Computer-Supported Studio-Based Learning:

Interaction Requirements for a Learner-Centered Computing Pedagogy

N. Hari Narayanan[1], Dean Hendrix[1], Lakshman Myneni[1] and Margaret Ross[2]
Computer Science and Software Engineering Dept[1].
Educational Foundations, Leadership and Technology Dept[2].
Auburn University, Auburn, AL 36849
{ naraynh | hendrtd | mynenls | rossma1}@auburn.edu

ABSTRACT

Studio-Based Learning (SBL) is a novel learner-centered and collaborative learning model for computing education. This model has been under development and evaluation in computing courses at three universities (Auburn, Washington State and Hawaii) with support from the National Science Foundation. Adapted from architecture and art education, as well as from collaborative problem-solving pedagogies, SBL has shown great promise in enhancing student engagement and performance. We discuss key elements of SBL, illustrate the model by describing its implementation and evaluation in a computing course at Auburn University along with an interactive system we designed to support SBL, and conclude by enumerating desiderata for computer-supported collaborative learning systems for SBL.

Categories and Subject Descriptors

H.5.3 [**Information interfaces and presentation (e.g., HCI)**]: Group and Organization Interfaces---Computer-supported cooperative work. K.3.2 [**Computer and Information Science Education**]: *Computer science education.*

General Terms

Design, Experimentation.

Keywords

Computer-supported collaborative learning.

1. STUDIO-BASED LEARNING

Studio-Based Learning is an instructional technique that emphasizes collaborative, design-oriented learning. The pedagogy is rooted in Platonism and realized most often in modern architectural and design education (Green & Bonollo, 2003). In a multi-university NSF-supported education research project (Hundhausen, Narayanan & Crosby, 2008), we have adopted and adapted this pedagogy to suit computing curricula.

The key characteristics of SBL adapted to computing are: (1) students are given complex design problems with multiple possible solutions rather than toy assignments; (2) students typically work in groups (though individual work is also allowed) to construct their own artifacts (design solutions) while considering various tradeoffs that are involved; (3) students present their solutions to their peers and instructor in a studio session, often called a "design critique" – typically this takes place in a studio or lab space where students display and present their work as well as view and provide feedback on others' solutions, but this can be accomplished online as well; (4) the groups return to the drawing boards to improve and resubmit their solutions based on the feedback they received; (5) optionally, the resubmitted solutions may further be reviewed and critiqued by peers, and (6) optionally the groups may respond to this second round of feedback by explaining how their solutions can be further improved. 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 (Boyer & Mitgang, 1996).

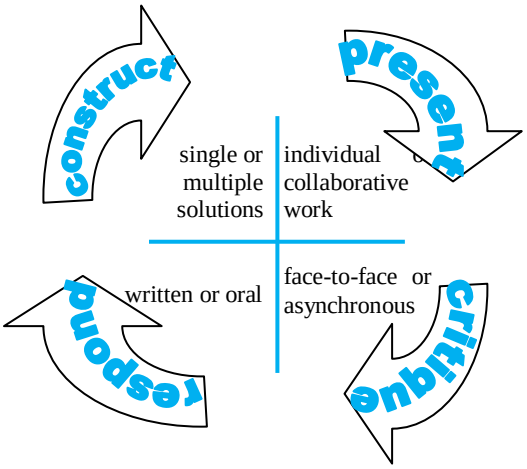

Figure 1. The SBL Model

What differentiates SBL from other pedagogical approaches are (1) the fact that the approach modifies labs or assignments and not the course lectures or exams, (2) the design critiques, and (3) the multi-staged approach to problem solving it espouses, in which students go through stages of artifact creation, presentation, critiquing, responding and refining (Fig. 1). The design critique is in fact considered the centerpiece of the model and forms the basis of a particular way of thinking and learning (Maitland, 1991). The expectation is that (1) working in groups and being empowered to take on a role that is traditionally reserved for the instructor – i.e., reviewing solutions and critiquing them – will make students more engaged and motivated to learn, and that (2)

Technical Report CSSE10-01, Department of Computer Science & Software Engineering, Auburn University, Auburn, AL 36849. January 2010.

by reflecting on and learning from multiple design exercises over the course of a semester, students will become more proficient in computational problem solving. Extant research (e.g., Hundhausen & Brown, 2004) supports this.

This paper illustrates the model of SBL in computing education by describing its implementation and evaluation in a course at Auburn University and an interactive system we designed to support SBL, and then, based on our experiences, proposes a set of requirements for Computer-Supported Studio-Based Learning (CSSBL) systems, which are collaborative learning systems tailored to the SBL pedagogical model.

2. IMPLEMENTATION & EVALUATION

The course Fundamentals of Computing II at Auburn University is a typical second course (henceforth called CS2) on data structures, algorithms and programming taken by sophomores. The instructional language is Java and the instructional IDE is jGRASP (jgrasp.org). CS2 is a four 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.

The course has five assignments. These 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.

All five lab assignments were studio-based. 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. Each assignment was completed by 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 initial design was to be submitted the night before the next lab. The next two lab sessions were design critiques during which each team presented their design to the class and received feedback from other students. These studio sessions 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 final two lab sessions were additional work sessions in which the teams could revise their designs and implementations, based on feedback from the studios. The final design and implementation were due at the end of the sixth lab day. Figure 2 shows an excerpt from design presentations done during studios.

After the assignment deadline had passed, the instructor assigned each student up to three other groups' work to review. This peer review required 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.

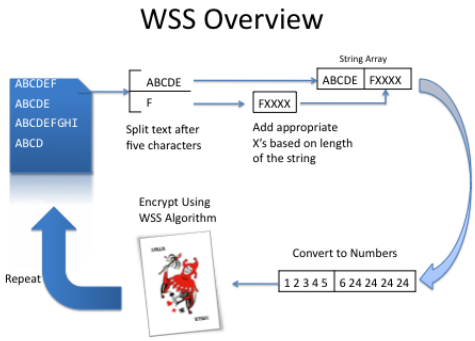

Figure 2. Excerpt from a studio presentation

The evaluation of the SBL model in CS2 (Hendrix et al., 2010) 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.

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 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 ($n = 55$). This is an indication of student learning.

To assess student attitudes and motivation, we focused on the following MSLQ subscales: intrinsic motivation, extrinsic motivation, self-efficacy, peer learning, critical thinking, self-regulation, and sense of community. We saw significant increases in intrinsic motivation (pre: 5.06, post: 5.57, $p<0.001$), extrinsic motivation (pre: 5.16, post: 5.54, $p<0.036$), self-efficacy (pre: 5.11, post: 5.92, $p<0.001$), peer learning (pre: 3.06, post: 3.89, $p<0.001$), and self-regulation (pre: 4.47, post: 5.26, $p<0.001$). Increase in the critical thinking subscale (pre: 4.11, post: 4.16) did not reach statistical significance.

To truly assess the effectiveness of this model, however, a studio-based offering of CS2 must be empirically compared to a traditional (non-studio) offering of the same course. To facilitate a rigorous experimental comparison, we offered the course in the studio format during the fall 2008 semester, results from which are reported above, and in the traditional (non-studio) format during the spring 2009 semester. This comparison data is summarized in Figure 3.

Though the post-test scores from the studio semester were higher than those from the traditional semester, this difference did not reach statistical significance. When we examined the graded items from the course (exams and programming assignments), we saw

higher performance with statistical significance from students who took the studio course. The average exam score in the studio semester was 74.99, while the average exam score in the traditional semester was only 65.74 ($p < 0.001$). This 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 (studio semester mean 79.64, traditional semester mean 70.06, $p < 0.01$). This indicates that students in the studio offering were able to develop better programs than the students in the traditional offering. Thus, we have accumulated evidence that the SBL model improves both the attitudes and performance of students.

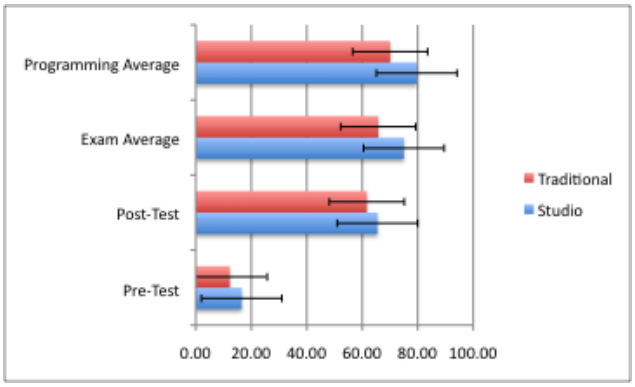

Figure 3. Student performance comparisons.

3. INTERACTIVE SYSTEM SUPPORT

From the outset, we felt that interactive system support was crucial to effective SBL implementation. As our initial focus was on classroom issues however, we chose to develop and use a basic system (Myneni, 2009) to support our SBL implementation, with the expectation that it would provide us with insights on what an "ideal" system might need. This system was hosted on a Linux server that users accessed through their web browsers. The server communicated with users as well as with a Mysql database. The system interface was built out of simple interaction objects: hyperlinks, buttons, pull-down menus and form-fill in PHP (see Fig. 4 at the end of this document).

It provided the following interaction capabilities to its users (instructor and students). Users logged in using ids assigned to them and passwords chosen by them. Students could individually upload and download files. Files created by some students could be made visible to other students by the instructor, for review purposes. The instructor could also create a rubric for reviews, in the form of a web questionnaire with text-boxes for free-form answers. Once the review assignments have been made, students could see how many solutions have been assigned to them for review, and submit their reviews on the web. Students could also see reviews of their work by others (anonymously) and could submit responses to those reviews. Reports about which students submitted what and when were generated for the instructor.

Our experience as instructors and feedback from students threw light on the weaknesses of this initial design. These fell into two categories: (1) lack of support to the instructor for creating and managing groups including group-wise access privileges/restrictions, and (2) lack of communication, sharing and collaboration support to students for group work. Reflecting on this experience led us to propose requirements for an ideal CSSBL system, discussed next.

4. REQUIREMENTS FOR A COMPUTER SUPPORTED SBL SYSTEM

Based on our experiences as instructors and on the feedback received from students, we realized that an advanced computer supported collaborative learning system has the potential to improve the student experience and make student and instructor tasks significantly easier. This led to the following design principle for a CSSBL system of the future: *An effective CSSBL system must support different users (instructors and students) in different ways during different stages of SBL.*

These are the twelve stages of a complete studio assignment.

(1) The instructor grouping students and assigning the design problem to be solved.
(2) Students obtaining their group assignments, and downloading or viewing on-line the design problem they have to solve.
(3) Students initially communicating and coordinating among themselves about how they will work as a group, whether there will be a group leader or not, allocating tasks and subtasks, arranging meetings etc.
(4) Group work to solve the design problem, which will typically involve individuals doing their assigned tasks, discussing the assignment with other group members and sharing artifacts, and eventual submission of an initial solution to the instructor.
(5) Group presenting their work to the instructor and rest of the class in a face-to-face or on-line studio session.
(6) Group work to improve their solution based on feedback received during the studio session and submission of a refined, intermediate solution.
(7) Instructor assigning which groups or individual students will review and critique which group's work and providing a rubric for review.
(8) Students obtaining their review assignments, and downloading or viewing on-line the design solution they are to critique.
(9) Individual or group work to review and critique others' solutions, and submission of reviews to the instructor.
(10) Students downloading or viewing all reviews and critiques of their solution.
(11) Group work to respond to the reviews and critiques, i.e., to explain how they might improve their design solution to address the problems other students identified, and submission of a final solution.
(12) Instructor reviewing each group's initial, intermediate and final solutions and the reviews it received from other students, and grading the assignment and reviews.

The users of the system switch between the instructor (stages 1, 7 and 12) and students (stages 2, 3, 4, 5, 6, 8, 9, 10 and 11). Their access privileges are also different, in that the instructor should be able to access and modify information regarding students and their assignments whereas students can only access and modify information assigned to them by the instructor. In other words, instructor will have privileges that students do not. There ought to also be a user designated as the sysadmin with all privileges.

An effective CSSBL system cannot simply have one type of user interface or interaction design because it has to support two

different types of users and, even more importantly, the nature of user tasks that the system needs to support change depending on the stage. In stage 1, user tasks are (1.1) grouping and (1.2) creating and uploading files for broadcast. In stage 2, user tasks are (2.1) retrieving and (2.2) viewing information and files. In stage 3, user tasks are (3.1) communication and (3.2) coordination of activities.

In stage 4, user tasks are (4.1) individuals working to complete their parts in solving the problem (this may be done using applications other than the CSSBL system) and (4.2) communicating with other group members. If the studio session (stage 5) takes place face-to-face, the system's role is limited to storing a video of the session for future review by the instructor and students. If the studio session is done on-line using the CSSBL system, then the corresponding user tasks are (5.1) broadcasting the video of the group's presentation in real time by streaming or by uploading a pre-recorded video that others can watch, and (5.2) engaging in text-based or audio-based discussions (question asking, answering, commenting and criticizing).

Stage 6 is similar to stage 4 with the same two user tasks. Stage 7 is similar to stage 1 with the same two user tasks. Stage 8 is similar to stage 2 with the same two user tasks. Stage 9 is similar to stage 4 with the same two user tasks. Stage 10 is similar to stage 2 with the same two user tasks. Stage 11 is similar to stage 4 with the same two user tasks. In stage 12, the main user task is (12.1) to retrieve different pieces of information in multiple ways – initial, intermediate or final solutions or reviews retrieved by student name or group identifier, and various summary reports.

This task analysis leads to a list of interaction capabilities that a CSSBL system needs:

(1) Support three classes of users (system administrator, instructor and students) with different access and modification privileges when they log in.

(2) Allow easy grouping of students in various ways: groups of different sizes and based on various grouping schemes such as random, alphabetical, rank-ordered (e.g., matched grouping based on class standing) or rule-based (e.g., male groups, female groups, mixed groups with at least two females, etc.). The instructor should be able to specify (or select) group sizes and grouping schemes, whereupon the system should automatically do the grouping. Once the groups have been formed, the system should automatically adjust the access privileges of students so that all students in the same group are able to share group files that are invisible to other groups. This supports user task 1.1.

(3) The instructor should also be able to assign work to the entire class (e.g., an assignment for the class) or assign different work to different groups (e.g., group 1 is assigned to review and critique the solution of group 2, group 2 is assigned to review and critique the solution of group 3, and so on). The instructor should be able to specify (or select) assignment schemes (e.g., random, group j's solution assigned to group j+1, etc.), whereupon the system should automatically adjust the access privileges of students so that all students in a group are now able to view or download files created by the groups assigned to them for review and critique. This supports user task 1.1.

(4) Provide for the uploading or streaming of electronic documents tagged with who (individuals or groups) is allowed to view and download them. This supports user tasks 1.2 and 5.1.

(5) Provide easy and flexible access to relevant information. For instance, when a student logs in, he or she should be able to easily locate documents assigned to him or her by the instructor or his or her group members. Similarly when an instructor logs in, he or she should be able to view and download any (or all) submission by individuals or groups, and be able to easily generate and view summary reports of student activity. This supports user tasks 2.1, 2.2 and 12.1.

(6) The system should provide broad and automatically tailored communication capabilities. For example, it should support multiple means of broadcast such as the instructor or a student emailing or texting all others, and a synchronous communication mechanism such as IM or chat or threaded discussions. It should also provide these communication channels in a tailored fashion so that as soon as the instructor determines student groupings, each student should get the capability to email, text, IM or chat with his or her group members with such communications remaining invisible to others. This supports user tasks 3.1, 3.2, 4.2 and 5.2.

It is worth mentioning here that though existing interactive applications provide one or more of these interaction capabilities, we are not aware of any single application that provides all of these functionalities in a seamless and usable way. A research challenge is to design a system with this spectrum of capabilities as a single application or as a mash-up that seamlessly stitches together a variety of applications. This is part of our ongoing research on SBL.

5. ACKNOWLEDGMENTS

The reported work is part of a multi-university research project in which other researchers from Auburn, the University of Hawaii, and Washington State University (M. Crosby, C. Hundhausen, M. Trevisan, and R. Vick) have participated. It is supported by the National Science Foundation under CPATH Grant No. CNS-0721927, but the views expressed are solely those of the authors.

6. REFERENCES

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.

Green, L. N., and Bonollo, E. 2003. Studio-based teaching: history and advantages in the teaching of design. *World Transactions on Eng. and Tech. Edu.*, Vol. 2, No. 2, pp. 269-272.

Hendrix, D., Myneni, L., Narayanan, H., & Ross, M. (2010). Implementing studio-based learning in CS2. To appear in *ACM SIGCSE Conference Proceedings*, ACM Press.

Hundhausen, C. D., and Brown, J. L. 2004. Personalizing and discussing algorithms within CS1 studio experiences: an observational study. *Proc. ICER 2005*, pp. 45-56.

Hundhausen, C. D., Narayanan, N. H., and Crosby, M. E. 2008. Exploring studio-based instructional models for computing education. *Proc. SIGCSE 2008*, pp. 392-396.

Myneni, S. M. (2009). Computer supported studio based learning. Unpublished Master's Thesis, Auburn University.

Maitland, B. M. 1991. Problem-based learning for an architecture degree. In: *The Challenge of Problem-based Learning*.

Figure 4. A screen capture of the initial CSSBL system design.