A Framework for CS1 Closed Laboratories

LEEN-KIAT SOH, ASHOK SAMAL, AND GWEN NUGENT
University of Nebraska, Lincoln

Closed laboratories are becoming an increasingly popular approach to teaching introductory computer science courses, as they facilitate structured problem-solving and cooperation. However, most closed laboratories have been designed and implemented without embedded instructional research components for constant evaluation of .the laboratories' effectiveness. As a result, it is not convenient to maintain and improve the laboratories over time so that they adapt to changing CS topics, curricula, and student needs. This article reports on an integrated framework for designing, implementing, and maintaining laboratories with embedded instructional research design. Although the activities reported here are part of our department-wide effort to cover CS0, CS1, and CS2, we focus here on the design and implementation of the labs for CS1.

Categories and Subject Descriptors: K.3.1 [**Computers and Education**]: Computer Uses in Education; K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*

General Terms: Design, Experimentation

Additional Key Words and Phrases: Closed laboratories, cooperative learning, instructional design

1. INTRODUCTION

Closed or structured laboratories are becoming an increasingly popular approach to teaching introductory CS courses [McCauley et al. 2003], per the recommendations of Denning et al. [1989] and ACM's Computing Curricula 1991 [Tucker et al. 1991]. Closed laboratories differ from open or unstructured ones in many ways. In open laboratories, students usually come and go at random, there is no set time at which they must all attend; whereas in a closed laboratory all students must attend at the same times. In each closed laboratory session, there is usually a set of objectives achieved by solving a set of specific problems. The students solve the problems, and by solving them, achieve the objectives. Each problem usually requires students to experimentally design and test small programs or use prewritten programs to investigate the problem. Further, in a closed laboratory, a lab instructor or supervisor is present to monitor and help the students, allowing the instructor to have a better grasp of how students solve problems.

There are several advantages in using closed labs in introductory CS courses. Students learn to be active learners at the beginning of their majors via goal-oriented problem-solving in a laboratory setting [Parker and McGregor 1995]. Doran and Langan [1995] demonstrated that in terms of Bloom's taxonomy [Bloom et al. 1964] labs promote the students' cognitive comprehension and application.

One study reported that although the closed labs did not improve the students' retention or project-completion rates, there was a qualitative improvement in student

The Reinventing CS Curriculum Project is supported in part by NCITE and CSE at the University of Nebraska.
Authors' addresses: L.-K. Soh and A. Samal, National Center for Information Technology in EDUCATION (NCITE), Computer Science and Engineering, University of Nebraska, 256 Avery Hall, Lincoln, NE 68588-0115; contact author's email: lksoh@cse.unl.edu; G. Nugent, Nebraska Center for Research on Children, Families, and Schools, University of Nebraska, 216 Mabel Lee Hall, Lincoln, NE 68588-0342.

learning in the closed lab sections [Kumar 2003]. Thweatt [1994] reported that in a CS1 statistical design course given to an experimental group (closed lab) and to a control group (open labs), students in closed labs consistently performed significantly better on comprehensive CS1 exams than those in open labs. Furthermore, opportunities to explore for possible solutions to problems posed in closed labs helped first-time programmers to overcome common hurdles, such as misconceptions about the nature of computers and programs [Lischner 2001]. Parker et al. [1990] found that closed laboratories demonstrate the scientific method and teach skills in data collection and analysis [Cupper et al. 1990]. The closed laboratory environment also facilitates cooperative learning among students (e.g., Oliver and Dalbey [1994] and also helps to increase student retention, as reported in Geitz [1994]. Finally, closed laboratories tend to provide a more flexible environment that can cater to students of different backgrounds and learning styles.

Developing closed laboratories for introductory CS courses could be time- and resource-consuming, especially when due to rapid changes in CS topics and the aptitudes of incoming students, the CS curriculum changes every few years. Hence, to find out whether and how a particular lab is useful, it is important to evaluate how laboratories impact student learning. Systematic quantitative and qualitative assessments of students via tests or surveys are needed. Moreover, to design the kinds of closed laboratories that will motivate students, a flexible design is needed so that cooperative learning and pair-programming can be conveniently implemented and evaluated.

Therefore, our approach to the design and implementation of closed laboratories at the Computer Science and Engineering Department at the University of Nebraska is based on an integrated framework that: (1) embeds instructional research design and assessment components into each laboratory; and (2) sets up an infrastructure for maintaining the educational resources over time. We developed a suite of CS1 (and CS2) labs with pre- and post-tests, activities, instructional scripts, and surveys. Due to the embedded design, we conducted several studies. For example, we evaluated the impact of cooperative learning vs. individual learning in the labs [Soh et al. 2005a]; the impact of learning objects vs. closed laboratories [Nugent et al. 2005]; and the impact of computer-supported cooperative learning vs. conventional cooperative learning [Soh et al. 2005c].

Our work here is part of the Computer Science and Engineering Department's effort (from 2002 on) at the University of Nebraska to revise and improve its computer science curriculum [Samal et al. 2005]. The project was motivated by several issues. First, the Association for Computing Machinery (ACM) and the IEEE Computer Society, the two leading professional organizations in computer science, released guidelines outlining core topics for a computer science degree program (ACM/IEEE 2001). After a careful review of the curriculum, our department decided to revise its CS curriculum to improve the quality of undergraduate CS education. Second, the department decided to put in place a curriculum that could adapt to rapid and continuous change in software development and information technology. Thus, our framework allows for evaluation and focused revision of educational resources.

In this article we present a systematic approach to design, implement, assess, and evaluate closed labs. We also report briefly on our educational research results.

2. RELATED WORK ON LABORATORY DESIGN

Closed labs were designed and implemented for a variety of introductory CS or computer programming courses: discrete structures, logic and computability [Hein 1993]; object-first programming [Roumani 2002[; XML [Bruce et al. 2004], and so on. Most closed labs, however, do not include pre- and post-tests, nor embedded instructional design as part of the framework. For example, Chavey [1991] built a set of structured laboratories for CS1, and his approach is similar to ours, in terms of the lab's overall design for teaching aspects of CS and computer programming under the supervision of a lab instructor. However, Chavey's approach did not include pre- and post-tests, and the review of the lab's design was based on qualitative surveys done by the students. This does not support the evaluation of different instructional designs, and the infrastructure for evaluating and improving the labs was not in place.

Our design borrows from the ideas of exploration [Lischner 2001] and object-first programming [Roumani 2002]. Lischner proposed a set of guidelines for including exploration in CS labs [Lischner 200]. An exploration is a structured dialog with the student in which the student reads a short program, answers questions about that program, makes predictions about the program's behavior, and then tests the predictions by running the program and answering follow-up questions. If a prediction is wrong, the student is asked to give a plausible explanation. Guidelines suggested by Lischner include (1) that the most effective explorations are short and to the point; (2) an exploration should contain a variety of questions, including some that are impossible to answer correctly; (3) exploration should direct the student toward effective models — after running the program, the student must be able to learn from the incorrect results; (4) exploration encourages students to pay attention to detail; and (5) exploration has measurable results using pre- and post-tests. In our lab design, we followed a similar set of guidelines for our experiments, while embedding instructional research into our labs. For exploration, each of our labs has components that are graded as part of the activity worksheets; student' understanding is assessed as part of the laboratory's pre- and post-tests.

Roumani [2002] outlined a set of design guidelines for an objects-first CS1 course; the guidelines that are similar to ours include the following: (1) labs are not to be thought of as evaluation tools but as educational instruments that complement the coverage in lectures and textbooks, as places where students should be allowed to discuss their tasks among themselves and/or to seek help from the instructors; (2) the labs must be portable and self-paced; (3) labs must be explorative in nature, where students have time to explore the correctness of a solution by writing tiny test programs; and (4) labs on object-based programming must be set in an abstraction that is credible and consistent. Each lab consists of three sections: exploration, exercises, and checking. Each explorative task asks the student to look for some feature in a given specification, to write a code fragment that uses (or implements) the feature, to add debugging I/O, and to then predict the output and verify it by actually running the code fragment. Checking requires the student to provide a method that accomplishes a stated task and to generate output with a specified format.

3. FRAMEWORK

Figure 1 shows our approach to the integrated framework of closed laboratories. The lab design phase consists of studies and discussions on CS topics among CS and education faculty and researchers. The CS part provides domain expertise, while the education part provides educational research design to ensure that each lab can support its goals. After this design phase, each lab is implemented, generating lab-related documents such as handouts, worksheets, and pre- and post-tests. Each lab is a stand-alone module. After the labs have been implemented we enter the third phase, which decides the particular instructional issues to be studied. For example, if we are interested in finding out how students working alone fare against those working in groups in the closed labs, then we modify the lab documents to allow us to keep track of the two groups of students, and designate the lab sections for, say, control and treatment groups. At this phase we also design and write our Institutional Review Board (IRB) application for approval. Then, in collaboration with the course and lab instructors, we deploy the labs. Our interactions are mainly with the lab instructors through weekly meetings, during which we monitor both student and instructor feedback on each weekly lab. The course instructor could also provide some feedback based on what he or she has been teaching during the week, or on the programming assignments the students were doing during the week, and so forth. This in-semester monitoring step allows us to (1) respond quickly to student needs and maybe change the remaining labs accordingly; (2) gather students' responses to the labs and address misunderstandings, provide additional help, or alert the course instructor; (3) make sure that the labs and the instructional studies are being carried out consistently by the lab instructor; and (4) obtain an overall view of the relative qualities of the labs -- for example, which labs are too difficult or too time-consuming. Thus, based on monitoring, we can evaluate the labs, and via the pre- and post-tests associated with each lab, assess student learning. We also collected surveys from students on each lab. Combining the data, we then refined the implementation of our labs, e.g., shortening a particular activity, providing a new pre-written test program, and so on. This refinement step is especially important in the first few semesters of deployment. As the lab materials become more refined, this step could be phased out until new labs are introduced.

The actual lab documents, the CSE department's administrative infrastructure, and an interdisciplinary group of researchers and faculty tie these phases together. The lab documents facilitate institutional memory and provide consistency between semesters and different instructors. The lab documents can also be disseminated for discussion and evaluation among peers interested in CS1 labs. The documents are also tangible items that we can improve over time. The administrative infrastructure that we refer to consists of the following components. First, as part of the Reinventing CS Curriculum Project, our group had the endorsement of the department to mandate our labs for courses not taught by members of our group, including conducting the instructional research studies. Second, our CS1 course usually has 70-90 students per semester, which translates to three or four lab sections with about 25 students each. This set-up allows us to conduct our studies conveniently by designating a lab for a particular design. Third, our university has an online testing tool (i.e., EDU) that allows us to put our pre- and post-tests online for students to take and then for the tests to be automatically graded and tallied. Our group comprises CS faculty and education researchers. Without this interdisciplinary collaboration, CS faculty alone could not have designed the labs with the embedded instructional research design, and could not have qualified the tests, surveys, and lab activities in terms of the students' cognitive processes. The emphasis was on developing novel approaches to deliver and assess course materials that promote "deep" learning, as well as to develop a framework to systematically evaluate the approaches and their short- and long-term effectiveness. Hence it is important that cognitive and experimental psychologists and instructional designers are an integral part of the effort from the beginning.

4. DESIGN AND IMPLEMENTATION

4.1 Laboratory Design

Each lab consists of five documents: (1) a student handout, (2) a laboratory worksheet, (3) an instructional script, (4) a pre-test, and (5) a post-test.

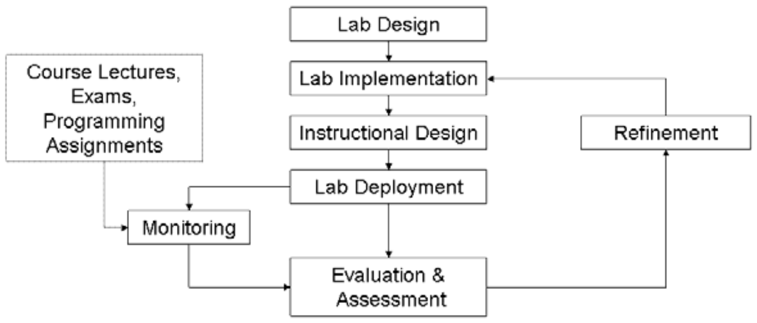

Fig. 1. Integrated framework for designing, implementing, deploying, evaluating, and refining laboratories.

The student handout serves several purposes: both as preparation guide and laboratory script. Each handout includes the lab objectives; a description of the activities to be performed during the lab (including the source code where appropriate); a list of references to supplemental materials that should be studied prior to the lab; and a list of supplemental references that can be reviewed after the student has completed the lab. The student handout also provides optional activities that can be completed during or following the lab to give students an opportunity for extra practice.

During each lab, students are expected to answer a series of questions for each activity and record their answers on a worksheet (paper). Worksheets contain questions specifically related to the lab activities and are intended to provide the students with an opportunity to find the answers through programming-based exploration. These worksheets also serve as an assessment tool to gauge the student's comprehension of topics learned and practiced in the lab.

In addition to the student handout, the lab instructor is given an instructional script with supplemental material that may not be covered during lectures, special instructions for t lab activities, hints, resource links, and useful insights. Additional space is provided at the end of the instructions for each activity, to allow the instructor to record his or her comments on the activity and suggestions for improving the lab.

The lab pre-tests are online, and students are required to pass them prior to coming to lab (however, students may take each pre-test as many times as necessary to achieve a passing score (80%)). The pre-test is open-book and open-note, and includes multiple-choice, short answer, and true/false questions. The goals of the lab pre-test are to encourage students to prepare for the lab and to allow them to test their understanding of the lab objectives and concepts prior to attending the lab. Questions for the pre-test are taken from a variety of sources, including the course textbook, other textbooks, and questions found on the web. Questions are categorized according to Bloom's taxonomy [Bloom et al. 1964].

During the last ten minutes of each lab, students take an online post-test as another measure of their comprehension of lab topics. Like the pre-test, questions are taken from a variety of sources and are also categorized according to Bloom's taxonomy [Bloom et al. 1964]. It should be noted that the goal of the post-test was designed to assess how well students understood the concepts after performing the activities specifically designed to reinforce the concepts.

The instructional script helps to institute a revision cycle, to enable us to learn how effective each lab is and to improve it for the next semester, and to be more responsive to the students' level of knowledge and comprehension of lab activities. The course and lab instructors also discuss the comments in the instructional script.

The post-tests help us embed instructional design into the labs and allow us to compare and contrast student performance in different lab sections, where students may be given different treatment.

Table I shows the list of CS1 laboratories and their corresponding objectives. We incorporated two event-driven programming labs; the first lab introduces students to the differences between event-driven programming and traditional sequential programming; the second lab addresses the capabilities and features of event-driven programming. We designed three testing and debugging labs. The first lab introduces the idea of debugging to students, teaches them how to debug using simple print statements and built-in IDE features and how to identify the different bugs. The second lab describes a more systematic, holistic approach to debugging, with different strategies such as a debug flag. The third lab introduces testing components, such as test cases and drivers, from the software engineering viewpoint. Overall, the three labs progress from simple, reactive debugging to more goal-directed debugging, and on to standardized testing.

Table I. Laboratory Topics and Goals for Our CS1 Course

Laboratories	*Objectives: Students should be able to*:
Introduction to Integrated Development Environment (IDE)	• Log into the network using a UNIX machine and a CSE account. • Create a directory on UNIX and traverse a UNIX directory structure. • Create and save a file on UNIX. • List the lab rules and hours of operation. • Understand the academic integrity policy of the Department of Computer Science and Engineering. • Know where to find help regarding CSE accounts and lab facilities. • Use an IDE to create and update source code and compile and execute a sample program. • Use the on-line hand in procedure to submit programming assignments and laboratory work
Simple Class	• Define and explain basic object terminology including class and object. • Identify the basic components of a Java program including private data member, public data member, public methods, private methods, and constructor. • Compile and execute a simple class. • Design and write a simple class.
Documentation	• Write clear, concise documentation describing a class and its members and methods. • Write Javadoc comments. • Use Javadoc to generate program documentation. • Determine when to use the three types of Java comments: single-line, block, and Javadoc.
Testing and Debugging I	• Describe the difference between testing and debugging. • Describe the differences between syntax, semantic (logic), and run-time errors. • Debug a simple program using print statements.

	• Debug a simple program using the debugger functionality in BlueJ.
File Input/Output	• Write Java programs that perform primitive file operations including open, close, read, write, and check properties. • Write Java programs that verify a file exists, check if a file is readable and/or writable, and check if a file is a directory. • Write Java programs that perform file I/O using the FileOutputStream, FileInputStream, FileWriter, and FileReader classes. • Explain the principles of stream I/O and distinguish between byte-oriented and character-oriented streams.
Applets and Applications	• Distinguish between an application and an Applet. • Create an Applet with multiple methods. • Execute Applets using three methods (i.e., Applet viewer, browser, and as an application). • Explain when to use Applets to solve a problem. • List the advantages and disadvantages of Applets.
Event-Driven Programming I	• Describe the difference between an event-driven model and a traditional sequential programming model. • Understand why event-driven programming is necessary and why it is used. • Understand the underlying principles and the meaning of event listening.
Exceptions	• Define what an exception is and why exceptions occur. • Explain the advantages of using Java exception handling over traditional error management techniques. • Distinguish between checked and unchecked exceptions. • Write a simple exception-handling routine to handle a single exception using *try–catch* and *try–finally* blocks. • Write methods that use the *throws* statement for exception handling
Graphical User Interface (GUI) and Swing	• Apply basic design principles to GUI design. • Use the Java Swing interface to create simple GUIs containing panes, buttons, labels, combo boxes, and radio buttons. • Write GUI applications that handle events.
Event-Driven Programming II	• Write a program with a single listener that handles multiple event *types* from a single event source. • Write a program with multiple listeners for a single event source. • Write a program with one listener for multiple event sources.
Testing and Debugging 2	• Use a variety of debugging strategies to identify bugs in programs. • Distinguish between a syntax error and a semantic error. • Use a debug flag to enable/disable debugging information.
Inheritance	• Explain what inheritance is, why it is used and when it is used. • Identify the superclass and subclasses in a given program. • Understand the meaning of the keywords: *abstract*, *protected*, *extends*, and *super*, and know when and how to use them.

	• Write abstract classes and abstract methods. • Explain the override property of inheritance. • Define a reusable class based on inheritance. • Derive new subclasses from a superclass to extend a solution to new problems. • Design and give an inheritance solution for a problem.
Testing and Debugging 3	• Write a driver module to test a Java program. • Use Java assertions to check program logic. • Write and execute a set of test cases to test an application. • Determine what types of tests are necessary for each type of problem (e.g., invalid input data, boundary tests, branch testing, loop testing).
Recursion	• Identify a recursive method. • Identify the three basic elements of a recursive method. • Determine when a problem should be solved using recursion. • Given a recursive mathematical definition for a problem, write a recursive Java method to solve the problem.

Interested readers may access our project website (http://cse.unl.edu/reinventCS) to download the lab modules above.

4.2 Instructional Research Design

In addition to embedding components that support instructional evaluation, the fact that we have multiple lab sections for each semester allows us to carry out rigorous instructional research. In the past four semesters, we carried out research designs that involve cooperative learning and learning objects.

For example, to study the impact of cooperative learning in labs, we set-up lab sections in the following manner [Soh et al. 2005a]. In one lab section the students worked in structured, cooperative groups with assigned group leaders. In another section they worked in non-structured, cooperative groups without a designated leader. Finally, in yet another section, students worked individually. Next, we used the total laboratory grades and pre- and post-tests that measure self-efficacy and motivation as our outcome measures. The combined outcome measures provide evidence of the effectiveness of laboratory pedagogy and achievement. Total laboratory grades were measured by combining post-test grades and worksheet scores for each lab. We also used the survey results on students' self-efficacy and motivation during the first and last week of the semester as a second outcome measure. We then performed an ANOVA statistics evaluation comparing total laboratory grades (or motivation/self-efficacy) for each laboratory group.

We also conducted another, similar, design for instructional research; but instead of cooperative groups we used learning objects [Nugent et al. 2005] to replace laboratories and to compare how three different groups of students performed: (1) students who were required to attend lab activities; (2) those required to go through the learning objects (but not lab activities); and (3) those required to go through lab activities *and* learning objects. Learning objects evolved from an object-oriented paradigm, which focused on the development of computer code components (called objects) that could be reused in multiple programming contexts. From an instructional standpoint, learning objects are small, stand-alone, mediated content "chunks" that can be reused in other instructional contexts and serve as building blocks to develop lessons, modules, or courses. We then performed an ANOVA statistical analysis to compare total laboratory grades for each laboratory section.

In addition, we conducted another study of a computer-supported cooperative learning system, called I-MINDS [Liu et al. 2004]. In our study, we compared the performance of three groups of students: (1) those who worked together via structured cooperative learning; (2) those who worked together through the Jigsaw cooperative learning model [Clarke 1994]; and (3) those who worked together using the I-MINDS-supported Jigsaw cooperative learning model. The goal of this study was to analyze the utility and feasibility of I-MINDS, in order to replace face-to-face interaction with computer-based interaction. With the pre- and post-tests, we were able to compare how the different groups of students performed.

4.3 Surveys

We developed two surveys to help evaluate our laboratories. The first measures the students' self-efficacy and motivation before and after the CS1 course [Soh et al. 2005a]. The second survey obtains additional feedback on laboratory resources from the students [Soh et al. 2005b].

Table II lists the Likert-type survey questions. Each question provides five choices: strongly agree (5), agree (4), neutral (3), disagree (2), and strongly disagree (1). Questions 1, 3, 7, and 8 gauge a student's self-efficacy; while questions 2, 4, 5, and 6 measure a student's motivation. This survey was administered at the beginning and end of the semester. The questions were adapted from the Motivated Strategies for Learning Questionnaire (MSLQ) developed by Pintrich and De Groot [1990]. Once we obtained the survey results, we computed a reliability measure (Cronbach's Alpha) with a mean and standard deviation value (a Cronbach's Alpha value of at least 0.8 is considered sufficient for statistical validity). We then computed ANOVA statistics that compared laboratory grades (post-tests) for each laboratory group.

Table III shows the survey questions administered to the students at the end of the semester. These Likert-type questions are designed to get additional feedback from the students about the laboratories' effects and effectiveness.

Based on the above, we were able to statistically correlate student responses to a pair of questions. In our research we investigated the relationships (a) between what students thought of the lab content/format and their motivation to prepare before the labs and pursue their studies further after the labs; (b) between how much students thought they learned and the usefulness of the labs; (c) and between the students' motivation and their view of the labs, by estimating motivation from the amount of time students spent before

Table II. Measures of Self-Efficacy and Motivation Before and After the CS1 Course

Self-efficacy/Motivation Survey Questions
1. I am confident in my CS knowledge and abilities.
2. I am motivated to learn more about CS/technology.
3. I did an excellent job on the problems and tasks assigned for this class.
4. I valued the opportunity to apply what was taught in lecture in a lab setting.
5. I valued the opportunities to interact and collaborate with other students in the class.
6. I was academically prepared to take this course.
7. Compared to other students in the class, I did well.
8. I think I will receive a good grade in this class.

Table III. Additional Survey Feedback after the CS1

Questions
1. The *pre-reading assignments* are useful
2. I always read all or most of the *pre-reading assignments*
3. I read the entire *student hand-out* before lab each week
4. I adequately prepare for labs each week
5. I usually try to do the *extra credit* each week
6. The labs are generally very useful
7. I have learned a lot in the labs each week
8. The *supplemental links* are useful
9. I visit several of the *supplemental links*, even after the labs
10. I take the labs seriously
11. My goal for each lab is to learn as much as I can
12. I learn something new in each lab
13. The labs help me in doing the homework and exams
14. I look forward to going to lab each week
15. My goal for each lab is to get it done
16. I like the format of the labs (hand-out/worksheet)
17. I enjoy the labs
18. I will use what I have learned in the labs in the future
19. The labs are an important supplement to the course
20. I would not be doing as well in the course without the labs

each lab and what they thought of it relative to their overall success in the course. We also analyzed what students who reported that they would not have done as well in the course without the labs thought of the labs' usefulness.

4.4 Other Analyses

To validate the labs' role and impact on learning CSI, we also systematically performed correlation analyses to find any predictive relationships that laboratory performance might have on other outcomes [Soh et al. 2005b]. Correlation values ranged from -1 to +1, with numbers closer to -1 or +1 being more significant, and numbers at or near zero showing no predictive relationship. Correlation was determined by measuring the degrees to which the two variables differed, adding them together and dividing the result by the degree to which the two variables varied separately. Specifically, we computed correlations between total lab scores and total test scores, and total lab scores and total homework scores. We also computed correlations between each lab score and each group of questions in the placement examination. The placement examination [Nugent, to appear], part of our Reinventing CS Curriculum project, was designed and deployed to place students in CS0 and CS1. The placement exam is based on CS concepts and programming knowledge, and is categorized into four levels of Bloom's taxonomy: knowledge, comprehension, application, and analysis [Bloom et al. 1964]. The exam is supported by comprehensive educational research in order to analyze student performance and improve test measurement indices. It is also rooted in a departmental initiative to reinvent our CS curriculum, with the long-term goal of better preparing students for upper-division courses, and ultimately for careers in the software and IT industries. And finally, to measure student learning in the course, the exam is combined with an additional 25 questions for pre-post testing

5. RESULTS

We now briefly report on our instructional research results.

- Both cooperative learning groups performed better than the individual group (e.g., F(2,181) = 4.681, p =< .05 for three semesters worth of data) as measured by the final total lab grade. There proved to be no significant difference between the structured and unstructured groups.
- Student motivation and self-efficacy seemed to actually decrease from the beginning of the semester to the end. We believe this may be due to an inflated sense of self-efficacy upon entering the course.
- The laboratory design was useful and appropriate in its format and content.
- Motivated students thought that the lab materials were useful and were willing to go through the content before each lab. Unmotivated students, however, failed to appreciate the labs. So it seems that the labs were not able to increase motivation for those students who did not want to prepare for the labs, prompting us to investigate ways to "enliven" some of the labs.
- We were encouraged by the observation that student performance in the labs correlates with their performance on exams and homework assignments.
- Students, without the benefit of face-to-face interactions, were able to make use of the computer-supported cooperative learning tool, and performed as well as students with face-to-face interactions in the post-test of each lab. However, we also observed that students rated their team activities more highly in face-to-face interactions. There are also indications that when a student was critical of his or her peers, this student's post-test score was likely to be higher.

For more details on our research results, please refer to Soh et al. [2005a] for discussions on cooperative learning design; refer to Soh et al. [2005b] to examine the relationships between closed labs and course activities in CS1; and see Soh et al. [2005c] on the computer-supported cooperative learning study.

6. CONCLUSIONS

We have reported on an integrated framework for designing, implementing, and maintaining laboratories with embedded instructional research design. Our framework has in place educational resources such as laboratory documents (e.g., hand-outs, pre- and post-tests, activity worksheets, instructional scripts), instructional research design, and surveys. The pre- and post-tests and surveys allowed us to conduct systematic studies on student performance, tracking it in each lab topic and across different semesters. The instructional scripts provided a paper trail for collecting comments for the further improvement of lab activities. Our research design and its accompanying ANOVA statistics and correlation analyses also provide insight on the effectiveness of specific pedagogy, lab materials, and relationships with other course activities. This in turn allowed us to revise our lab designs over the past few semesters as well.

We have also written a set of CS2 lab activities for our project and will conduct similar experiments to assess how CS2 labs impact student learning. We are currently designing a set of CS0 lab activities, and will also design additional learning objects and evaluate their impact on student learning, to continue improving the validity and confidence of the results of our project.

ACKNOWLEDGMENTS

We thank A. Zygielbaum, R. Sincovec, J. Lang, S. Person, C. Riedesel, C. Chen, S. Kasinadhuni, J. Bernadt, A. Kosenander, T. Fink, K. Halsted, X. Liu, J. Mallik, and S. Das for their help in implementing this project.

REFERENCES

ACM/IEEE JOINT TASK FORCE ON COMPUTING CURRICULA. 2002. *Computing Curricula 2001*: *Computer Science.* IEEE Press.

BLOOM, B.S., MESIA, B.B., AND KRATHWOHL, D.R. 1964. *Taxonomy of EducationalObjectives.* In two volumes: *The Affective Domain* and *The Cognitive Domain.* David McKay, New York.

BRUCE, R., BROCK, J.D., AND BOGERT, K. 2004. X-Lab: XML-based laboratory exercises for CS1. In *Proceedings of the 42nd ACM Annual Southeast Regional Conference* (Huntsville, AL). ACM, New York, 434-435.

CHAVEY, D. 1991. A structured laboratory component for the introductory programming course. In *Proceedings of the 22nd SIGCSE Technical Symposium on Computer Science Education, SIGCSE'1991* (San Antonio, TX). ACM, New York, 87-95.

CLARKE, J. 1994. Pieces of the puzzle: The jigsaw method. In *Handbook of Cooperative Learning Methods*, S. Sharan, ed., Greenwood Press, Westport, CT, 1994.

DORAN, M.V. AND LANGAN, D.D. 1995. A cognitive-based approach to introductory computer science courses: Lessons learned. In *Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE'95, Nashville, TN). ACM, New York, 218-222.

GEITZ, R. 1994. Concepts in the classroom, programming in the lab. In *Proceedings of the 25th SIGSE Symposium on Computer Science Education* (SIGCSE'94, Phoenix, AZ). ACM, New York, 164-168.

HEIN, J.L. 1993. A declarative laboratory approach for discrete structures, logic and computability. *ACM SIGCSE Bull. 25,* 3 (1993), 19-25.

KUMAR, A.N. 2003. The effects of closed labs in computer science I: An assessment. *J. Comput. Sci.Colleges 18,* 5 (2003), 40-48.

LISCHNER, R. 2001. Explorations: Structured labs for first-time programmers. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education* (SIGCSE'2001, Charlotte, NC). ACM, New York,154-158.

LIU, X., ZHANG, Z., SOH, L.-K., AL-JAROODI, J., AND JIANG, H. 2003. A distributed, multiagent infrastructure for real-time, virtual classrooms, In *Proceedings of the International Conference on Computers in Education* (ICCE2003, Hong Kong, Dec. 2-5), 640-647.

MCCAULEY, R., PARRS, W., POTHERING, G., AND STARR, C. 2003. A proposal to evaluate the effectiveness of closed laboratories in the computer science curriculum. *J. Comput. Sci. Colleges 19*, 3 (2003), 191-198.

NUGENT, G., SOH, L.-K., SAMAL, A., AND LANG, J. **2006**. A placement test for computer science: Design, implementation, and analysis. *Comput. Sci. Edu. 16,* 1 (2006), 19-36.

NUGENT, G., SOH, L.-K., SAMAL, A., PERSON, S., AND LANG, J. 2005. Design, development, and evaluation of a CS1 learning object for CS1. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE'2005, Monte de Caparica, Portugal, June 27-29), 370.

OLIVER, S.R. AND DALBEY, J. 1994. A software development process laboratory for CS1 and CS2. In *Proceedings of the 25th SIGSE Symposium on Computer Science Education* (SIGCSE'94, Phoenix, AZ). ACM, New York, 169-173.

PARKER, B.C. AND MCGREGOR, J.D. 1995. A goal-oriented approach to laboratory development and implementation. In *Proceedings of t he 26th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE'95, Nashville, TN). ACM, New Yprk, 92-96.

PARKER, J., CUPPER, R., KELEMEN, C., MOLNAR, D., AND SCRAGG, G. 1990. Laboratories in the computer science curriculum. *Comput. Sci. Edu. 1*, 3 (1990), 205-221.

PINTRICH, P.R. AND DEGROOT, E.V. 1990. Motivational and self-regulated learning components of classroom academic performance. *J. Edu. Psychol. 82,* 1 (1990), 33-40.

ROUMANI, H. 2002. Design guidelines for the lab component of the objects-first CS1. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (SIGCSE'2002, Covington, KY). ACM, New York, 222-226.

SAMAL, A., NUGENT, G., SOH, L.-K., LANG, J., AND PERSON, J. 2005. Computer science curriculum at theUniversity of Nebraska. In *Technology-Based Education: Bringing Researchers and Practitioners Together*. L. Pytlikzillig et al. eds. Information Age Publishing, 203-224.

SOH, L.-K., SAMAL, A., PERSON, S., NUGENT, G., AND LANG, J. 2005a. Closed laboratories with embedded instructional research design for CS1. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (SIGCSE'2005, St. Louis, MO, Feb. 23-27). ACM, New York, 297-301.

SOH, L.-K., SAMAL, A., PERSON, S., NUGENT, G., AND LANG, J. 2005b. Analyzing relationships between closed labs and course activities in CS1. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE'2005, Monte de Caparica, Portugal, June 27-29), 183-187.

SOH, L.-K., KHANDAKER, N., LIU, X., AND JIANG, H. 2005c. Computer-supported structured cooperative learning. In *Proceedings of the International Conference on Computers in Education* (ICCE'2005, Singapore, Nov. 28-Dec. 2).

THWEATT, M. 1994. CS1 closed lab vs. open lab experiment. In *Proceedings of the 25th SIGCSE Symposium on Computer Science Education* (SIGCSE'94, Phoenix, AZ). ACM, New York, 80-82.

TUCKER, A. ET AL. 1991. *Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force.* ACM Press, New York.

Received August 2005; revised December 2005; accepted February 2006