Studio-Based Learning in CS2: An Experience Report

Lakshman Myneni
CSSE Dept.
3101 Shelby Center
Auburn University
Auburn, AL 36849
+1 334-844-6352
mynenls@auburn.edu

Margaret Ross
EFLT Dept.
4018 Haley Center
Auburn University
Auburn, AL 36849
+1 334-844-3084
rossma1@auburn.edu

Dean Hendrix
CSSE Dept.
3101 Shelby Center
Auburn University
Auburn, AL 36849
+1 334-844-6352
hendrtd@auburn.edu

N. Hari Narayanan
CSSE Dept.
3101 Shelby Center
Auburn University
Auburn, AL 36849
+1 334-844-6352
naraynh@auburn.edu

ABSTRACT
Recently there has been a surge of interest in making computer science education attractive to potential students, motivating to current students, and relevant to graduating students. We are exploring a new pedagogical approach called studio-based learning as a means to reinvigorate computer science education. Adapted from architectural education, this instructional model emphasizes learning activities in which students (a) design computational solutions to problems that lend themselves to multiple solution strategies, and (b) present and justify their solutions to their instructors and peers for critical review and discussion. In this paper we describe the studio-based approach, discuss how it was implemented in CS2, and present preliminary evaluation results.

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

General Terms
Design, Experimentation, Human Factors.

Keywords
Computer science education research, CS2, peer review, studio-based teaching and learning.

1. INTRODUCTION
The studio-based instructional model as practiced by architectural schools is in the form of the "design studio," a place where students set up their own workspaces, and create and present their designs [6]. As students work on design tasks in this common space, they develop a "community of practice," providing support and feedback to each other. The design studio curriculum involves a series of design problems, which may either be a sequence of progressively more challenging design problems, or various components of a large design project. A key aspect of the design studio is the design critique. Design critiques are review sessions in which students present their evolving solutions to the instructor and the class for feedback and discussion. Boyer and Mitgang [1] state in their comprehensive review of architecture education that "the core elements of architectural education—learning to design within constraints, collaborative learning, and the refining of knowledge through the reflective act of design—have relevance and power far beyond the training of future architects."

Researchers from Auburn University, University of Hawaii and Washington State University have embarked on a research project to adapt and apply the studio-based learning (SBL) approach to computing education [2]. As part of this NSF-supported effort, we implemented the studio-based instructional model in a CS2 course at Auburn University that is taken by a variety of undergraduate majors: computer science, software engineering, computer engineering and wireless engineering. Key aspects of the SBL model are:

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. This means that 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. They must then articulate their solutions and justifications to the entire class for peer review, feedback and discussion, in writing, orally or both.

d. Their peers and the course instructor evaluate these and 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.

Through these steps students get experience in: (1) individually and collaboratively solving algorithm and software design problems, (2) evaluating and selecting among alternate designs based on considerations of correctness, efficiency and other engineering design issues, (3) explaining their solutions to others in writing and through oral presentations and argumentations, (4) critically analyzing each others' solutions in peer reviews, and (5) reflecting on and learning from these design exercises over the course of a semester, thus becoming more proficient practitioners of computational problem solving.

To appear in the Proceedings of the 46th ACM Southeast Conference, Auburn, AL, USA, March 28-29, 2008. ACM Press.

2. IMPLEMENTATION OF SBL IN CS2

Fundamentals of Computing II (COMP 2210) is the second course in a series of three that computer science, software engineering, computer engineering and wireless engineering majors take at Auburn University. This course corresponds to the course referred to as CS2 in computing education literature. Students learn about data structures such as arrays, lists, trees, hash tables, etc., and algorithms that access, manipulate and solve problems with these data structures. The course has a laboratory component. Students meet twice a week in 75-minute long lab sessions, in which they work on their individual programming assignments with the help of teaching assistants.

Traditionally, the instructor would assign programming problems in class, students would work on them outside class and in the lab sessions and submit their solutions, which were graded for correctness and efficiency by teaching assistants. In fall 2007, this approach was changed to the SBL model in which five out of six assignments were designed to have the features we described above. The assignments remained individual assignments. However, each was presented as a problem (e.g., develop a game playing program for the common word game Boggle) that could be solved with multiple computational strategies. Students were asked to first think about various strategies, choose one, and explain the strategy they chose and justify it in writing, using verbal explanations and pictures. Their submissions of strategy explanations, visualizations and justifications were made anonymous and provided to the entire class for viewing on the web. In addition, each student was assigned up to four submissions of others for critical review, and asked to submit the reviews in writing through the web. Following this, the students implemented their strategy in Java and submitted their code for grading. Finally, each student was given the option of orally responding to criticisms of his/her approach in a lab session.

The next section presents our observations and findings about the impacts that this change in CS2 had on students.

3. PRELIMINARY FINDINGS

3.1 Performance

Students were required to provide critiques of others' project assignments, and were observed discussing their projects and responding to critiques during lab sessions throughout the semester. Observation protocols were developed. The initial protocol focused on appropriateness and communicative quality of responses to positive and negative comments and discussion. We used a scale of 1 = not appropriate, 2 = appropriate-low quality, 3 = appropriate-some depth, and 4 = in-depth analysis, self-assessment, and synthesis. Initial discussions tended to be brief and perfunctory, focusing on rote descriptions of code or strategy with little detail provided. Also, students put little thought into critiques of their work initially, mainly indicating they either agreed or disagreed with the comments or indicating that the critiques did not provide useful feedback. Because the scale descriptors were deemed to be too undefined or broad, the protocol was revised. Rather than merely rating student responses, we indicated whether or not students summarized strategy, acknowledged positive comments, and read critical comments. Additionally, we placed more emphasis on explanations related to why responses were (or were not) meaningful by including written comments about the responses.

As the semester progressed, students began to provide more detailed descriptions of their projects that included why they chose specific strategies. For example, during the second studio session of the semester, one student indicated that he was not really sure his work was reviewed because the critique didn't provide information related to the strategy used. By the end of the semester, students responded to critiques in a thoughtful manner, indicating why they agreed or disagreed with it and, at times, suggesting ways they might have improved their work. One student's comment during the last observation included an explanation of what was said in the critique. He additionally indicated that he felt like the student providing the critique understood the work. The progression from perfunctory to thoughtful discussion suggests improved performance in two ways. First, we can posit that written critiques improved throughout the semester indicating that students were more adept at thinking through code and strategy at a deeper level than they were at the start of the semester. Second, students demonstrated an improved ability to critically think about their own work based on feedback from others.

However, several issues still remained by the end of the semester:

1. The students had difficulty understanding some of the open-ended critique questions. For example, many students indicated that they were unclear about the meaning of the question "How original was the problem solving strategy?" Critique questions are now being developed to provide more clarity in relation to what is expected.
2. Accurately and fully recording responses to critiques was difficult during observations. Because of this difficulty, we decided to videotape critiques and responses during lab time in future. Videotapes can be reviewed at a later date, providing the opportunity to stop, review, and discuss parts, making data collection more accurate and complete.
3. Students expressed confusion in relation to exactly what was expected in their responses to critiques. Therefore, the critique and response process will be modeled for the students by us in future, thereby clearing up confusion about the process and, hopefully, forming a foundation for higher quality responses. Also, students will be given more lab time to critique and respond to comments in critiques, which we expect will also facilitate higher quality responses.

3.2 Attitudes and Motivation

The Motivated Strategies for Learning Questionnaire (MSLQ, [4]) was used to assess learning motivation. Items on the MSLQ are scored using a 7 point Likert-type scale with 1 = not at all true of me and 7 = very true of me. Validity was addressed using factor analytic procedures and reliability (alpha) coefficients are reported for the instrument's scales. The MSLQ manual reports the results of a structural model with path coefficients for the Intrinsic Motivation and Extrinsic Motivation scales ranging from .44 to .71 and alpha coefficients of .74 and .62.

A repeated measures analysis of variance (ANOVA) was used to assess the difference in intrinsic motivation level from pre-survey to post-survey (n = 40). Intrinsic motivation items (n=4) were

averaged to form the intrinsic motivation scale. Results were statistically significant, Wilks' Lambda $F(1,39) = 52.66$, $p < .001$, with a large effect size, partial $\eta^2 = .575$. The pre-survey mean was 5.0 (SD = .82) and the post-survey mean was 6.1 (SD = .42). These results suggest that student motivation to learn in the studio based course increased through the semester.

3.3 Anecdotes or Student Response Examples

Five students were interviewed at the end of the semester. Interview questions were open-ended and addressed perceptions of learning, interest and motivation, and sense of community with others in the class. Example questions include:

a. Learning: Did you find the process of completing programming projects helpful to you in learning about computer programming? Why or why not?

b. Motivation: Did the course keep you interested and motivated to learn? Why or why not?

c. Community: Are you comfortable giving and receiving feedback on computer programming?

Generally, students indicated that they had expected the course to be difficult but that they did enjoy it and learned from it. One student specifically indicated that he was pleasantly surprised by the creative projects. Most thought that the hands-on projects were more helpful than the book or lectures, and appreciated the opportunity to apply what they had learned. One interviewee indicated that the projects helped him think logically and analytically.

Interview responses were a bit more varied in relation to motivation. One student said that the course had decreased his/her interest in computer science. However, this was not the general consensus. All of the other interviewees reported that the course increased their interest in computer science. Specific aspects of the course that were mentioned included the opportunity to think through problems, theory (as opposed to programming) and practical application through projects.

One complaint about the peer review process was that the feedback was often lacking or sub-par. Two of the five interviewees specifically noted this lack of quality. However, all indicated that when they did get good feedback, it was helpful or interesting to them. For example, one student related that the reviews gave him/her insight. Another stated that the reviews helped further understanding. Other perceived benefits include increased interest, improvement of programming skills, and better understanding of the problem.

4. CONCLUSION

These preliminary findings hint at the potential of SBL as an instructional approach that could potentially increase student enjoyment in problem solving, and motivation and interest in computer science, all of which have been cited in recent literature as factors critical to reinvigorating computing education [3,5]. Our current work is on using the lessons of the previous semester to revise and implement SBL in CS2 and CS3 in spring 2008, and to compare not only affective changes in student perceptions but also student learning outcomes between traditional and SBL implementations of CS2 and CS3 at Auburn University. Furthermore, we plan to compare and correlate our findings with those from our partner universities.

5. ACKNOWLEDGMENTS

This paper is based on work done at Auburn University as part of a multi-university research project in which researchers from Auburn, 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 NSF 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 NSF.

6. REFERENCES

[1] 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.

[2] Hundhausen, C.D., Narayanan, N. H., and Crosby, M. E. 2008. Exploring studio-based instructional models for computing education. In Proceedings of the 39th ACM Technical Symposium on Computer Science Education (Portland, OR, March 12 - 15, 2008). ACM Press, New York, NY, to appear.

[3] Patterson, D. A. 2006. President's letter: Computer science education in the 21st century. Communications of the ACM 49, 3, 27-30.

[4] Pintrich, P. R., Smith, D. A. F., Garcia, T., and Mc Keachie, 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.

[5] Ross, J. 2007. Perhaps the greatest grand challenge: Improving the image of computing. Computing Research News 18, 3, 4.

[6] Schon, D. 1983. The Reflective Practitioner: How Professionals Think in Action. New York: Basic Books.