Implementing Studio-Based Learning in CS2
Dean Hendrix, Lakshman Myneni, Hari Narayanan, and Margaret Ross*
Computer Science and Software Engineering
Educational Foundations, Leadership and Technology*
Auburn University
Auburn, AL 36849
{hendrd | mynenls | naraynh | rossma1} @auburn.edu

ABSTRACT
This paper presents an experience in designing, implementing, and evaluating a studio-based learning model for CS2. Adapted from architecture and art education, as well as from collaborative problem-solving pedagogies, studio-based learning has shown great promise for computing education. Key elements of studio-based learning include exploring multiple solutions to a problem, justifying the choice of one solution, and being subject to, as well as providing, peer reviews. We describe the design, implementation, and revision of a studio model for CS2, and then present the results of an evaluation of the model when compared to traditional instruction in CS2.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Computer science education, Curriculum.

General Terms
Design, Experimentation.

Keywords
Studio-based learning, CS2, computer science education research, peer review, design critiques, pedagogy.

1. INTRODUCTION
Studio-based learning is an instructional technique that emphasizes collaborative, design-oriented learning [10]. The pedagogy itself is not new; its roots extend as far back as Platonism and it is fully realized in modern architectural and industrial design education [8]. Proponents of studio-based learning argue that its learn-by-doing approach and its high degree of interaction, collaboration, and feedback offer many advantages to the student [2, 12].

While there is a notable example of a full-scale implementation of studio-based learning in a computing curriculum [5], adoption of the pedagogy by the computing education community has not been common [10]. Only recently has the literature evidenced a renewed interest in studio-based learning for computing programs. In the face of declining enrollments, the computing community has gone through a healthy self-examination and concluded that it is critically important to increase student enjoyment in problem solving and raise student motivation levels and their interest in computer science. In short, new approaches to computing education are needed; approaches that engage and excite students yet at the same time effectively facilitate learning. Prior work has shown that studio-based learning can be effective in meeting these goals [9, 10, 13].

While the studio-based learning paradigm is somewhat standardized and uniform in classic design fields such as architecture, art, and industrial design, its implementation in computing education can vary widely. In particular, design activities adapted to early courses in computing tend to be small-scale software design and programming, which are narrower in design scope than the corresponding activities found in domains such as architecture. However, recent research (e.g., [1]) suggests that the design perspective and the studio-based instructional approach are still valuable to consider for early computing courses such as CS2.

Previous work has identified two learning activities as being fundamental to studio-based learning [10]: (1) Students construct their own artifacts (e.g., design, source code, visual description, etc.) as appropriate to the problem at hand. (2) Students present their artifacts for review in sessions commonly called design critiques or design crits. It is this second activity in particular that differentiates studio-based learning from other pedagogical approaches. Indeed, the design critique is considered the centerpiece of the model and forms the basis of a particular way of thinking and learning [12].

We can expand these two core principles into specific course activities as follows [13]: (a) Students are given non-trivial problems for which they have to design and implement computational solutions individually or in groups. (b) These problems are amenable to multiple solution strategies. Thus, students have to consider alternate solutions and their tradeoffs in terms of efficiency and software engineering considerations, choose the best, and justify their choice. (c) Students must articulate their solutions and justifications to other students for peer review, feedback, and discussion. This can be done orally, in writing, or both. (d) Their peers and the course instructor must provide comments and criticisms, again in writing, orally, or both. (e) Students are given the opportunity to respond to this feedback and modify their solutions appropriately.
it is the expectation that by reflecting on and learning from these design exercises over the course of a semester, students will become more proficient in computational problem solving and more engaged in the learning process.

This paper reports on our experience in adapting studio-based learning to the CS2 (data structures) course at Auburn University. We completed the course design and adaptation over a two-semester period (fall 2007, spring 2008), with the first semester being a pilot and the second being a full studio implementation. We used a formative evaluation based on both formal and informal data to guide our decisions and shape the ultimate form of the studios [19]. At the end of this two-semester period, we had developed a robust studio-based learning model for CS2.

To measure the efficacy of our SBL model in CS2, we used the subsequent two semesters (fall 2008, spring 2009) to compare our studio-based learning model to a traditional, non-studio offering of the course. The CS2 course was offered in studio format during fall 2008 and in a traditional, non-studio format in spring 2009. Data was collected that allowed us to compare the difference in both student learning and student motivation and attitudes between studio-based instruction and traditional instruction.

We have organized the paper as follows. Section 2 provides a brief overview of related work, while Section 3 describes our particular approach in detail. Section 3 also provides a discussion of our two-semester implementation and revision of the studio model, including the results from our initial formative evaluation. Section 4 discusses the initial results from our experimental comparison of a studio-based offering of CS2 to a traditional offering of the same course, and draws conclusions from the resulting data analysis. Section 5 concludes the paper with a brief summary of our work and results.

2. RELATED WORK

One of the most extensive implementations of studio-based learning in computing education is the Bachelor of Information Environments degree at the University of Queensland. This program combines core computer science content with courses on design and immersive design studios fashioned after the architectural studio model [5]. The Bachelor of Information Management and Systems (BIMS) degree at Monash University is another example of an extensive implementation of the studio model in computing education. The BIMS degree uses an architecture-like studio model that incorporates not only a studio-based approach to teaching, but also a studio-centric physical layout of facilities and lab space [3].

Other reports of studio-based learning activities in the computing education literature tend to be focused on a small number of higher-level courses like software engineering [6] and human-computer interaction [14], or they tend to be focused on a particular learning style such as experiential learning [7] or active learning [1, 11]. We are unaware of any other work that articulates and evaluates the particular model of studio-based learning that we have implemented.

3. IMPLEMENTING SBL IN CS2

The CS2 course at Auburn University is typical in content, focusing on data structures and associated algorithms from an object-oriented perspective. The instructional language is Java and the instructional IDE is jGRASP (jgrasp.org). CS2 is a 4 credit hour course and meets for 2.5 clock hours of lecture per week and 2.5 clock hours of lab per week. All students meet together for the same lecture, but they meet separately in small, 75-minute lab sessions twice weekly. Each lab session has no more than 23 students, and the total course enrollment is typically between 60 and 70 students per semester.

3.1 Pilot Semester

We began the implementation of a studio-based learning model for CS2 in the Fall 2007 semester. This was treated as a pilot semester to give us a chance to try things out and learn from our mistakes. Data was collected for formal analysis, but we made many adjustments as the semester went along based on an intuitive sense of what seemed to be working and what seemed to need improvement.

There were five programming assignments during the pilot semester, and we used a studio-based learning model for the last four. Each of those assignments was approximately 2 to 2.5 weeks in duration. The assignments were at a level of complexity similar to many of the Nifty Assignments traditionally presented at SIGCSE conferences (http://nifty.stanford.edu). An assignment presented a problem with multiple possible solution strategies, and students were required to develop both a “design” and an implementation. The “design” consisted of a high-level description of at least two possible solutions, a visualization of each solution being applied to a sample instance of the problem, and a justification of the choice that the student made in choosing a strategy to implement. The implementation consisted of the Java source code files that expressed their chosen strategy. The design and implementation were due at the same time.

After the assignment deadline had passed, the course TAs uploaded all the submissions to a website and assigned each student up to five other students’ work to review. This peer review required the students to read and evaluate each other’s work and make meaningful comments regarding its correctness, efficiency, creativity, etc. Students were assigned “posting codes” at the beginning of the semester and were instructed to use these codes in place of their names in identifying their work. Thus, the peer reviews were performed anonymously.

Students were given three days to perform their assigned peer reviews using an online review system. During the lab session immediately following the peer review deadline, students were allowed to orally respond to the reviews that their work had received. All students who discussed their reviews in lab were given three extra points on the assignment.

The pilot semester went generally well, and it allowed us to make small adjustments during the semester as well as plan larger changes for the next semester. The major adjustment that we made several times through the pilot semester was changing the point distribution for the various components of a lab assignment. We were accustomed to grading only source code and were hesitant to award substantial points to any other component. It became apparent, however, that unless there were significant point values associated the design, peer reviews, and oral response, students wouldn’t put as much effort into those components as we would like.

Once the pilot semester ended we held a “lessons learned” planning session for the next semester. The overall sense was
definitely positive, and we were convinced that the studio-based pedagogy held much promise for CS2 (see [13]).

We identified four major changes to implement during the next semester. First, students had performed the peer reviews while they were working on the next assignment. Although very few students actually complained about this, we felt that this overlap was a distraction for students and that it would be better if the peer reviews occurred after one assignment is submitted and before the next is assigned. We also decreased the number of reviews that each student had to perform to three (instead of five) to allow more effort to be expended on each one. Second, the post-mortem nature of the peer reviews didn’t give students a real chance for the give-and-take that we wanted to see during the labs. Students were reviewing and commenting on things that were already completed and couldn’t be changed, and the authors had little opportunity to present their work and respond to questions. We felt that there would be value in introducing a charrette component [4] to the labs, where students would present their works-in-progress to classmates and receive feedback that could be incorporated into their solution. Third, due to enrollment numbers and the length of the lab sessions, we planned to have students work in pairs on assignments in order to accommodate the charrette. Finally, we decided that the assignments should include more architectural issues rather than focusing more closely on the algorithmic level. Thus, designing appropriate interfaces, classes, and their associations was given more weight in the assignments.

3.2 Full Studio Semester

Spring 2008 was the first “full studio” semester, having incorporated the lessons learned from the pilot offering of CS2. By this point, we had distilled the following five student activities as being fundamental to our implementation of the studio-based learning model: (1) Develop multiple solution strategies. (2) Justify the choice of one solution strategy. (3) Present work-in-progress to peers and respond to comments. (4) Submit final work to peers for review. (5) Review the work of others.

All five lab assignments were studio-based. Each assignment was completed in teams of two, chosen by the instructor on a per-assignment basis. Six lab sessions were allocated to each assignment. The first two labs were work sessions during which team members worked together to complete their design and begin their solution in source code. The next two lab sessions were “studio labs” during which each team presented their design to the class and received feedback from other students (the charrette). The final two lab sessions were additional work sessions in which the teams could finalize their designs and implementations, possibly based on feedback from the studios. The assignment was due at the end of the sixth lab day.

Requirements for the design submission were kept the same. The design consisted of a high-level description of at least two possible solutions to the problem, a visualization of each solution being applied to a sample instance of the problem, and a justification of the choice that the student made in selecting a strategy to implement. However, the specific content of students’ designs changed slightly since there were more architectural issues to address than before. The design was due the night before the first studio lab, and each team presented their design during one of the studio labs for that assignment. The studio labs were designed to encourage input and a free exchange of ideas among the students. Teams were informed that they could change any element of their proposed solution based on the feedback they received in the studio. Figures 1 and 2 show excerpts from sample design presentations done during studios.

Assignment 2 Studio
Team 17
Comp 2210

Figure 1. Excerpt from a studio presentation, architectural level.

WSS Overview

Figure 2. Excerpt from a studio presentation, algorithmic level.

The implementation consisted of the Java source code files that expressed their chosen strategy and it was due one week after the studio labs. Students had to turn in their design (possibly revised) along with their implementation. After the assignment deadline had passed, the submissions were uploaded to the online review system. Students had approximately four days to complete the review of up to three other students’ work. The next assignment was posted after the review deadline had passed.

3.3 Formative Evaluation

The formative evaluation of our studio-based learning model for CS2 addressed student performance, attitudes, and motivation. A pre/post content test along with graded items from the course were used to measure student performance, and the Motivated Strategies for Learning Questionnaire (MSLQ) from the National Center for Research to Improve Postsecondary Teaching and Learning was used to assess student attitudes and learning motivation.

We developed a pre/post content test that covers most of the major “core” topics in the CS2 course. The pre-test is designed to be a minimalist content test for a typical CS2, and it was developed in concert with two other universities [13]. The pre-
test was administered to the students during the first week of class during both semesters. The post-test, which contained the same questions as the pre-test, was administered as a subset of the comprehensive final exam for the course during each semester. The post-test questions were not labeled, separated from other questions, or identified in any way to the student. From a student’s perspective, there was no “post-test” per se for the course, only a final exam.

In the pilot semester (n = 49), the average pre-test score was 27.46 and the average post-test score was 55.06, for a 27.60-point improvement from pre-test to post-test. In the full studio semester (n = 55), the average pre-test score was 16.65 and the average post-test score was 57.00, for a 40.35-point improvement from pre-test to post-test. Certainly these results tell us that the students increased their mastery of CS2 content in both course offerings (as expected). But the results also suggest the possibility that the changes we made from the pilot semester to the full studio semester were effective. Indeed, the “learning curve” difference between the pilot semester and the full studio semester suggests that the full SBL model offered learning advantages to the students.

The difference between average pre-test scores for the two course offerings suggests that students in the pilot semester were, as a whole, at a significantly higher achievement level coming into the course than the students in the full studio offering. The difference between the two groups’ post-test scores, however, was not statistically significant. That is, by the end of each course offering both groups were at essentially the same achievement level. This suggests the possibility that the full studio approach offered definite, measurable improvements over our initial pilot implementation.

A comparison of the lab assignment averages from the pilot semester and the full studio semester reveals another improvement. The pilot semester lab assignment average was 68.6 while the full studio semester lab assignment average was 89.1, which is a statistically significant difference of 20.5 points (p < 0.001). This indicated that the full studio approach offered a greater net gain in student performance.

Comparing the improvement in pre-test to post-test scores for both semesters tells us that student learning increased in both the studio and traditional offerings of the course (as expected). The mean pre-test score from the studio offering was higher by 4.29 points than the mean pre-test score from the traditional offering. The mean post-test score from the studio offering was similarly higher by 3.83 points than the mean post-test score from the traditional offering. There is not a statistically significant difference between the two post-test means. So we can conclude that both groups ended the course at essentially the same level of mastery of the content covered by the pre/post-test questions.

The particular performance measures that we used as the basis of our evaluation were student scores on the pre-test and post-test, and their average scores on the exams and programming assignments. This data is summarized in Figure 3.

Compared to the studio offering, the traditional offering did not demonstrate statistically significant differences in student performance. Therefore, we conducted comparisons of student performance in programming labs and exams of the course, and these are described below. Furthermore, though there was a statistically significant difference between the pre-test means of the studio and traditional offerings, we do not believe that this is indicative of the students of the studio offering entering the course with more prior knowledge. We gave students explicit instruction about not guessing the answers to the pre-test questions in the Spring 2009 semester (the traditional offering) after noticing the possibility that this might have been happening before. Such explicit instruction was not given for the pre-tests in prior semesters including the Spring 2008 semester (the first studio offering). We subsequently found that pre-test means from the three semesters prior to Spring 2009 were higher than the Spring 2009 pre-test mean. Therefore, we should be cautious about
drawing conclusions about prior knowledge based on pre-test comparisons.

![Graph showing student performance measures](image)

**Figure 3. Summary of student performance measures.**

When we examined the graded items from the course (exams and programming assignments), we saw significantly higher performance from students who took the studio offering. The average exam score in the studio semester was 74.99, while the average exam score in the traditional semester was only 65.74. This difference of 9.25 percentage points is statistically significant (p < 0.001) and suggests that students in the studio offering were able to perform at a higher level on exams than the students in the traditional offering of the course. Students in the studio offering also exhibited significantly better performance in programming tasks as well. The average programming assignment score in the studio semester was 79.64, compared to 70.06 in the traditional semester. This difference of 9.58 percentage points is again statistically significant (p < 0.01) and suggests that students in the studio offering were able to develop better programs than the students in the traditional offering.

5. SUMMARY

We have adapted a studio-based learning model for a typical CS2 course and evaluated its effectiveness. In our model of studio-based learning, students get experience in: (1) individually and collaboratively solving computational problems, (2) evaluating and selecting among competing solutions based on engineering design issues, (3) explaining their solutions to others in writing and through oral presentations, (4) critically analyzing each other’s solutions in peer reviews, and (5) reflecting on and learning from these studio sessions over the course of a semester. A formal evaluation in which the studio-based learning model was compared to a traditional offering of a CS2 course revealed that the studio-based learning model offered significant benefits to students in terms of both course content mastery and programming achievement. We are quite optimistic that the studio-based learning model will prove to be an effective approach to reinvigorating computing education.

6. ACKNOWLEDGMENTS

This paper is based on work done at Auburn University as part of a multi-university research project in which researchers from Auburn, the University of Hawaii, and Washington State University (A. Agrawal, M. Crosby, D. Hendrix, C. Hundhausen, S. Myneni, H. Narayanan, M. Ross, M. Trevisan, and R. Vick) are participating. This work is supported by the National Science Foundation under CPATH Grant No. CNS-0721927.

7. REFERENCES


