

## **Adapting a Studio-Based Learning Model for CS2**

**Dean Hendrix  
Lakshman Myneni  
N. Hari Narayanan  
Margaret Ross**

Technical Report CSSE08-03

Department of Computer Science & Software Engineering  
Auburn University  
Auburn, Alabama 36849-5347, USA

5 pages  
October 29, 2008

# Adapting a Studio-Based Learning Model for 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

hendrtd | 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 processes such as urban planning, 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 our two-phase approach to implementing studio-based learning in a traditional CS2 course, as well as our initial evaluation results.

## Categories and Subject Descriptors

K.3.2 [Computer and Information Science Education]:  
*Computer science education.*

## General Terms

Design, Experimentation.

## Keywords

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

## 1. INTRODUCTION

Studio-based learning is an instructional technique that emphasizes collaborative, design-oriented learning [11]. 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, 15].

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 [11]. 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 [11, 16].

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. However, two learning activities have been identified as being fundamental to studio-based learning [11]:

1. Representation Construction – Students construct their own artifacts (e.g., design, source code, visual description, etc.) as appropriate to the problem at hand.
2. Representation Presentation – 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 [15].

We can expand these two core principles into specific course activities as follows [16]: (a) Students are given meaningful 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.

## 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 a 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 [18], or they tend to be focused on a particular learning style such as experiential learning [7] or active learning [14, 20]. Other examples from the literature include implementations that are directed toward the development of a particular technical skill or particular attitude that affects learning [9, 19].

We are unaware of any other work that articulates and evaluates the particular model of studio-based learning that we have implemented. The approach taken in [10] is the most closely related to our own, but focuses on student construction of algorithm animations to support learning in CS1.

## 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 [12]. The course is required for several different engineering majors and enrollment is typically between 60 and 70 students per semester. 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. The instructor teaches the lecture section while graduate teaching assistants administer the labs. Students generally consider CS2 to be difficult, particularly in relation to their CS1 experience.

### 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 experiment, try things out, and learn from our mistakes. Data was collected for formal analysis (see Section 4), 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 similar level of

complexity as many of the Nifty Assignments (<http://nifty.stanford.edu>) presented at previous SIGCSE conferences. 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 was a very valuable exercise. 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 Section 4 and [16]). We were pleased with the way that the studio approach forced students to think critically about multiple ways to solve a problem and weigh the tradeoffs of each. We were also pleased with the learning effect that we could see from the peer reviews.

We identified two 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. 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, 13] to

the labs, where students would present their works-in-progress to classmates and receive feedback that could be incorporated into their solution. 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.

### 3.2 Full Studio Semester

Spring 2008 was the “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 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. The assignment was due at the end of final lab day (11PM).

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. The design was due at 11PM 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.

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. The next assignment was posted after the review deadline had passed.

We again concluded with a lessons-learned meeting at the end of the full studio semester. We had been able to implement all the “core SBL” elements, and the informal and anecdotal feedback we received throughout the semester was very positive. While not attributes of the studio-based learning model itself, the programming teams raised their own issues that bear a closer look in the future. Also, the lock-step nature of the lab schedule leaves no room for getting behind. Overall, however, our sense of success was very good.

## 4. EVALUATION

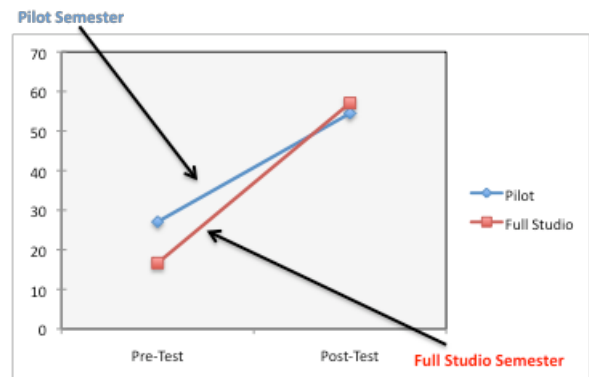
The evaluation of our studio-based learning model for CS2 has focused on assessments of student performance, attitudes, and motivation. A pre/post content test was used to measure student performance, and the Motivated Strategies for Learning Questionnaire (MSLQ, [17]) was used to assess student attitudes and learning motivation.

### 4.1 Performance

We developed a pre/post content test that covers most of the major topics in the CS2 course. There are 40 questions on the pre-post content test, and a total of 80 points possible. The pre-test was administered to the students during the first week of class during both semesters. Students were instructed to answer only questions that they felt relatively sure of, rather than guessing in hopes of partial credit. While the pre-test was a required element of course participation, it was made clear to the students that their performance on the pre-test would in no way affect their course grade.

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 = 43), the average pre-test score was 27.07 and the average post-test score was 54.47, for a 27.40-point improvement from pre-test to post-test. In the full studio semester (n = 54), the average pre-test score was 16.65 and the average post-test score was 57.07, for a 40.42-point improvement from pre-test to post-test. These results tell us that the students increased their mastery of CS2 content in both course offerings (as expected), but the results also suggest 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 (see Figure 1) suggests that the full SBL model offered learning advantages to the students.



	Pilot (Means)	Full Studio (Means)
Pre-test	27.07	16.65
Post-test	54.47	57.07

Figure 1. Test scores from pilot and full studio semesters

The difference between average pre-test scores for the two course offerings indicates 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. By the end of each course offering both groups were at essentially the same achievement level. Thus, the full SBL model facilitated a greater net gain in student performance.

## 4.2 Attitudes and Motivation

### 4.2.1 MSLQ

The Motivated Strategies for Learning Questionnaire (MSLQ) was used to assess student attitudes and learning motivation. The MSLQ consists of 81 items and has 6 motivation subscales and 9 learning strategy scales. Students respond to each item with a value on a 7 point Likert-type scale. A response of 1 indicates that the item is "not true at all of me" while a response of 7 indicates that the item is "very true of me." Students completed the MSLQ during the first week of class as a pre-survey and again during the last week of class as a post-survey. Students were informed that there was no score associated with the survey, but completing it was a required participation component of the course.

To assess student attitudes and motivation we focused on the following MSLQ scales: intrinsic motivation, extrinsic motivation, self efficacy, peer learning, critical thinking, self regulation, and sense of community. Results for both the pilot semester and the full studio semester are summarized in Figure 2. An up-arrow symbol (↑) beside an entry indicates a statistically significant increase.

For the pilot semester, the mean scores in all scales increased from pre-survey to post-survey, with statistically significant gains in intrinsic motivation, self efficacy, critical thinking, and self regulation. The full studio semester saw increased means in all 7 scales with statistically significant gains in all scales except critical thinking. When comparing the post-survey means of the two course offerings we see statistically significant gains in extrinsic motivation, self efficacy, and self regulation from the pilot semester to the full studio semester.

### 4.2.2 Interviews and Informal Responses

The project evaluator interviewed five students from each semester. Her questions were open-ended, addressing students' perceptions of learning, interest and motivation in the course, and sense of community with classmates. Students from the pilot semester reported that they generally enjoyed the course and learned from it [16]. Many viewed the labs as particularly useful, and were pleasantly surprised at the level of creativity demanded by the lab projects. They reported increased motivation, due in large part to the lab, as exemplified by the statement made by a student indicating s/he liked labs specifically because they were applied and were like solving a puzzle.

The students from the full studio course also generally reported positive attitudes toward collaborative problem solving. Consistent with views from the pilot semester, these students viewed the lab activities as having a particularly positive impact on learning. They stated that peer written feedback as well as oral feedback during presentations helped them identify deficiencies in

their solutions and consider alternative approaches. Though they expressed some reservations, such as unequal work by some group members and uninteresting presentations by some groups, their overall reactions were quite positive, as can be seen from these interview quotes: "I find, you know, the processes that we went through...which algorithm we use, even coming up with the algorithm in the first place, the whole problem solving methods that we did,...very engaging... another sign that I am in the right major." "Peer review, I guess, gives people a chance to review your work to see how someone else might have solved the same problem...same with the design phase I guess, you could say...gave people a chance to stand up and face the rest of the class and present their work...maybe even give other people ideas...[who] might be sitting out there lost on where to go."

Pilot Semester (n = 40)			
Scale	Pre-Survey Means	Post-Survey Means	
Intrinsic Motivation	5.00	6.01	↑
Extrinsic Motivation	5.03	5.16	
Self Efficacy	4.67	5.46	↑
Peer Learning	3.65	3.70	
Critical Thinking	4.32	4.85	↑
Self Regulation	4.28	4.60	↑
Sense of Community	2.66	2.72	

Full Studio Semester (n = 45)			
Scale	Pre-Survey Means	Post-Survey Means	
Intrinsic Motivation	5.06	5.57	↑
Extrinsic Motivation	5.16	5.54	↑
Self Efficacy	5.11	5.92	↑
Peer Learning	3.06	3.89	↑
Critical Thinking	4.11	4.16	
Self Regulation	4.47	5.26	↑
Sense of Community	2.05	2.82	↑

Figure 2. MSLQ means from pilot and full studio semesters

## 5. SUMMARY AND FUTURE WORK

We have successfully adapted a studio-based learning model for a typical CS2 course. 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. Acquiring these skills in CS2 should not only prepare students for greater success later in the computing curriculum, but should also help mitigate some of the struggles that new college graduates report having on their first job [1].

An initial evaluation of our studio-based learning model shows that students are learning and are improving in key measures of attitudes and learning motivation. An important evaluation that remains to be done is a comparison between a full studio offering and a traditional (non-studio) offering of CS2. We have collected data from a summer offering of CS2 that was taught in a traditional format, but due a large difference in sample size (54 v. 10), we have not performed a statistical analysis. Spring 2009 will be the next traditional offering of CS2 and it will have an enrollment large enough to facilitate a meaningful comparison. Even so, based on the existing evaluation and on anecdotal evidence and informal feedback from the summer traditional offering of CS2, 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. Any opinions, findings, and conclusions expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## 7. REFERENCES

- [1] Begel, A., and Simon, B. 2008. Struggles of new college graduates in their first software development job. *Proceedings of 39<sup>th</sup> Technical Symposium on Computer Science Education (SIGCSE 2008)*, March 12-15, 2008, Portland, Oregon, pp. 226-230.
- [2] Boyer, E. L., and Mitgang, L. D. 1996. *Building Community: A New Future for Architecture Education and Practice*. Princeton, NJ: The Carnegie Foundation for the Advancement of Teaching.
- [3] Carbone, A., and Sheard, J. 2002. A studio-based teaching and learning model in IT: what do first year students think? *Proceedings of the 7<sup>th</sup> Annual Conference on Innovation and Technology in Computer Science Education*, Aarhus, Denmark, 2002, pp. 213-217.
- [4] Denning, P. J., and Yaholkovsky, P. 2008. Getting to “we.” *Communications of the ACM*, Vol. 51, No. 4, April 2008, pp. 19-24.
- [5] Docherty, M., Sutton, P., Brereton, M., and Kaplan, S. 2001. An innovative design and studio-based CS degree. *ACM SIGCSE Bulletin*, Vol. 33, Issue 1, March 2001, pp. 233-237.
- [6] Gannod, G. C., Burge, J. E., and Helmick, M. T. 2008. Using the inverted classroom to teach software engineering. *Proceedings of the 30<sup>th</sup> International Conference on Software Engineering*, May 10-18, 2008, Leipzig, Germany, pp. 777-786.
- [7] Gonsalvez, C., and Atchison, M. 2000. Implementing studios for experiential learning. *Proceedings of the Australasian Conference on Computing Education (ACE 2000)*, Melbourne, Australia, pp. 116-123.
- [8] Green, L. N., and Bonollo, E. 2003. Studio-based teaching: history and advantages in the teaching of design. *World Transactions on Engineering and Technology Education*, Vol. 2, No. 2, pp. 269-272.
- [9] Howles, T. 2005. Community and accountability in a first year programming sequence. *inroads – The SIGCSE Bulletin*, Vol. 37, No. 2, June 2005, pp. 99-102.
- [10] Hundhausen, C. D., and Brown, J. L. 2004. Personalizing and discussing algorithms within CS1 studio experiences: an observational study. *Proceedings of the 2005 International Workshop on Computing Education Research (ICER 2005)*, October 1-2, 2005, Seattle, WA, pp. 45-56.
- [11] Hundhausen, C. D., Narayanan, N. H., and Crosby, M. E. 2008. *Proceedings of 39<sup>th</sup> Technical Symposium on Computer Science Education (SIGCSE 2008)*, March 12-15, 2008, Portland, Oregon, pp. 392-396.
- [12] jGRASP: An Integrated Development Environment with Visualizations for Improving Software Comprehensibility. <http://www.jgrasp.org>
- [13] Lennertz, B. 2003. *The charrette as an agent for change. New Urbanism: Comprehensive Report and Best Practices Guide*, 3<sup>rd</sup> Edition. Ithaca: New Urban Publications, 2003, pp. 12-28.
- [14] Ludi, S., Natarajan, S., and Reichlmayr, T. 2005. An introductory software engineering course that facilitates active learning. *Proceedings of the 36<sup>th</sup> Technical Symposium on Computer Science Education (SIGCSE 2005)*, February 23-27, 2005, St. Louis, MO, pp. 302-306.
- [15] Maitland, B. M. 1991. Problem-based learning for an architecture degree. In: *The Challenge of Problem-based Learning*, Boud and Feletti (Eds.).
- [16] Myneni, L., Ross, M., Hendrix, D., and Narayanan N. 2008. Studio-based learning in CS2: An Experience Report. *Proceedings of the 46<sup>th</sup> ACM Southeast Conference (ACM-SE 2008)*, March 28-29, 2008, Auburn, AL, pp. 253-255.
- [17] Pintrich, P.R., Smith, D. A. F., Garcia, T., and McKeachie, W. J. 1991. *A manual for the use of the Motivated Strategies for Learning Questionnaire (MSLQ)*. National Center for Research to Improve Postsecondary Teaching and Learning.
- [18] Ungar, J. M., and White, J. A. 2008. Agile user-centered design: enter the design studio – a case study. *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2008)*, April 5-10, 2008, Florence, Italy, pp. 2167-2177.
- [19] Woodley, M., and Kamin, S. N. 2007. Programming studio: a course for improving programming skills in undergraduates. *Proceedings of the 38<sup>th</sup> Technical Symposium on Computer Science Education (SIGCSE 2007)*, March 7-10, 2007, Covington, KY, pp. 531-535.
- [20] Wulf, T. 2005. Constructivist approaches for teaching computer programming. *Proceedings of the 6<sup>th</sup> Conference on Information Technology Education*, Newark, NJ, 2005, pp. 245-248.