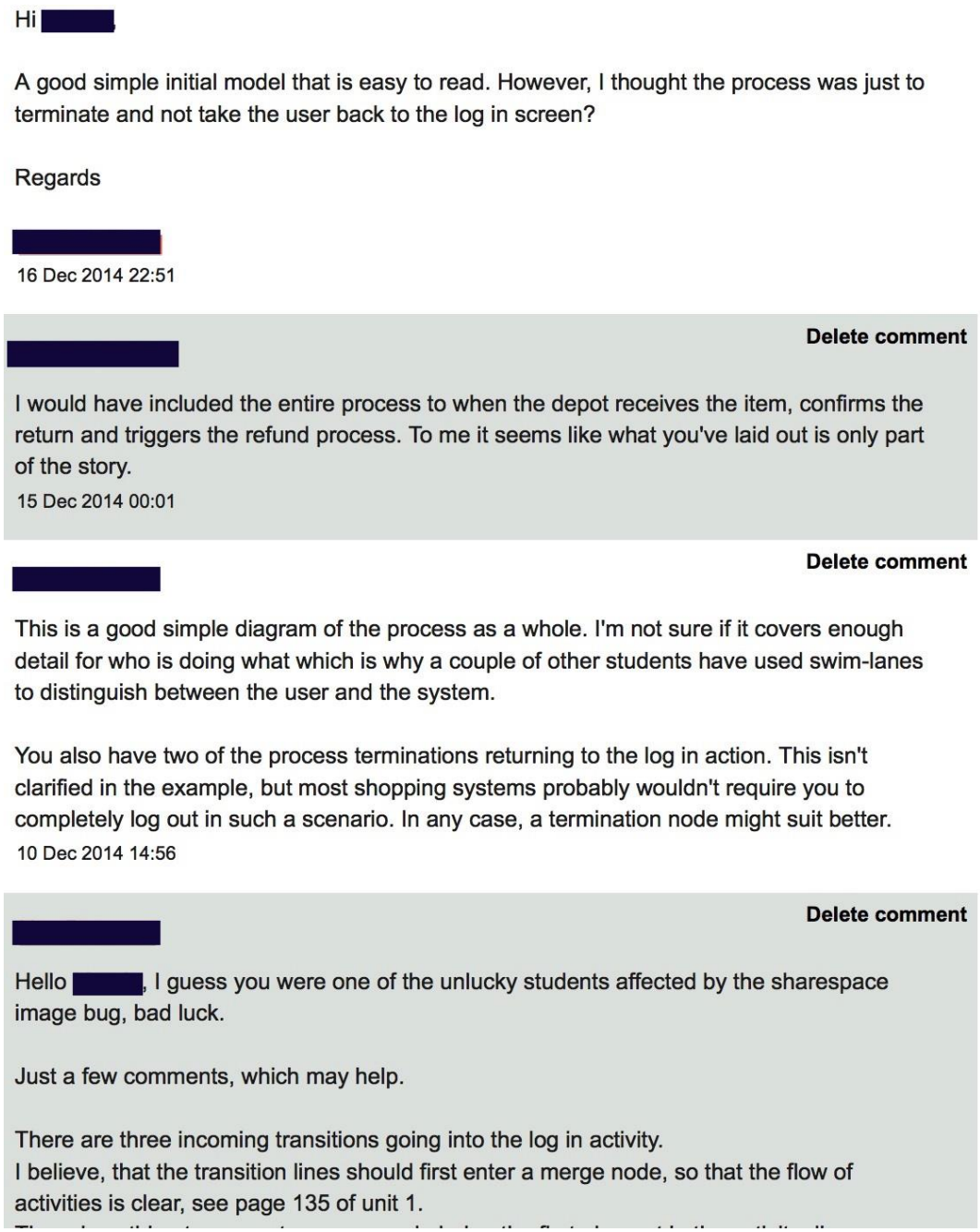

Figure 2: Some TM354 students' comments[3] in OpenStudio

4.3 The tutor survey

The tutor questions were posted as threads in a tutor forum, and the discussion was kept open for a couple of weeks. There was an average of 9 replies per question from a total of 17 tutors. Tutors were asked:

- whether their students needed help uploading their work; what kind of support students needed; whether they felt their students were adequately prepared for giving peer feedback;
- the level of engagement of their students with the OpenStudio activities; whether tutors needed to contact students to encourage engagement;

[3] Sharespace is the name given to OpenStudio in TM354

- whether the activities provided a richer learning experience; and whether richness and variety of learning was reflected in students' answers to the assessments.

All but one of the tutors said that their students needed no help in uploading their work, but some reported some queries from students about why they could not initially see others' work. This was done intentionally; students' work was only visible to other students once they had uploaded their own. All tutors thought that students were adequately prepared to give feedback although the quality of feedback varied.

The engagement of students was very high, as mentioned above, partly because the activities were compulsory. However, timing was an issue: if students' work was not uploaded in time, and comments were not given in time, this would affect follow-up discussions. Most tutors had to send out reminders to some students.

All but one of the tutors agreed that the learning experience had become richer because of the OpenStudio activities, as students were able to '*learn from each other and to appreciate that there may be several valid solutions to a particular problem*'. Several tutors commented that the activities helped students to interact, see other solutions and reflect on differences.

Tutors also felt that the richness and variety of students' learning was reflected in their assessment answers, although some tutors were disappointed with some of their students' reflection on the process: '*had hoped they would have been able to say what they thought of the feedback they were given, and why they accepted/rejected it ... all too often they just enumerated the changes they'd made to their work*.'

5. Discussion of results

The use of OpenStudio in TM354 was intended to both simulate agile practices, such as the daily stand-up meeting, and to support the development of a set of skills, in the context of teaching agile software development. The overwhelming feedback from students and tutors was positive and this tool will continue as an integral part of the learning and assessment in TM354.

The emphasis in the design of the activities was on the development of skills rather than on the execution of the practices. This is justified as without the skills the practices cannot be easily adopted; also the teaching of agile practices fits better in a project module (which TM354 is not). However, the engagement of students with an iterative process of presentation, feedback and reflection can be considered as giving them the flavour of an agile practitioner context.

Using OpenStudio has helped students to gain a more realistic perspective of how modelling should be carried out in agile software development, and it has promoted rich interaction among students. Many students gave valuable feedback, and tried to reach a shared understanding of what the models represented. There were a considerable number of high quality comments, as shown in Figure 2.

There are, however, aspects that need to be improved: in particular, the quality of the reflection upon feedback (Kear et al., 2016; Walker, 2015). Students' reporting of how feedback had contributed to the improvement of their models was at times shallow and lacked a good critical discussion. One tutor commented: '*[students] were happy to say where the model they were commenting on could be improved, but did not always relate differences between the model and their own'*. Also the quality of feedback across the student cohort was variable, with some tutors mentioning that they had to provide extra support, as not

many students '*were familiar with critically evaluating other people's work in an academic environment such as this.*'

We prepared students for giving feedback by running a preparatory activity; this was however, a fairly generic activity prepared by the University Library. We could customise this activity to more specific feedback on models and their understanding. A similar preparatory activity could be tried for engaging students in a critical discussion of options and rationale for decisions they take while modelling.

6. Conclusion

This paper presented a case study of a studio-based approach to support the teaching of agile software development at a distance. The case study and its evaluation demonstrate that the approach supports the development of a set of key transferable skills: sharing, discussing, giving feedback and reflecting on feedback received. These skills are all important for agile software development. The approach also gave students a flavour of some activities undertaken in an agile practice context. The feedback collected from tutors and students was very positive, so we will continue to use this approach in the module.

The evaluation suggested ways in which the approach can be improved. Students can be given more specific support and preparation, in particular for the development of deeper critical reflection and academic evaluation. Improvements are needed to the stage of the approach where students reflect on the feedback they have received, and decide whether and how to use it. We will be focusing on this aspect in future, and will provide preparatory activities to support students in reflecting on, and using, feedback.

The success of the studio-based approach in TM354 suggests that the approach can be extended to other activities and modules. Other OU modules that have used OpenStudio have experienced similar positive outcomes. In particular, the studio learning approach supports the development of core skills in demand by employers, such as: negotiation; receiving feedback constructively and incorporating it into one's own development; and application of analytical and critical thinking skills.

References

Brocke, C. vom. (2011). How to Leverage Virtual Learning Communities for Teaching Agile Communication Skills? The eGroups Case at the University of Münster in Germany and Massey University in New Zealand. *Knowledge Management & E-Learning: An International Journal*, *3*(4), 644–664. Retrieved from http://www.kmel-journal.org/ojs/index.php/online-publication/article/viewArticle/147

Browne, J. (2010). *Securing a Sustainable Future for Higher Education*. *Independent Review of Higher Education Funding and Student Finance*. Retrieved from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/31999/10-1208-securing-sustainable-higher-education-browne-report.pdf

Cennamo, K., Douglas, S. A., Vernon, M., Brandt, C., Scott, B., Reimer, Y., & McGrath, M. (2011). Promoting creativity in the computer science design studio. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education - SIGCSE '11*, 649. http://doi.org/10.1145/1953163.1953344

Devedžić, V., Milenković, S. R., & Deved, V. (2011). Teaching Agile Software Development: A Case Study. *IEEE Transactions on Education*, *54*(2), 273–278. http://doi.org/10.1109/TE.2010.2052104

Holmstrom, H., Conchúir, E. Ó., Ågerfalk, P. J., & Fitzgerald, B. (2006). Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance. *CGSE '06.*

International Conference on Global Software Engineering, 3–11.

Hummel, M., Rosenkranz, C., & Holten, R. (2012). The Role of Communication in Agile Systems Development: An Analysis of the State of the Art. *Business and Information Systems Engineering*, *5*(5), 343–355. http://doi.org/10.1007/s12599-013-0282-4

Hundhausen, C. D., Narayanan, N. H., & Crosby, M. E. (2008). Exploring studio-based instructional models for computing education. *ACM SIGCSE Bulletin*, *40*(1), 392. http://doi.org/10.1145/1352322.1352271

Jalali, S., & Wohlin, C. (2010). Agile practices in global software engineering - A systematic map. *Proceedings - 5th International Conference on Global Software Engineering, ICGSE 2010*, 45–54. http://doi.org/10.1109/ICGSE.2010.14

Jones, D., & Lloyd, P. (2013). Which way is up ? Space and place in virtual learning environments for design. In *2nd International Conference for Design Education Researchers* (pp. 552–563). Oslo.

Kear, K., Donelan, H., & Williams, J. (2014). Using Wikis for Online Group Projects : Student and Tutor Perspectives.

Kear, K., Jones, A., Holden, G., & Curcher, M. (2016). Social technologies for online learning: theoretical and contextual issues. *Open Learning: The Journal of Open, Distance and E-Learning*, *31*(1), 42–53.

Maher, P. (2009). Weaving agile software development techniques into a traditional computer science curriculum. *ITNG 2009 - 6th International Conference on Information Technology: New Generations*, 1687–1688. http://doi.org/10.1109/ITNG.2009.175

Oldfield, S. J., & Morse, D. R. (2007). Exploiting Connectedness in the Informatics Curriculum. *Innovation in Teaching and Learning in Information and Computer Sciences*, *6*(3), 27–46. http://doi.org/10.11120/ital.2007.06030027

Paasivaara, M., Durasiewicz, S., & Lassenius, C. (2008). Distributed agile development: Using Scrum in a large project. *Proceedings - 2008 3rd IEEE International Conference Global Software Engineering, ICGSE 2008*, 87–95. http://doi.org/10.1109/ICGSE.2008.38

Quinn, B., Barroca, L., Nuseibeh, B., Fernández-Ramil, J., Rapanotti, L., Thomas, P., & Wermelinger, M. (2006). Learning software engineering at a distance. *IEEE Software*, *23*(6), 36–43. http://doi.org/10.1109/MS.2006.169

Rodriguez, G., Soria, Á., & Campo, M. (2015). Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to scrum. *Computer Applications in Engineering Education*, *23*(1), 147–156. Retrieved from http://onlinelibrary.wiley.com.libezproxy.open.ac.uk/doi/10.1002/cae.21588/epdf

Shadbolt, N. (2016). *Computer science degree accreditation and graduate employability: Shadbolt review*. Retrieved from https://www.gov.uk/government/publications/computer-science-degree-accreditation-and-graduate-employability-shadbolt-review

Thomas, E., Barroca, L., Donelan, H., Kear, K., Jefferis, H., & Rosewell, J. (2016). Online conversations around digital artefacts: the studio approach to learning in STEM subjects. In S. Cranmer, M. de Laat, T. Ryberg, & J. A. Sime (Eds.), *10th International Conference on Networked Learning 2016*. Lancaster.

Walker, M. (2015). The quality of written peer feedback on undergraduates' draft answers to an assignment, and the use made of the feedback. *Assessment & Evaluation in Higher Education*, *40*(2), 232–247.

Wermelinger, M., Hall, J. G., Rapanotti, L., Barroca, L., Ramage, M., & Bandara, A. (2015). Teaching software systems thinking at the Open University. *2015 IEEE/ACM 37th IEEE International Conference on*

Software Engineering (ICSE), 16-24 May 2015, *vol.2*, 307–310. http://doi.org/10.1109/ICSE.2015.161

Yu, X., & Petter, S. (2014). Understanding agile software development practices using shared mental models theory. *Information and Software Technology*, *56*(8), 911–921. http://doi.org/10.1016/j.infsof.2014.02.010