# Observations of a Software Engineering Studio: Reflecting with the Studio Framework

Christopher N. Bull
*Lancaster University, UK*
*c.bull@lancaster.ac.uk*

Jon Whittle
*Lancaster University, UK*
*j.n.whittle@lancaster.ac.uk*

## *Abstract*

*Studio-based learning for software engineering is a well-received concept, despite its apparent lack of uptake across institutions worldwide. Studio education affords a variety of highly desirable benefits, and is also popular amongst its students. This paper presents Lancaster University's software engineering studio, details of its implementation, observations made throughout its first year, evidence of its successes, and reflections against the recently defined studio framework. This paper aims to provide useful information for anyone that is considering utilizing a studio-based approach.*

## 1. Introduction

Studio-based learning in software engineering (software studios) is an alternative to traditional lecture-oriented courses, which promote several human-centric aspects of learning, including collaboration, mentoring and peer-learning. Despite interest in studio-based learning [1], there appear to still be very few implementations of software studios, yet the potential benefits of studio education are numerous [2] [3] [4]. An important element often left out of descriptions of studios is that they support several non-technical skills, such as communication and teamwork, which are deemed essential for any software engineer [5]. It is also a commonly held belief that "*students are motivated by assessment*" [6], and failure, generally speaking, is not conducive to grades. However, giving students permission to fail is important. As Jazayeri discussed, it is said that good judgment comes from experience, and experience comes from bad judgment [5]; studios provide enough flexibility for students to not only make those mistakes, as part of their learning experience, but to also reflect and overcome them.

Traditional lecture-based education is prominent in the computing discipline, which is echoed by others, stating that "*Instruction continues to be conducted in the traditions established when computer science emerged from mathematics decades ago*" [3]. Another opinion is that lectures frequently teach through memorisation, rather than teaching understanding [7]. Conversely, studios are environments for students to practice and learn by doing. Studios originated centuries ago from disciplines such as architecture, design and art; they are often project-based, encourage collocated students, support reflective practice [8], and they emphasise the use of soft skills – to name just a few elements. Previous work describes the importance of communication, during 'critiques', for the transmission of design knowledge in studios [9]. We have also recently gained a clearer understanding of what really constitutes studio education through the formulation of the 'studio framework' [4].

If publications are taken as example implementations of studio-based education, then besides the limited number in existence, they also significantly vary in execution and guiding principles; this in itself is not necessarily a negative point, as each will have their



978-1-4799-4970-0/14 © 2014 IEEE CSEE&T 2014, Klagenfurt, Austria

own successful elements. However, this makes it difficult to compare or benefit from shared implementation details. Success or failure of software studios are also "*usually assessed through end-of-term questionnaires*" [1], which may provide limited information on their own. Apart from the benefits a studio provides, one of the reasons to explore software studios further is because students enjoy them [1].

At Lancaster University we are doing our bit to explore this phenomenon by implementing our own software studio. This studio is used for Part II undergraduate students (second year and above) studying our Bachelor of Science (BSc) in Software Engineering. This paper describes the first year that our studio has run (2012-2013), including a description of how it is implemented, and a discussion about year-long observations. This is done by using the studio framework to reflect on our specific studio implementation.

## 2. Background

Interpretations of studio education for computer science and software engineering have been explored for over 20 years, with the earliest known studio implemented at Carnegie Mellon in 1990 [10]. Over this time there have only been a relatively small number of attempts at implementing a studio, yet project-based learning (PBL) has seen wider use. Studios share several similarities and benefits to PBL, but PBL does not determine how students should interact, for example, the culture of critique or the collocation of students in a studio. Studios are often project-based, or will have particular threads of project work, but a studio is more than just a project-based course.

Although there does not appear to be an up-to-date and exhaustive review of all studios in software engineering, there is still some interest; Carter et al have recently presented a brief overview of 5 studios in computer science [1]. Software studio implementations often report varying levels of success, but it has been indicated that, in certain studios, the student's content mastery is "*as much or more than students in the same courses taught in the traditional way*" and also that "*students in studio courses have shown higher levels of motivation and engagement than students in traditional courses*" [11].

Despite the apparent success of the limited number of documented studios, it is also apparent that none of them follow a shared definition or understanding of studio education – making comparisons difficult. Some point towards literature that only gives broad or vague descriptions, and a small number relied on the tacit knowledge of someone who originated from the disciplines that gave us studios, a consultant of sorts. Environments that were claimed to be studios may have been effective, but were they truly studios? Do they offer what studios in other disciplines offer? It is difficult to succinctly define studio practice due to the inherently complex nature of studio education. As such, previous work has provided a 'studio framework' to help solidify a definition and understanding of studio practices [4] in place of a single over-simplified definition. Due to their complex nature, studios should not be considered a binary state (i.e. whether it is or is not a studio), there will be some spaces which are more 'studio-like' than others. This framework is based on interviews with architects, designers and artists. Each of the categories within the framework, succinctly summarised below, are offered with an associated description:

***Physical environment*** – The room needs to be supportive of the categories in this list by generally being open and reconfigurable, providing students with control of the room, and also providing opportunities for a variety of group, individual and social spaces.

***Facilitation of studio*** – This relates to how the studio is managed. The students should be encouraged to use the space as they wish – encouraging a sense of ownership. Rules regarding the use of the space should not be restrictive, e.g. 24 hour access and allowing food and drink. Further, there should be small groups of students (approximately 10), and high availability of staff, encouraging richer interactions.

***Modes of education*** – A studio should provide a variety of education methods. Teaching staff fall into a coaching/mentoring role. There is a large emphasis on the self-learning process, supported by peer-learning elements, and further supported by flexible and impromptu teaching.

***Awareness*** – Studios should support greater awareness amongst its students. Visual work is recommended, as well as placing work on display (as work-in-progress or final products). Visibility of work helps students see other's work, improves capability to reflect, and increases and improves social interactions.

***Critique*** – This is an important part of reflective practice. Critique is used for providing feedback and developing ideas. It occurs in multiple formats (formal and informal, group and individual) and should come from peers (e.g. peer-coaching), as well as staff.

***Culture*** – Widely agreed as the most important aspect of studio education. A studio culture should be social and foster a sharing culture, and yet sensitive to supporting a good work ethic – which also helps support peer-learning elements. Students' attitudes should point towards treating the studio like a second home. Serendipitous interactions are also very important.

***Individual's characteristics*** – Despite the studio often being described as open and for groups of students, the studio should support the students as individuals too. This is achieved through offering private and quiet spaces, and also allowing and encouraging personalisation of space.

***Inspiration*** – When designing, students should be encouraged to be creative in their designs and solutions, which is helped by supporting inspiration. This is improved by students being in close proximity with each other and allowing the studio to be playful. Having the studio contain extra materials or media relevant to their work can also help.

***Collaboration*** – Collaborative activities are common in studio education. To better support collaboration a studio should support spaces for organised and impromptu collaboration, and also contain equipment to support these interactions.

***Digital technology*** – Studios do not require digital technology; whilst all of the other categories refer to aspects that should exist within a studio, this one is a warning about the use of certain digital technologies potentially diminishing the studio; e.g. reducing social interactions and visibility of work. However, it can improve access to work.

## 3. Implementation

The software engineering studio in BSc Software Engineering is a new endeavour at Lancaster University, which offers a dedicated studio for its students. The intention is that they can use it like a 'second home', spending the majority of their time there, and it forms a central component to their education. The studio started in the academic year 2012/13, and was occupied by second year undergraduate software engineering students. In subsequent years it will be occupied by second and third year students.

### *3.1. Guiding Principles*

The intent of Lancaster's software studio, as set out by course proposal documents, is to serve as "*both a lab for students engaged in conceiving, designing and developing software products as well as an approach for teaching software engineering in the lab which emphasizes practical hands-on work and experimentation with a variety of software engineering techniques.*" Further to this brief description, a simple list of five guiding principles were provided, to better serve the teaching staff in understanding the purpose of the studio and the roles they will play:

1. "*A focus towards on-demand, self-directed learning that favours practical experimentation over traditional lectures*"
2. "*A cohort approach to training in which groups of students work together in a mutually supportive shared space*"
3. "*An emphasis on experimentation where students gain hands-on knowledge of competing software development processes and techniques and learn how to choose between them based on practical experience*"
4. "*Close collaborative work with academic staff who act as mentors rather than instructors*"
5. "*A focus on underlying principles rather than the fads and notations of the day*"

### *3.2. Curriculum*

The degree provides compulsory modules for second and third year software engineering students, but also shares core modules with the computer science degrees, to more efficiently teach essential skills and concepts that are common across the degrees. All first year students, software engineering and computer science, do the same traditional lecture-based initial year – they specialise and diverge once entering their second year. The studio was not implemented across the entire degree primarily due to the practicalities of implementing a new course and integrating it with the current curriculum, however, the studio modules do form a significant part of the degree. There is also an unanswered question as to whether studio-based education is suitable for all types of course content, for example algorithms. The studio is currently associated with three modules:

- *SCC230 Core Studio (Second year)* – This is the module observed in this paper. The students put a variety of learned software engineering techniques into practice in a group project, giving them hands-on practical experience within small groups. Projects were done in 3 groups of 4 students. This module also introduces the students to the studio. It has timetabled studio sessions, called "workshops", every two weeks, for three and a half hours over the first two terms (20 weeks, October-April). Group supervisors will also meet with their groups weekly.
- *SCC330 Networked Studio (Third year)* – Students work in larger groups, 2 groups of 6 students. The emphasis of this module is the implementation, integration and networking of software modules.
- *SCC331 Live Studio (Third year)* – All students in the year form a single group, and work towards a very large project. They will experience industrial size and strength projects, with a focus on building a live system that will be deployed and could have commercial or research value.

Second year students also take SCC204 Software Design alongside their studio module, a non-studio lecture-based software engineering module shared with computer science students.

It provides them a basic theoretical underpinning of software engineering concepts. The studio modules have no lectures, and are 100% coursework-based assessment. Instead of lectures, the emphasis is on coaching and regular workshops, which are intended to be interactive sessions, with a variety of staff present. The students are allowed and expected to use the studio outside of their timetabled sessions. A lot of emphasis is put on the significance of the studio module, and a large amount of self-study is necessary outside of the workshops.

### *3.3. The Room*

Physical space is only part of the studio puzzle, but an important one nonetheless. Below is a floor plan, figure 1, which represents the basic layout of the room:

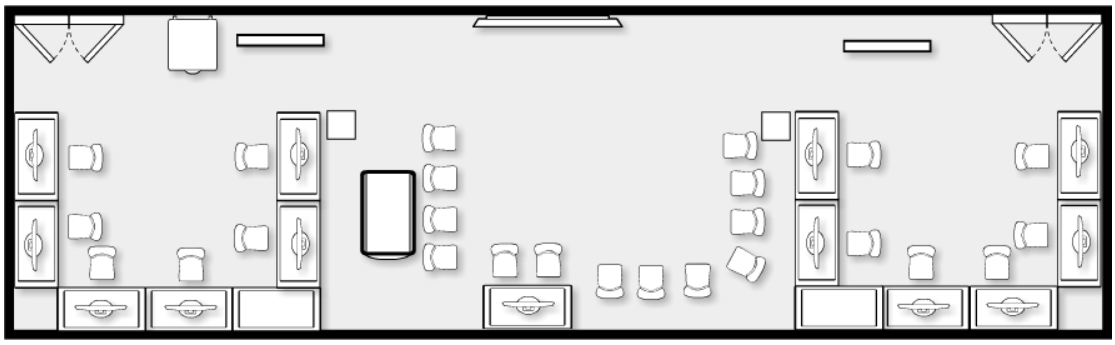

**Figure 1. Studio floor plan**

The room is split into three main areas: two areas with computers laid out on tables in a 'U' shape (on the left and right side of the room), and a central area with no desks but many chairs – providing space for activities away from computer screens. The left and right sections are where the students do the majority of their individual and group work. The middle area is used for activities that benefit from the freedom of space, such as conversations, presentations and impromptu demonstrations. There is also a number of free-standing whiteboards mounted on wheels in the studio, which can be moved to change the dynamic of the space.

## 4. Methodology

Data gathering was conducted throughout the academic year of 2012-2013, and primarily involved year-long ethnography (specifically, participant observation) and is complimented by student feedback through questionnaires and a focus group. The student feedback is used to provide the students' perspectives, and forms part of the ongoing enquiry to continuously improve the studio. Briefly summarising all of the feedback: the students enjoyed the studio and the workshops, they used the space a lot outside of timetabled sessions, found the module intense, but requested very little to be changed.

### *4.1. Participant Observation*

The primary method of information gathering on the studio; observations were made and field notes taken whilst the researcher was immersed in the environment (not simply as an onlooker). Observations were planned throughout the entire module, during the workshop sessions (over a 20 week period); the researcher took on the role of a teaching assistant (TA) as well as that of the observer. By taking on both roles simultaneously, it allowed the observer to perform 'moderate participation' [12]. Being a TA to the students allows a mixture of direct

involvement with the students, but also provides a level of detachment from them, allowing for objectivity during the sessions (TAs do not constantly interact with students). The students were made aware that this particular TA would also be observing.

Observing as a TA allowed the researcher to follow Howell's phases of participant observation [13]: 1) establish rapport, 2) submerge into community, 3) record observations, and 4) analyse data. Moderate participation does not necessarily lend itself to establishing a rapport and submerging into the community as much as greater levels of participation (i.e. active or complete participation). However, in the researcher's experience, TAs build a strong rapport with their students. Being perceived and utilised as a TA allows observation of behaviour from a teaching perspective, yet TAs are generally not strictly treated as an authority figure. TAs frequently have friendly interactions with the students, and are very approachable.

### *4.2. Analysis*

The field notes taken during the participant observation were analysed by coding the observations and identifying themes in the data, then reflecting on them with the studio framework. Observations made in the field notes were consistent with the feedback from the student questionnaires and focus group.

## 5. Reflecting on Lancaster's Studio

Reflections on the studio were made frequently throughout the year. A sample of these reflections is given below, placed in the context of the studio framework, each briefly describing how Lancaster's studio has satisfied a particular category, or not, with concrete examples given.

***Physical environment*** – The room is laid out to be spacious and supports a lot of movement. The layout of the room, shown earlier in figure 1, was designed to support a variety of activities, and is particularly suited to group activities. It is also intended to be flexible, with equipment and furniture expected to be moved around the room to suit the students' needs. The room is secured by card access, which ensures that the students can leave personal or work items in the room and on the sides, providing an element of comfort.

One example of flexibility observed in the room was the use of the wheel-mounted whiteboards, which were frequently moved around: they were moved into the group's area when used, put against the wall when not used (out of the way) and moved near the group when used as a point of reference (but not physically interacted with). Although we provided the students with permission and the ability to be flexible with their environment and furniture, it was not used as we expected. Fortunately, that flexibility manifested itself in another form: their use of the moveable whiteboards. These created on-the-fly changes to the dynamic of the space, which alleviated the fact that there were no dedicated individual or private spaces. There were also no dedicated social spaces, as space is a premium, but this was balanced by the social culture that the students adopted.

***Facilitation of studio*** – The way a studio is facilitated will have a significant impact on the studio and the culture within. For this reason the staff do not dictate how the space should be used and have always encouraged the students to use it as they require (e.g. not allocating desks to students). This explicit encouragement from the staff has been very useful; staff members coming to terms with handing over responsibility of the room to the students is one thing, but the students will still share a frame of mind that restricts their activities within the space, because technically they are guests in the university

owned space. This is why making the expectations explicit is important, and even reminding them during the first few weeks. Further, we told our students at the start of the course that they could be as flexible with the space as they wanted (move tables, chairs, desktop computers etc.). In practice this rarely happened, finding instead that the space should support semi-permanent placement of students, as they tended to "nest" themselves in particular places within the environment and stayed there.

To provide a space conducive to the amount of time the students will inherently be in the studio, we allowed 24 hour access to the room (achieved through keycard access). Further to this, creature comforts were allowed, most of which are not necessarily found in normal computer labs; these include a printer, 2 wall-mounted whiteboards, 3 wheel-mounted whiteboards, a wall-mounted touch TV, a horizontal touch table, bookshelves, cupboards, coat stands, a plumbed water cooler, a fridge and a kettle. All of these enable the students to work as long as they need in the studio – becoming a more comfortable working environment than a computer lab. We also allowed students to eat and drink in this studio, which is not common in computer labs, but is beneficial [14].

Our approach to the studio meant that we had multiple members of staff in all workshop sessions, which led to richer interactions between the students and staff. Constant questioning [10] is also implicitly used in our studio, which is reinforced by having multiple staff members' perspectives, as it encourages the students to think through problems and encourages reflective practice. Questions inherently invoke reflection.

Lastly, some of our staff had difficulty, at the start of the course, transitioning to studio education. They employed techniques they were familiar with and performed lectures in a few of the earlier weeks. This undermined the purpose of the studio in those specific workshop sessions, but was quickly addressed. Based on student feedback, these were also the sessions the students gained the least from.

***Modes of education*** – Our studio is geared towards practical group-based learning, and the flexibility of educational approaches has been heavily supported by the attitudes of the staff and the layout of the studio space. Workshops are longer than normal lecture sessions, at 3 and a half hours, and there is usually a theme for individual sessions, but that does not mean that the theme will be the only topic explored. The length of the sessions allows for topics to be introduced and then practiced in the context of the student's projects; a sizable chunk of teaching staff's time is spent mentoring and critiquing individual groups. Often the staff have performed impromptu teaching, taking an associated topic, or an element the students are struggling with and performing a quick demo on-the-fly. Techniques used in the workshop sessions typically included practical exercises, student presentations and demonstrations, discussions and critiques.

The relationship between staff and student is based on mentoring and coaching. This better supports flexible teaching as well, because the staff members can be more in tune with the student's efforts and understanding. It is also reinforced by the practice of constant questioning, asking appropriate questions during coaching to encourage reflective thinking in the students.

Another observation was the practice of peer-learning. This was not explicitly planned for this year, but at times it emerged that the students were teaching each other. This was observed as a result of group work, but it also occurred across groups; all 3 groups were working on very different projects, but sometimes there were mutual or similar problems.

***Awareness*** – The first thing that becomes obvious when walking into our studio is all of the work around the room, on the walls and on the whiteboards. Software engineering

is inherently not a very visual field, but the students have made their ideation, diagrams and processes very visible to all in the studio. The studio is also very social, so the students have been quite familiar with how the other groups were getting on.

The most prominent aspect of our studio that favours awareness was through the use of the whiteboards – they provide constant visualisation of work. When they are not directly interacted with, they are often pulled up close to the group and used as reference material, and are fairly clear from a distance. This also helps promote multiple tiers of vertically positioned information – if everything is placed at similar heights as monitors then the information can easily be obscured.

***Critique*** – This is helped most through the approach of coaching and constant questioning. It can be used in various forms of critique (formal, informal, group and individual), all of which are utilised in our studio, and encourages the students to reflect on their project, process and even the answers they are giving to the questions. The students show off their work and get in-depth feedback every week, with the intention of developing ideas and solutions throughout the project – allowing the students to make mistakes, but crucially, learn from them. They have 'permission to fail'.

Soft skills also see thorough use through the various and frequent methods of critique, and there was also a distinct emphasis, from the staff, on using and improving soft skills. There was also an entire studio session at the end of the year dedicated to presentation skills, verbal communication (using famous speeches) and written communication.

***Culture*** – The students have stated that when working with other computer science students in one of the shared modules, they would work in the computer science lab (a traditional computer lab). But when working on their software engineering work, and often when working on individual computer science work, they worked on it in the software engineering studio. The students say that this is because it is "our space" (pointing to a sense of ownership of the space) and that it was often a better work environment. The students have referred to the studio as "homely".

The studio also enables greater relationships between staff and students, making them much more approachable and also having a playful dynamic to some of their encounters. This builds a stronger rapport.

***Individual's characteristics*** – There are no dedicated private and quiet spaces, although this did not present itself as an issue. The lack of these spaces is likely mitigated by the fact that the studio is large and is geared towards group work, and also that no sensitive projects or data are handled. There have also been a few times when a group segregated themselves from the rest of the room using the wheel-mounted whiteboards.

In fact, the students see the studio as the quiet space when compared to the computer lab in the computer science degree, as stated in a summary of the fourth survey given by the student representative: "*As a collective we value having the quiet space separate from the rest of computer science very much! It's a great space to collaborate and work independently and the whiteboards are [a] great asset*" – in reference to the free-standing, moveable whiteboards. What makes it a "quiet space" is not definitively known, but it could be a mix of: a) fewer students than the CS lab, and b) less disruptive, as the CS lab will have multiple classes and topics throughout the day.

***Inspiration*** – Outside of the workshop sessions, music is often played through the wall-mounted TV, but on occasion a student would show something that they found

personally interesting. The music is not disruptive because everyone consents to it. During the focus group it became very clear that the students were inspired to work on their projects because they came up with the project idea, and therefore had a vested interest in it.

***Collaboration*** – With most work in our studio being group-work oriented, supporting collaboration is very important. The tables laid out in 'U' shapes, seen in figure 1, as well as the centre of the room, provide a starting point for a variety of collaborative activities. The single most used element for collaboration in our studio was the wheel-mounted whiteboards. Primarily a piece of equipment to support collaboration, but due to the fact that they are easily manoeuvrable, they also provided the ability to create impromptu collaborative spaces – they allow the students to change the dynamic of the room, as they see fit. These whiteboards, as well as 2 other wall-mounted whiteboards, saw a significant amount of use, and the students often mentioned how indispensable they were to them. They were used for many things, including informal notation as well as structured diagrams, both are great for externalising information and supporting face-to-face communication [15].

***Digital technology*** – The studio framework states that the use of digital technologies is detrimental to the studio experience, which is an interesting dichotomy when used for software engineering. The problem exists because computers reduce social interactions and the visibility of work. The students were not forced to use any particular technology, which is most obvious through their frequent use of the whiteboards during group and ideation sessions. Another tactic that was employed was through the use of pair programming. Some of the students decided to employ that method as it suited them, not because they were forced to try it out.

Also noteworthy was the limited use of the wall-mounted touch-screen TV, losing out to the whiteboards. It was primarily used for presentations, but also for playing music videos in the background.

## 6. Conclusions

Studios are a very beneficial educational approach, replacing or sometimes complimenting traditional teaching methods. In this paper we have shared an insight into our studio implementation, and reflected on ethnographic observations with reference to the 'studio framework' [4].

Our studio was initially only intended for smaller groups of students, which resonates with the studio framework, although ideas for future work include scaling up the course. The small group size has enabled numerous rich interactions between the staff and students, but the staff have not necessarily put more time into a studio than a lecture-based counter-part. Whilst there are no figures to support this, the staff do less preparation than lectures – studios focus more on on-the-fly coaching.

Our studio was made possible by being able to get hold of a crucial resource: a sizable dedicated room. Space is an expensive commodity in universities, so it may not be feasible for all institutions. Further investigation is needed to explore the implications of limited resources on studios.

Based on the observations and student feedback, one significant aspect of our studio was the use of the wheel-mounted whiteboards. A seemingly simple asset, these provided a wide range of benefits, and were always being used: supporting collaboration, changing

physical layout of the room through positioning and allowing greater awareness to other students, to name a few. Another prominent aspect was the approach of constant questioning [10], which is inherent in our studio approach. It helps build a culture supportive of critique and encourages further reflective practice in the students.

Our studio has been successful so far, but as the students are yet to finish the degree, it is too early to tell if they are achieving better grades. However, several of the students have indicated that the studio module is their favourite of all modules in their degree – not bad for the first year of any course. Beyond the students, the staff are also enjoying the studio, putting their time into the studio because they want to. What really excites us is seeing the students, now in their third year, enjoying the course even more – if that was possible.

## Acknowledgment

This work was undertaken as part of the post-disciplinary HighWire CDT programme at Lancaster University. HighWire is funded by the Digital Economy Programme: a Research Councils UK cross council initiative led by EPSRC and contributed to by AHRC, ESRC and MRC.

## References

[1] A. Carter and C. Hundhausen, "A review of studio-based learning in computer science," *J. Comput. Sci. Coll.*, vol. 27, no. 1, pp. 105-111, Oct, 2011.

[2] O. Hazzan, "The reflective practitioner perspective in software engineering education," *J. Syst. Softw.*, vol. 63, no. 3, pp. 161-171, Sept, 2002.

[3] C. D. Hundhausen et al., "Exploring studio-based instructional models for computing education," *SIGCSE Bull.*, vol. 40, no. 1, pp. 392-396, Mar, 2008.

[4] C. N. Bull et al., "Studios in Software Engineering Education: Towards an Evaluable Model," in *Proceedings of the 35th International Conference on Software Engineering (ICSE '13)*, San Francisco, CA, USA, 2013, pp. 1063-1072.

[5] M. Jazayeri, "The education of a software engineer," in *Proceedings of the 19th International Conference on Automated Software Engineering*, Linz, Austria, 2004, pp. xviii-xxvii.

[6] J. Süß and W. Billingsley, "Using continuous integration of code and content to teach software engineering with limited resources," in *Proceedings of the 34th International Conference on Software Engineering*, Zurich, Switzerland, 2012, pp. 1175-1184.

[7] E. Mazur, "Farewell, lecture?," Science, vol. 323, no. 5910, pp. 50-51, Jan, 2009.

[8] D. Schön, *Educating the Reflective Practitioner*. San Francisco: Jossey-Bass, 1987.

[9] B. Uluoğlu, "Design knowledge communicated in studio critiques," *Design Studies*, vol. 21, no. 1, pp. 35-58, Jan, 2000.

[10] J. Tomayko, "Teaching software development in a studio environment," *SIGCSE Bull.*, vol. 23, no. 1, pp. 300-303, Mar, 1991.

[11] N. Narayanan et al., "Transforming the CS classroom with studio-based learning," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, Raleigh, NC, USA, 2012, pp. 165-166.

[12] K. M. DeWalt and B. R. DeWalt, "Observation and Participation," in *Participant Observation*. Lanham: Rowman Altamira, 2002, ch. 2, pp. 20-23.

[13] J. T. Howell, *Hard living on Clay Street: Portraits of blue collar families*. Garden City, NY: Anchor Press, 1973.

[14] D. Herrick, "Food and drink in computer labs: why not?," in *Proceedings of the ACM SIGUCCS 40th annual conference on Special interest group on university and college computing services*, Memphis, TN, USA, 2012, pp. 161-164.

[15] M. Cherubini et al., "Let's go to the whiteboard: how and why software developers use drawings," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2007, pp. 557-566.