A REVIEW OF STUDIO-BASED LEARNING IN COMPUTER SCIENCE*

Adam S. Carter and Christopher D. Hundhausen
Human-centered Environments for Learning and Programming (HELP) Lab
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752 USA
{cartera, hundhaus}@wsu.edu

ABSTRACT

Studio-based learning (SBL), a pedagogical technique that promotes learning through the iterative construction and review of problem solutions, is becoming increasingly popular in computer science education. This paper reviews the use of SBL in computing education with respect to the ways in which it has been implemented, its impact on the curriculum, and the educational outcomes it has promoted.

INTRODUCTION

The notion of studio-based learning in formal educational settings has been around for almost a century, dating back at least to the Bauhaus School of Design [1]. One of the best philosophical discussions of the impact and importance of SBL can be found in Donald Schon's classic book *The Reflective Practitioner* [2]. There, Schon introduces the idea of reflection in action, the idea that experts often evaluate and critique their actions as they take place. Through this process, experts "[carry] out an experiment which serves to generate both a new understanding of the phenomenon and a change in the situation" [2]. As discussed by Schon, this reflective process naturally takes place in the architectural design studio.

In the SBL model observed and described by Schon, the instructor typically assigns the class a design specification that students must then implement. Throughout the project's lifecycle, the instructor holds periodic design reviews with each student. These reviews follow a format that guide the student to think more deeply about their solution

---

and the problem at large. From this reflection, students gain a better understanding of both the immediate problem, and of architecture as a whole.

In addition to informal design discussions, students also participate in formalized design "crits" where their designs are presented to other students, instructors, and outside critics. Schon does not discuss the implications of these more formal critiques, but it is presumed that these too have educational merit. From this description, we can identify the following key characteristics of studio-based learning:

- Classroom assignments should primarily be project-based.
- Student work should be periodically evaluated both formally and informally through design critiques.
- Similarly, students should be required to engage in critiquing the work of others.
- Design critiques should revolve around the artifacts typically created by the discipline.

How do these broad characteristics apply to computer science? Computer science has traditionally been taught as either a form of mathematics or engineering [3]. However, as noted by Docherty et al. [3], computer science is different from mathematics and other engineering disciplines in that computer science is not a "tame" discipline. Metrics of success and stopping criteria are often unclear. As such, correct solutions are often developed and evaluated iteratively. Docherty et al. note that these unclear characteristics of computer science are also shared by other design sciences (e.g. architecture), and argue that computer science education should be similar to the educational techniques used by these other design sciences.

Perhaps because of its similarities to other design sciences, we believe the SBL model is readily adaptable to the computer science discipline. In many computer science courses, classroom assignments are already project-based. Assignments are often designed to introduce a given programming principle or encourage a new way of thinking about a given topic. Furthermore, many colleges and universities make large capstone projects a requirement for graduation. Even more encouraging is the fact that like architecture, computer science generates its own set of design artifacts, including program code, architectural diagrams, algorithm visualizations [4], and even correctness proofs. As such, it is plausible that many of the same critiquing techniques discussed by Schon [2] could also be applied to the artifacts generated by computer scientists.

For computer science educators interested in integrating the SBL model into their courses, perhaps the most important open question is that of how best to integrate periodic reviews and critiques of computer science design artifacts into the curriculum. To explore that general question, this paper surveys several initial attempts to integrate SBL into the computer science curriculum. In so doing, we aim to address the following more specific questions:

1. What have been the motivations for implementing SBL in a particular course?
2. How was SBL actually implemented in the classroom?
3. How did researchers gauge the success or failure of their particular SBL implementation?
4. What general conclusions did the researchers draw?

EXPLORING STUDIO-BASED CURRICULUM IN COMPUTER SCIENCE

More than a dozen descriptions of the use of SBL in computer science courses have appeared in the literature. This section describes implementations at five different universities in terms of our exploratory questions. While we believe our review presents a representative sample of SBL implementations, it is by no means an exhaustive review, which is beyond the scope of this paper.

The University of Queensland, Australia

Researchers at the University of Queensland [3] noted several similarities between computer science and design sciences like as architecture. From this observation, the university developed the Bachelor of Information Environments degree, merging the three previously separate "streams" of computer science, interaction design, and project development into a studio-based curriculum. Each stream appears to be taught using traditional methods (i.e. lecture); however, the subjects taught in all three converge in a mandatory studio-based course. The studio courses revolve around the collaborative creation and presentation of projects that relate to what is being taught in each of the three streams. The effectiveness of the new degree program was assessed through attitudinal surveys. No exact numbers were given, but the anecdotal examples given in the paper suggest that students enjoyed the studio-based curriculum.

Monash University, Australia

Similar to the work done at the University of Queensland, Lynch et al. of Monash University redesigned their Bachelor of Information Management and Systems (BIMS) around the concept of SBL [1, 5]. The primary motivating factor for implementing a SBL-based curriculum stemmed from the desire to better prepare students for jobs in industry [5]. Lynch et al. noted that in addition to programming skills, employers are also interested in an employee's ability to collaborate effectively with coworkers, a skill that is often left untouched in traditional curriculum.

In redesigning the BIMS program, Lynch et al. [5] recognize that SBL needs to be more than just a classroom teaching technique. Thinking larger, Lynch et al. argue that SBL needs to be supported on three levels. First, the teaching space (physical layout of the classroom) needs to be set up in a manner to support collaboration amongst students. Most classrooms are designed for lectures, which make them difficult spaces for facilitating group work. Second, the course needs to be taught using SBL techniques. Lastly, coursework needs to be supported through a collaboration-oriented IT infrastructure. In other words, students need the technical tools necessary to make sharing work easy. The BIMS degree program addresses all three concerns. Classrooms were redesigned, SBL was the dominant teaching technique, and the latest software-based groupware was made available to the students.

SBL was implemented through a mandatory "studio" course to be taken alongside other coursework that was presumably taught using more traditional pedagogies. Similar to what was described by Schon [2], the main focus of the studio class was for groups of students to generate portfolios of past work and achievements [5]. These portfolios were

then presented to faculty and the general public. The work that goes into the portfolio accounted for 80% of students' overall grades [1].

The success of the new BIMS program was assessed through informal student interviews and a questionnaire. The informal interviews revealed that students enjoyed interacting and collaborating with other students [5]. An analysis of the questionnaires revealed that students preferred SBL over lecture, preferred working in groups, and recognized that SBL further developed their knowledgebase and skill set. Students had generally unfavorable marks for the portfolio, often indicating that they would prefer to have had more concrete direction for its construction.

The University of Victoria, Canada

The University of Victoria took a more localized approach when integrating SBL into computer science. Rather than undergoing a complete curriculum redesign, Estey et al. [6] decided to implement SBL in just a single course on game design. Again, the primary motivating factor for introducing SBL into the curriculum was to model industry practices in the classroom. Researchers were most interested in promoting communication and collaboration skills in particular.

Contrary to the University of Queensland and Monash University, lecture and SBL were integrated into a single course. Students gained new information through lecture, which was then reinforced with studio-based labs. Additionally, rather than plunging students into a full-fledged studio-based experience, Estey et al. decided to gradually introduce students to SBL practices. Introduction activities included writing a review for a video game and debating the merits of Batman and Superman. The concept of gradual introduction continued through the end of the first major programming project in which students witnessed experts giving constructive critiques on group presentations. The final, largest project fully integrated SBL into the learning process by making peer review an essential component to every stage of the project.

The success of the game design program was assessed through student questionnaires. As was the case with the other SBL programs discussed thus far, results were very positive. Students indicated that studio-based techniques improved their sense of community, developed motivation, and provided alternate ideas and views to consider. Based on these results, Estey at al. plan to implement SBL in other courses while performing additional analysis on its effectiveness.

Washington State University

Motivated that the observation that so-called "soft skills" (e.g., communication, teamwork, collaboration) are increasingly coveted by the software profession, researchers at Washington State University have focused extensively on the peer review aspect of SBL. Borrowing from standard code inspection techniques used in industry, Hundhausen et al. [7] explored the use of the pedagogical code review (PCR) in introductory programming courses. In a PCR, students are assigned to review teams, which are led by trained moderators (upper-division or graduate computer science students). Teams step through each team member's code one line at a time, checking it against a standard list of best coding practices.

In two separate studies, PCRs replaced three lab sessions in an introductory computer science course [7, 8]. To measure the effectiveness of PCRs, Hundhausen et al. investigated the effect that PCRs would have on a student's attitudes through a modified version of the Motivated Strategies for Learning Questionnaire (MSLQ) and test performance [8], as well as investigating the types of issues logged during a PCR [7]. Results indicated no significant differences in the test performance of students who participated in PCRs and students who did not. However, two notable attitudinal differences were measured. First, PCRs promoted an increase in attitudes toward peer learning that approached significance. Second, Whereas self efficacy (a measure of students' perceptions of their ability to program) decreased significantly among students who did not participate in the PCRs, no such decrease was measured in students who participated in PCRs.

Auburn University

Starting in 2007, Auburn University began integrating SBL into their CS2 curriculum [9, 10, 11]. Like the researchers at Washington State University, researchers at Auburn are motivated by the observation that present pedagogical methods in computer science fail to prepare students for jobs in the computing profession. Unlike other universities, studio-based activities at Auburn occur both in and outside of class. As at the University of Victoria, students present their work and critique the work of others within a laboratory setting. Additionally, students must spend time outside of class by visiting the course's website to review the work of other students. In these reviews, both the author and reviewer are anonymous.

Researchers at Auburn have used multiple measures to gauge the effectiveness of their SBL implementation. As at Washington State University, student attitudes were captured using the MSLQ. Also, the quality of online student reviews was analyzed for effectiveness. Finally, grade comparisons were made between the studio-based course and one that was taught without SBL methods.

Results from the MSLQ indicate that SBL promoted significant gains in student intrinsic motivation, extrinsic motivation, self-efficacy, peer learning, self-regulation, and sense of community [9]. In addition, Myneni et al. [11] found that review quality was quite poor at the beginning of the semester, but became more detailed as the semester progressed. Myneni et al. believe that such gains show an improvement in students' critical thinking skills. To compare the effectiveness between SBL and traditional instruction, Hendrix et al. [9] compared scores on assignments, tests, and a minimum skills test. They found a significant difference between assignment and test scores but no significant difference on the minimum skills test. These results indicate that SBL does a better overall job at instruction but both methods are sufficient at delivering the core content.

CONCLUSION

Having presented a brief overview of SBL in computer science, we now reconsider our original research questions.

What Are The Motivations For Implementing SBL In A Particular Course?

Most researchers used SBL as a means both to reinvigorate a curriculum and to give students exposure to industry practices. Only three of the universities surveyed (Queensland, Washington State, and Auburn) believed there to be an overall deficiency in computer science pedagogy. Researcher motivations did not appear to have an effect on the particular ways in which SBL was implemented at each university.

How Was SBL Actually Implemented In The Classroom?

For all curriculum-based SBL, student interaction plays a critical role. Usually, interaction was fostered through group presentations and/or peer review of assignments. Also noteworthy is that SBL never replaced lecture. Instead, SBL was used to reinforce lecture material. This seems to imply that researchers believe that SBL is appropriate for reinforcing ideas already learned but not for introducing new ones. However, we wonder whether it might be effective to reverse this by having lectures reinforce topics covered in the SBL labs. In this scenario, lectures would use the studio experiences as motivation for discussing a new topic or introducing new material. We believe this scenario warrants further investigation.

How Did Researchers Gauge The Success or Failure Of Their Particular SBL Implementation?

The success of a particular SBL implementation was usually assessed through end-of-term questionnaires completed by students. In general, students responded favorably to studio-based activities. At only two universities (Washington State and Auburn) did researchers compare their studio-based courses to the same courses taught only with lecture. Both of these found that the studio-based versions of the course produced measurable benefits, both in the form of higher homework and test scores (Auburn only), and student attitudes (Washington State and Auburn). These results are promising, but need to be followed up with additional comparisons at other universities before SBL can claim superiority.

What General Conclusions Did The Researchers Draw?

Our review indicates that students nearly universally enjoy SBL. Further, researchers and educators have a high opinion of SBL, and there seems to be a shared interested in further investigating SBL.

This review highlights two deficiencies in the current literature than can be expanded by future researchers. First, it is clear that the SBL space is in need of additional comparative studies. Comparisons between SBL and other pedagogies will allow us to better situate SBL in the context of computer science education. Investigating key aspects of the SBL process, similar to the work done by Hundhausen et al. [7], will provide us with insights on how to best maximize the benefits of SBL. Second, it is worth exploring the interplay between lecture and SBL. All of the cited studies appear to use SBL to reinforce lecture material. Yet, perhaps we would see larger improvements if roles were reversed and SBL were used to drive class lectures.

REFERENCES

[1] Carbone, A., Sheard, J., A studio-based teaching and learning model in IT: what do first year students think?, *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, 213-217, 2002.

[2] Schon, D. A., *The reflective practitioner*, Basic Books, Inc., 1983.

[3] Docherty, M., Sutton, P., Brereton, M., Kaplan, S., An innovative design and studio-based CS degree, *SIGCSE Bull.*, 33 (1), 233-237, 2001.

[4] Hundhausen, C. D., Brown, J. L., Designing, Visualizing, and Discussing Algorithms within a CS 1 Studio Experience: An Empirical Study, *Computers & Education*, 50, 301-326, 2008.

[5] Lynch, K., Carbone, A., Arnott, D., Jamieson, P., A studio-based approach to teaching information technology, *Proceedings of the Seventh world conference on computers in education: Australian topics - Volume 8*, 75-79, 2002.

[6] Estey, A., Long, J., Gooch, B., Gooch, A. A., Investigating studio-based learning in a course on game design, *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 64-71, 2010.

[7] Hundhausen, C. D., Agrawal, A., Fairbrother, D., Trevisan, M., Integrating pedagogical code reviews into a CS 1 Course: An Emperical Study, *Proceedings of the 40th ACM technical symposium on Computer science education*, 291-295, 2009.

[8] Hundhausen, C., Agrawal, A., Fairbrother, D., Trevisan, M., Does studio-based instruction work in CS 1?: an empirical comparison with a traditional approach, *Proceedings of the 41st ACM technical symposium on Computer science education*, 500-504, 2010.

[9] Hendrix, D., Myneni, L., Narayanan, H., Ross, M., Implementing studio-based learning in CS2, *Proceedings of the 41s ACM technical symposium on Computer science education*, 505-509, 2010.

[10] Hundhausen, C. D., Narayanan, N. H., Crosby, M. E., Exploring studio-based instructional models for computing education, *Proc. 39th SIGCSE Technical Symposium on Computer Science Education*, 392-396, 2008.

[11] Myneni, L., Ross, M., Hendrix, D., Narayanan, H., A Review of Studio-Based Learning in Computer Science, *Proceedings of the 46th Annual Southeast Regional Conference on XX*, 253-255, 2008.