Experiences with lab-centric instruction

Nathaniel Titterton, Colleen M. Lewis, Michael J. Clancy

Computer Science Department, University of California Berkeley, Berkeley, USA

Dr. Nathaniel Titterton, Computer Science Division, EECS Department, University of California, Berkeley, CA 94720-1776 nate@berkeley.edu

(Received 16 October 2009; final version received 30 March 2010)

Lab-centric instruction emphasizes supervised, hands-on activities by substituting lab for lecture time. It combines a multitude of pedagogical techniques into the format of an extended, structured closed lab. We discuss the range of benefits for students, including: increased staff interaction, frequent and varied self assessments, integrated collaborative activities, and a systematic sequence of activities that gradually increases in difficulty. Instructors also benefit from a deeper window into student progress and understanding. We follow with discussion of our experiences in courses at U.C. Berkeley, and using data from some of these investigate the effects of lab-centric instruction on student learning, procrastination, and course pacing. We observe that the lab-centric format helped students on exams but hurt them on extended programming assignments, counter to our hypothesis. Additionally, we see no difference in self-ratings of procrastination and limited differences in ratings of course pace. We do find evidence that the students who choose to attend lab-centric courses are different in several important ways from students that choose to attend the same course in a non-lab-centric format.

Keywords: closed lab; embedded assessment; collaborative learning; procrastination; computer programming; blended learning environments; pedagogy; CS1; CS2.

Introduction

Lab-centric instruction is based on the premise that computer science students learn more by doing than by listening to lectures (Laurillard, 1993). It increases hands-on, supervised lab time while reducing lecture and discussion time. We have designed, piloted, and run lab-centric versions of several lower-division courses at U.C. Berkeley since 2002. In this paper we spend the first half presenting a detailed description of lab-centric instruction and its benefits, and in the second half reporting on our experiences with these courses, including comparisons between lab-centric and non-lab-centric courses on student learning, attitudes, and behavior.

What is lab-centric instruction?

Lab-centric instruction is not a single innovation. It is the combination of dramatically increasing emphasis on lab time with the integration of promising pedagogical practices that this enables. In shifting to lab-centric instruction we have combined a multitude of common and proven techniques to enhance student learning, the result of which changes how students and instructors spend their time (Clancy, Titterton, Ryan, Slotta, & Linn, 2003). At a basic level, lab-centric instruction uses a closed lab, defined as one that provides activities to be accomplished during a scheduled time under the supervision of a member of the course staff. A closed lab can be contrasted with an open lab, in which student tasks are less structured and attendance is optional.

Our typical lab-centric course provides one hour of lecture and a total of six hours of supervised lab each week. The lecturer and the lab instructor are usually different people, with

the course instructor lecturing and the teaching assistant supervising the labs. We have used this format in lower-division courses that are otherwise taught in our department's traditional format: a weekly schedule of three hours of lecture, two hours of closed lab, and one hour of discussion led by a teaching assistant. In this paper, we will refer to this format at "traditional", although it does differ from common practices at other institutions. We discuss this point further in the section below on constraints for adopting the lab-centric format.

Several factors contributed to evolution of our lab-centric format. We have run closed labs in our lower-division courses since the early 1980's. In the early 1990's, we introduced peer instruction (Mazur, 1997) in our courses to try to make lecture time more productive. Elsewhere on campus, Marcia Linn was supervising the development of the WISE learning environment, on which our first delivery system was overlaid, and researching the pedagogical effects of it (Linn, Davis, Bell, 2004). In transitioning from the traditional format to lab-centric instruction, we have added an additional hour of supervised work, moving from six hours a week to seven. We believe that this does not increase the amount of time students spend on the course. In lab-centric instruction there are no office hours, and students spend some of their time in lab working on homework or projects that would normally be attempted without supervision. We believe benefits to student experience and learning are mainly attributable to the other structural benefits and pedagogical innovations in lab-centric instruction.

How students spend their time in lab-centric instruction

The students spend the majority of lab time working individually or in small groups working through a sequence of activities presented through an online learning management system. Originally, this was a locally built system based on the WISE learning environment (Slotta & Linn, 2009); currently, we use the Moodle system with local modifications (Dougiamas & Thomas, 2003). Some three-hour lab sessions, especially those early in a course, can contain forty to fifty separate activities.

The list below shows some of the types of activities used within our lab-centric courses. Most of these activities are well established research-based pedagogical tools and techniques.

- Quiz - A lab session begins with a short quiz, covering material from the previous class. Student responses are reviewed by lab-instructors (the teaching assistants, in our courses) before feedback is given to students.
- Gated collaboration - A question is posed to the students. After responding online, a student is able to view the answers provided by their classmates. In this way, students are required to engage with the content before reading possible answers. Gated collaborations typically use questions that lend themselves to multiple solutions where viewing other students' answers shows a diversity of answers and provides students with experience evaluating alternative responses.
- Self Assessment - An embedded assessment where students get instant feedback customized according to their responses.
- Exercise - An activity, such as a programming or debugging task, where there is no explicit assessment or initial collaboration.
- Brainstorm/Reflection – A prompt that encourages students to reflect or make a note in their course journal. These may encourage reflection on problem solving or debugging process or on current understanding.
- Design/Defend discussion - An activity that encourages the evaluation and refinement regarding design, either online or face-to-face.

- Discussion (online or face-to-face) - A discussion that begins through an online forum and continues with face-to-face discussion depending upon the pacing of students.
- Project - A three- to four-week extended homework assignment with time in lab dedicated to project work and evaluation of progress toward a solution.

How instructors spend their time in lab-centric instruction

All lab time is supervised by a member of the course staff who monitors student progress both on- and offline. Online presentation of activities frees instructional staff to roam during class time, answer questions, tutor one-on-one or one-to-few, and help all students make progress. Tools that allow instructors to monitor student interactions with the system help the instructors to identify and assist those students who have misconceptions, or those students who are falling behind.

Example lab-centric curriculum

Table 1 details a sample day of curriculum on testing and debugging from a lower-division lab-centric course. The activities in it are based on a procedure named contains1MoreThan that should determine whether its first string argument is the result of inserting exactly one character into its second argument. Addressing the issue of positive test bias (Zweben, Stringfellow, and Barton, 1989; Leventhal, Teasley. and Rohlman, 1994) is a central focus of this sample day.

Activity type	Activity contents

Gated collaboration	A student proposes to test `contains1MoreThan` with the following calls:
	`System.out.println (contains1MoreThan ("ABC", "BC"));`
	`System.out.println (contains1MoreThan ("ABC", "AC"));`
	`System.out.println (contains1MoreThan ("ABC", "AB"));`
	Give the shortest implementation of `contains1MoreThan` that would produce the right answers for those calls.

	Describe the deficiencies of this set of test calls.
	Provide a missing test case, along with a description of a bug that it would reveal.
Self Assessment	The class `buggy1.class` contains a program with a buggy version of `contains1MoreThan`. Generate a test suite that reveals the bug.
	The class `buggy2.class` contains a program with a buggy version of `contains1MoreThan`. Find a pair of arguments that reveal the bug and are as short as possible.
Exercise	The class `buggy3.class` contains a program with a buggy version of `contains1MoreThan`. Find the bug. Keep track of the test cases you used.
Reflection	Put an entry in your journal describing test cases that you needed to reveal the bugs of the previous activities, especially those you hadn't previously thought of.
Design/defend discussion	Based on your experiences on the previous activities, design and defend a thorough black-box test suite for `contains1MoreThan`.
Gated collaboration	Suppose a solution to the function is based on the following pseudocode {…} Design and defend a thorough glass-box test suite for `contains1MoreThan`.
Brainstorm/ reflection	List boundary conditions for the following complex problem {…}. Describe your solution.
Discussion (online or face-to-face)	(*for half the class*) Explain why it would be easier to test your own version rather than a version that someone else has written. (*for the other half*) Explain why it would be easier to test someone else's version than your own. Then try to persuade someone in the other group to agree with you.

Table 1. An overview for a day of curriculum on testing and debugging in CS 1.

Benefits and constraints

Structure: opportunities from the structure of lab-centric instruction

Opportunity for active learning

The essential motivation for creating a lab-centric course is the shift in emphasis from lecture to lab. Such a shift is well supported by literature regarding active learning and more specific research showing the benefits of labs. Numerous commentators have observed that a lecture is likely to lead at best to passive learning (e.g. Laurillard, 1993; McKeachie, 2005; Elbe, 1976). In contrast, hands-on lab time engages students in active and discovery learning. Research on discovery learning suggests that it better supports domain-specific learning than direct instruction (see Chen & Klah, 1999 for a review). Urban-Lurain and Weinshank (1999) also describe a course that trades lecture time for lab time, which shares some features with our approach.

Research, as surveyed by McCauley, Pharr, Pothering, and Starr (2004), supports the use of closed labs as used in lab-centric instruction. Hofstein and Lunetta (2003) surveyed characteristics of labs in science courses, which we believe are comparable to computer science

labs. They found that labs provide an environment for experimentation and inquiry, facilitate student community and collaboration, and foster positive and improved student attitudes and interest in science. In addition, Parker, Cupper, Kelemen, Molnar, and Scragg (1990) note that closed-lab activities facilitate the integration of theory and practice, and alleviate the tension between learning and evaluation.

In line with these positive results, it appears that there is a trend toward the use of closed labs in lower-division computer science courses (McCauley & Manaris, 2002; McCauley, Starr, Pharr, Stalvey, & Pothering, 2006). While the use of a closed lab does not make our pedagogy unique, the emphasis on closed labs over lectures is more dramatic than more typical class organizations and affords a number of additional structural benefits.

Opportunity for targeted tutoring

The lab format effectively decreases the student-teacher ratio, allowing instructors to engage in one-to-one and group tutoring sessions, fostering a kind of apprenticeship learning (Collins, Brown, & Newman, 1989). Students in our lab-centric courses ask questions in the context of the day's lab activities. This provides course staff relevant context and time to diagnose and address the root of the issue, which is typically not available in questions arising in email or office hours. By monitoring student learning on- and offline, instructors can initiate tutoring sessions at the moment a misconception or confusion is occurring, making tutoring sessions more efficient and affording deep student learning (Bransford, Brown, & Cocking, 1999; diSessa & Minstrell, 1998).

Individual or small group tutoring has long been known to be the best method of instruction (Bloom, 1984; Cohen, Kulik, & Kulik, 1982), and has been shown to work without extensive training on the part of the instructor (e.g., Chi, de Leeuw, Chiu, & La Vancher, 2001). The main drawback of tutoring—that it is very time-intensive and, therefore, costly—is reduced by the lab-centric format. Students spend most of their time learning from the materials and their collaborators, while the tutoring is efficiently used when confusion is present. At our institution the estimated work time for teaching assistants has remained constant in transitioning from our traditional format to the lab-centric format.

Opportunity for support throughout learning

In Berkeley's traditional lower-division CS courses, the student activities typically vary in difficulty, with leaps in complexity between labs, homework and projects. Labs typically focus on introducing low-level concepts, which involve learning and practicing relatively simple tasks. Subsequent homework assignments and projects represent a marked increase in complexity and difficulty. In these traditional courses, the least challenging activities take place with the support of course staff in lab, while students work on the most difficult homework and projects in relatively isolated. Lab-centric courses can provide a wide variety of activities, arranged so that the gradient between easy and hard concepts is gentler than in traditional courses. Moreover, lab content can span a range of difficulties, such that students have the support of their peers and instructor when they tackle the most difficult challenges in the course.

Opportunity for developing a non-defensive community

The structure of lab-centric instruction allows the staff to get to know the students and build a community where we strive to avoid the pitfalls present in other computer science classrooms, such as the defensive climate described by Barker, Garvin-Doxas, and Jackson (2002). Staff

easily learn the names of the students with whom they spend such so much time coaching and supporting. The offline collaboration activities encourage students to get to know and help each other. Opportunities for students to show off and engage in hostile behavior are limited because of the distributed context of the labs. Lab-centric courses can create an environment that is cooperative, relaxed and focused on student learning.

Opportunities for course refinement to improve learning

The fine-grained activities in our lab-centric curricula expose a substantial amount of information about student conceptions and misconceptions, some of which instructors may be unaware of. In subsequent offerings, activities may easily be added or revised to address the confusion (Ryan, 2006). Our curriculum management system provides tools that allow the instructor to monitor student progress and, more importantly, understanding. This information can be shared amongst course staff to provide better direction to teaching assistants and better feedback for the lecturer for future sessions. With the reduced amount of lecture and the resultant decrease in lecture preparation time, instructors can focus on refining or inventing curriculum components (Clancy *et al.*, 2003). Lectures that remain can be tightly focused on actual rather than assumed student needs.

Opportunities for flexibility

The online presentation of lab activities provides an opportunity for students to progress through the material at their own pace. However, for the purpose of classroom collaboration it is optimal for students to be roughly synchronized. We believe this balance between flexibility for the student and synchronization for effective collaboration represents an unavoidable tension in lab-centric courses. We believe that this balance may be adjusted to meet the goals and culture of the adopting institution.

Constraints on the adoption of the lab-centric format

Several constraints affect the wider adoption of the lab-centric approach in computer science courses (Titterton & Clancy, 2007). The following section discusses constraints in implementing lab-centric instruction and the corresponding solutions developed at our university.

Constraints on Lab Staff

At institutions where teaching assistants lead sections (such as ours) as well as institutions where a single instructor meets with a single lab-sized group of students (e.g., high schools and most community colleges), the lab-centric format need impose no additional staffing requirements over the traditional format.

In our traditional courses a 10-hour/week teaching assistant has 4 contact hours: 2 hour in lab, 1 hour leading a discussion section, and 1 hour of office hours. In lab-centric courses, teaching assistants have 6 contact hours, all taking place in the lab. We believe that the elimination of discussion sections substantially reduces teaching assistant preparation requirements. Additionally, we eliminate teaching assistants' office hours, letting their increased time in lab fulfill that function.

Constraints on Course Instructor

The labcentric approach drastically reduces lecture preparation for course instructors. New responsibilities include increased supervision and training of lab staff as well as the creation of

lab activities. While this creation of lab activities is a large investment of instructor time, we have a growing body of refined content and, when authoring new materials, find that we rarely start from scratch. The key to easing this constraint is the sharing of content.

We are currently developing an innovative authoring environment to support both novice and experienced instructors in developing and refining curriculum (Carle, Clancy, & Canny, 2007; Carle, Canny, & Clancy, 2006). Additionally, we plan to soon provide an online collaborative space for lab-centric instructors to gather and support each other. We are optimistic about the potential benefits of such collaboration because lab-centric courses allow curriculum to be more effectively shared and refined than lecture slides. Lab-centric content unambiguously specifies what the students do, because the pedagogical details are captured within the lab curriculum rather than hinted at by bulleted lists.

Constraints on Lab Space

With more scheduled lab time, some institutions could experience a shortage of computer laboratory space. At our university we have curtailed the open lab hours to accommodate the increased demand on laboratory space. We expect that the growing ubiquity of student-owned computers and wireless connectivity will ease these constraints.

Activities: Benefits from the Activities of Lab-Centric Instruction

The following section discusses the research-based foundation of individual activities used within our lab-centric instruction. (A considerable amount of the research has come from the WISE project—Web-based Inquiry Science Environment—directed by Marcia Linn and supported for many years by the National Science Foundation.) There is no particular set of activities that, by their presence, strictly define the lab-centric approach. However, the types of activities described below were incorporated throughout the design of our curricula because of their compatibility with our goals of collaboration, frequent feedback and scaffolded learning, and their amenability to delivery in a closed lab setting.

Collaboration

Research has shown that collaborative activities have a myriad of benefits in all phases of learning and coursework (Slavin, 1995; Koshman, 1996; Koshman, Hall, & Miyake, 2002; Johnson, Johnson, & Smith, 1991; Hoadley 2004; Dubinsky, Mathews, & Reynolds, 1997). Still, collaboration in Berkeley's traditional computer science courses is limited to project work and the solving of laboratory exercises. That is, students collaborate during the act of programming, but rarely while engaging in metacognitive reflection on the practice of programming or higher-order thinking. Through additional in-class lab activities we can provide the opportunity to facilitate collaboration around higher-order thinking and metacognitive reflection through both on- and offline collaboration.

Face-to-face collaboration. The closed labs in a lab-centric course present a blended environment of on- and offline collaboration. Our face-to-face collaboration activities are a cornerstone of our lab environment and prime the classroom community for further online collaboration. The face-to-face collaboration activities range from pair programming (Williams & Upchurch, 2001) and group problem solving to debates about program design, discussion of techniques for time management, and sharing of debugging tactics. These face-to-face collaborative activities serve to reinforce the core curriculum and support students in building

skills such as explaining, questioning, and reflecting that we consider essential to the education of future computer scientists.

The threaded discussions activity. Contribution to a threaded discussion is a frequent homework assignment in our courses. These activities are intended to encourage metacognitive reflection such as considering why language features exist, reflecting on best practices in testing and debugging, examining mental models of recursion and other algorithms, and so forth. Students are encouraged to exercise this metacognitive reflection and have the opportunity to see the reflections of other members of their course.

The gated collaboration activity. We developed a "gated collaboration" activity to supplement typical class collaboration with online collaboration. In a gated collaboration, students are required to provide an answer to a challenging question *before* they view the answers of their classmates. The format ensures students engage in critical thinking by making them both construct an answer and review other student answers. Gated collaboration provides the opportunity for students to answer tough questions, yet also provides opportunities for sharing strategies for debugging or avoiding particular kinds of errors, for comparing design or development approaches, and for criticizing and defending alternatives. This collaborative sensemaking fosters engagement across the students in the class. With experience reading and reviewing other student answers, students can practice evaluating and critiquing explanations (Chi *et al.*, 1994).

Formative assessment

Educational research has shown that students have difficulties monitoring their own understanding in computer science courses. (Davis, Linn, & Clancy, 1995; Linn & Kessel, 2003). Many students claim that they understand aspects of a programming language, when in fact they are basing their understandings on faulty analogies with natural language processing (Soloway & Spohrer, 1988). Computer science instruction, as well as instruction in other disciplines, should aim to provide formative assessments of student understanding, provide feedback, and wherever possible adjust instruction based on evaluation (Bransford *et al.,* 1999). In Berkeley's traditional-format courses, students may receive no feedback—even pertaining to possible misconceptions of core concepts—until they perform badly on the first midterm exam.

The self-test activity. Research has shown the effectiveness of embedded assessments in curricular activity sequences (Tanimoto *et al.*, 2002). We strive to include a myriad of activities that provide the student with tailored feedback. One instantiation of embedded assessments is a simple self assessment, in which students are presented with a problematic situation or question. They answer the question either by typing their answer or selecting an answer option. The students receive instant feedback and are directed to information that specifically addresses their assumed mental model. The self-test provides students a penalty-free opportunity for evaluating their understanding.

The quiz activity. Quizzes take place at the beginning of lab either once or twice a week and review content from the previous lab. The quizzes are graded; however, quiz scores are capped such that students receive all points for the quiz portion of their grade if they have an average score of 70% or greater across all quizzes. The goal is to provide students an incentive to keep up with the curriculum, while retaining the main focus of the quiz as feedback to the student rather than punishment for a lack of understanding. As with all students' submitted work, these quizzes help instructors identify patterns of misunderstandings and students that need additional support.

The project checkpoints. In each of our courses, students take on larger programming assignments in the form of multi-week projects. For each of these projects we provide time in lab for supervised project work and feedback from a TA. These sessions serve as an explicit opportunity to reflect on group functioning and collaboration (Heller & Hollabaugh, 1992) and attempt to reduce students' propensity to procrastinate. This scaffolding with intermediate project deadlines and feedback align with many of the findings in the literature on procrastination. For instance, Ross and Nisbett (1991) claim that structured situations can lessen procrastination, Wolters (2003) and Shaffer and Edwards (2007) recommend setting proximal goals for larger assignments, and Wesp (1986) finds specific benefits for daily quizzes.

Structuring and Scaffolding Programming

The case study activities. Many of the lab activities are structured for students to work with framework code as they gradually develop the skills necessary to design and program their own projects. Our lab-centric courses make use of case studies (Linn & Clancy, 1992) to provide models for applying design, development, and analysis skills. Each case study is a narrative describing the paths from a problem statement to one or more solutions. Individual students bring diverse perspectives to the case study and focus on different aspects of the narrative and accompanying code. Our lab-centric courses use additional activities to take advantage of this diversity, such as summarizing the case study and commenting on classmates' summaries. In the lab curriculum, students modify the provided code as well as engage in discussion around any design patterns presented in the case study.

The scaffolded fill-in the blank coding activity. We have tools so that students can fill in a segment of a program and have the results evaluated within the web browser. These scaffolded programming tasks focus student attention on the central aspect of the task, while hiding aspects of the code that might be overwhelming or inaccessible for students at their current level of experience. For example, instead of writing an entire function, students can provide the predicate to an existing if-else structure in an attempt to produce a particular output or a particular error. We have a tool that can evaluate arbitrary Scheme code in the browser, allowing activity authors to provide very tailored feedback. Another tool modifies a Java editor to present a file with editable regions and hidden instructor-provided evaluation code.

Experiences with lab-centric instruction

Overview of Lab-Centric Offerings

Our first attempt at a lab-centric course was in the summer of 2002 at University of California at Berkeley, for a CS1 course in Scheme. It was perceived as a great success (Clancy *et al.,* 2003). We have used the lab-centric format continuously for this course ever since, and it has been taught by four different instructors. In 2004 we piloted a lab-centric Java-based data structures course (CS2) at U.C. Berkeley. Designed to follow a Scheme-based programming course for majors, it included an introduction to the Java language. This course has been taught in the lab-centric format four times since the original offering, all by the same instructor. In 2007, we piloted a lab-centric version of our lower-division computer architecture course, including instruction in C and MIPS, and have since offered it three additional times, taught by two different instructors. We have also been involved with four different lab-centric CS1 offerings in Java for other departments and/or institutions either as consultants, curriculum designers, or instructors. However, data from these offerings have not been analyzed.

Hypotheses Regarding Lab-Centric Instruction

At the highest level, we wish to determine how students benefit from the lab-centric format. Below we present four more focused research questions followed by our rationale and context for each question.

1) Does lab-centric instruction improve student performance on exams and multiweek projects?
2) Are there self-selection effects between lab-centric and traditional course populations?
3) Does lab-centric instruction decrease student procrastination?
4) Are students in lab-centric courses less likely to perceive the pace of the course as too fast?

In line with our pedagogical goals, our primary research question examines whether lab-centric instruction can improve student learning. Below, we distinguish between exam and multi-week project scores. We hypothesize that lab-centric instruction helps students with both outcomes, but will help students more with project work, because the labs emphasize activities that are more relevant to project-based evaluations. We have seen trends suggesting such a relationship in earlier, less systematic analyses (Titterton, Clancy, & Lai, 2008). Through our experiences with lab-centric instruction we have developed theories about how lab-centric instruction might serve underrepresented populations in traditional courses and we sought to investigate if students appeared to self-select into lab-centric or traditional courses. Our third hypothesis, suggested by early interviews with students from CS1, is that the highly structured sequencing of the curricular activities will decrease student procrastination. Finally, we hypothesize that because of the greater level of scaffolding, students will be less likely to feel the pace of a lab-centric course is too fast than they would a traditionally formatted course.

Data sets and limitations

While establishing separate yet rigorously comparable lab-centric and traditional courses has been infeasible at our institution, below we provide details of the methods of comparison available within each of our lab-centric courses (Computer Architecture, CS2 and CS1), and discuss how we can evaluate the data to address our hypotheses. In our analyses, we will generally use responses from survey data in combination with two separate outcomes: a sum of a student's project scores, and a sum of a student's exam scores, under the assumption that projects and exams measure different aspects of student learning.

Computer Architecture - Fall 2007

Although we have taught Computer Architecture in the lab-centric format four times, the offering that most closely approaches our research ideal was that of fall 2007. That semester we offered a single lab-centric section (n = 21) and four traditional sections (n = 126). The course was co-taught, with one instructor giving the lectures and another instructor developing the labs and assessments for both formats. All students were assigned the same projects, exams, and homework. With this course structure, we are able to directly compare student performance

through a sum of scores on the three exams (combined exam) and sum of scores on the four projects (combined project). In addition, students in both formats completed three course surveys to measure demographic information, perception of course climate, and other information.

While this offering isolates aspects of lab-centric instruction, there are two limitations in this design. First, students self-selected either lab-centric or traditional as their course format. As we shall show, the student populations differed in several important ways. Second, students may have exposed themselves to the other format or interacted with students from the other format. For instance, some students enrolled in the lab-centric section attended some lectures, and some students enrolled in the traditional sections used the lab-centric materials in various ways.

Data Structures (CS2)

We analyze student performance and survey responses across the lab-centric and traditional versions of CS2 from the fall of 2004 through the spring of 2009. Altogether, we analyze records from a total of 939 students including three lab-centric offerings from one instructor (with 89, 141, and 133 students) and four traditional offerings from two different instructors (two offerings each, with 99, 101, 144, and 232 students). Technical problems with the course survey in the fall of 2008 traditional offering rendered it unusable.

Multiple surveys were given over the semester in each of the lab-centric offerings. In the traditional offerings, a much shorter survey was given at the end of the semester. Some of the survey items between the two formats were duplicated for comparison. The instructors for both versions of the course offered a small number of course points for taking the survey, and the response rates generally exceeded 90%.

Although the general syllabus for each offering was the same, the ordering and emphases of the materials differed. Additionally, each offering included qualitatively different projects and exams, although some of the projects were duplicated between offerings from the same instructor. For each student, we generated a sum of project scores and a sum of exam scores. These scores were converted into standardized scores based on the mean and standard deviation within that offering to adjust for differences in point scales and grading rubrics. As such, we are unable to use these scores to directly compare the lab-centric and traditional formats: the average score for each offering is 0, by definition. However, we are able to look at interactions between format (lab-centric and traditional) and one or more other variables (e.g., self-ratings of programming competence) on our outcomes. Based upon all of these limitations, we do not consider this data a random sample from a hypothetical population; consequently, we will report observational trends rather than probabilities from hypothesis testing. These explanatory, rather than confirmatory, analyses shed light on our hypotheses. However, they cannot be expected to firmly answer our research questions.

Introductory Programming (CS1)

We believe that our CS1 for non-majors is our best lab-centric course, as it has undergone the most refinement. It has been taught over twenty semesters, including summers, and students have been given extensive surveys in each offering to guide our refinement. Unfortunately, the current lab-centric format was adopted before baseline data was collected, and no traditional-format semesters have been offered since. As such, we are completely unable to compare the lab-centric format directly with anything. Consequently, our analyses can only examine what sorts of differences there are between groups of students (e.g., those that do well and those that do poorly, females versus males, and so forth).

Findings and Implications

Learning effects

We examine hypothesis 1— does lab-centric instruction improve student performance on exams and projects— through data from the fall of 2007 computer architecture course. As mentioned above, we are unable to examine hypothesis 1 directly through data in the CS2 offerings: with normalized outcomes, the means are all exactly zero. In the fall of 2007, a visual examination shows that the lab-centric group does slightly worse on the combined project score and slightly better on the combined exam score. However, t-tests fail to find any significant differences.

For the fall 2007 computer architecture course, we have several variables that can be used as covariates to help tighten the relationship between lab-centric membership and the exam and project scores. These include:

- Controlling for the other outcome: The project and exam combined scores are quite correlated (r=.656), meaning that there is a big part of exam score that project score will predict and 'remove' from the predictive model.
- Learning style: Students were asked about their experiences in prior courses, and whether they learned best in lab, lecture, or by themselves. The majority of students in the lab-based group believe they learn best in lab, while the non-lab-centric group are much more evenly split, as discussed in the section on self selection effects below. Response to this question has no significant relationship to project or exam scores.
- Realistic letter grade: Students were asked "Realistically, what grade do you expect to get in this course". Student response is highly associated with both exam and project outcomes, and, as discussed in the section on self selection effects below, is also associated with membership in the lab-centric or traditional sections.

A general linear model including main effects for the above factors (as well as lab-centric group membership) and a term for interaction between lab-centric group membership and lecture learning style shows a significant relationship between lab-centric membership and combined exam score ($F=4.65$, $p<.05$). Membership in the lab-based group is predicted to increase exam score by approximately 12 points. The mean of combined exam score is 81, with a standard deviation of around 19, giving an effect size of $d=0.63$ (using Cohen's d). The maximum and minimum combined exam scores are 114.4 and 35.5 respectively.

A similar model predicting combined project score shows very similar significance ($F=4.78$, $p<.05$). However, membership in the lab-centric group is associated with a *lower* project score by about 5.4 points. The mean of combined project score is 28.5 with a standard deviation of about 9, giving an effect size of $d=0.6$. The maximum and minimum combined project scores are 39.9 and 3 respectively.

We hypothesized that lab-centric instruction would help student scores on projects more than exams, given the type of interactive curriculum and skills we emphasize in the lab-centric format. Our hypothesis, then, was roundly rejected! We will continue to analyze these results more thoroughly. The combined scores for exams as well as for projects are likely hiding quite a bit of meaningful structure.

The nature of the computer architecture course may be a partial cause of these findings: in comparison to the CS1 course and, to a large extent, the CS2 course, the computer architecture course is a series of small, independent units that don't much build upon each other. There is no "final" project, for instance. These smaller assignments may not have benefited from the lab-centric format as much as an extended project might.

It is also important to note that although the lab instructor was experienced in lab-centric course development, students in the lab-centric group were working through first-generation materials that have been refined over subsequent semesters.

Data from the CS2 offerings can also be used to predict the effect of lab-based group membership on outcomes when controlling for other variables. There are, however, far fewer covariates that can be included in the model. As mentioned above, we wish to avoid complicated confirmatory analyses with the CS2 dataset. Any result would be *very* difficult to interpret. With the covariates specified above, excluding the realistic letter grade variable (which wasn't asked in the surveys of all the CS2 offerings), lab-centric membership again reduces project score and raises exam score, validating the trends shown in the fall of 2007 offering of the computer architecture course.

Self-selection effects

As discussed above, students self-selected their membership in all courses. However, this selection may be based on factors that contribute to the outcomes measured. In this section, we present evidence that the samples of students are different in several ways.

Learning style

In the initial survey of the computer architecture course in the fall of 2007, students were asked about their learning style in prior lecture courses. Table 2 details the question and the student responses. There is a strong relationship between responses and lab-centric group membership (X^2=13.6, p<.005). In general, students who choose to attend the lab-centric format largely believe they learn best in lab and discussion, while students in the traditionally format course were more evenly spread. This question was not asked in our CS2 surveys.

In university lecture courses that you have taken prior to this semester, do you feel that you generally learned more:	Lecture	Lab / Discussion	While studying by myself or in a group	There are no general patterns
Non-lab centric (n=97)	15.5% (15)	30.9% (30)	16.5% (16)	37.1% (36)
Lab based (n=21)	5.8% (1)	66.7% (14)	5.8% (1)	23.8% (5)
Total (n=118)	13.6% (16)	37.3% (44)	14.4% (17)	34.7% (41)

Table 2. Results from fall 2007 for the survey question on lecture learning style.

Reluctance to ask questions

In the same initial survey of the computer architecture course in the fall of 2007 we asked several questions related to students' perception of the "climate" of university classrooms. These questions, shown in Table 3, ask students about their reluctance to ask questions in other courses and why they might feel that way. Because these questions do not ask about the current course, differences in responses between students in the lab-centric and traditional course populations represent differences students bring to the computer architecture course.

1	In discussion sections in other courses, how often are you reluctant to ask a question because you are nervous about speaking in front of other students?
2	In discussion sections in other courses, how often are you reluctant to ask a question because you don't want to waste the time of the [teaching assistant] or other students?
3	In discussion sections in other courses, how often are you reluctant to ask a question because you are afraid your classmates will think you are stupid?
Responses	Almost always; Most of the time; Sometimes; Rarely/Never; I don't know.

Table 3. Example of classroom climate questions in course surveys.

Table 4 shows responses for question 2; the distribution of responses for the other questions is similar. Responses are not randomly distributed (X^2=173, p<.001); rather, students in the lab-centric version are more likely to be reluctant to ask questions. In the combined CS2 courses, the distribution of responses is similar to that shown in Table 4, albeit with less difference between the formats and a much larger sample size.

Reluctance to ask questions in order not to waste time	Rarely	Sometimes	Most of the time	Almost always
Non-lab centric (n=101)	59.4% (60)	28.7% (29)	9.9% (10)	2.0% (2)
Lab based (n=18)	33.3% (6)	50.0% (9)	16.7% (3)	0% (0)
Total (n=119)	55.5% (66)	31.9% (38)	10.9% (13)	1.7% (2)

Table 4. Results from fall 2007 for the survey question "In discussion sections in other courses, how often are you reluctant to ask a question because you don't want to waste the time of the [teaching assistant] or other students?"

Students who are more reluctant to ask questions in these ways are disproportionately choosing the lab-centric format over a traditional format.

Expected grades

Students in the fall 2007 computer architecture course were asked at the beginning of the semester "Realistically, what grade do you expect to get in this course". Answers are shown in Table 5. Students in the lab-centric course have far lower expectations for their grade (X^2=7.6, p<.05). Note that there is a highly significant relationship between expected grade and the project and exam outcomes: the higher one's expected grade, the higher the project and exam score. Therefore, the students entering the lab-centric section believe they are less likely to do well in the course, and this belief is, in general, associated with lower course grades. Students in traditionally formatted CS2 offerings were not asked this question, making analysis of those offerings impossible.

Expected grade in course	A+/-	B+/-	C+/-
Non-lab centric (n=96)	75.0% (72)	22.9% (22)	2.1% (2)
Lab based (n=21)	47.6% (10)	52.4% (11)	0% (0)
Total (n=117)	70.1% (82)	28.2% (33)	1.7% (2)

Table 5. Responses in fall 2007 computer architecture offering to survey question "Realistically, what grade do you expect to get in this course?"

We need to do further research on why a lab-centric course selects for these characteristics. This may be a general trend, or it may be unique to the student body at institutions like ours. In early pilot studies of our CS1 for non-major offerings, we believed that students who were likely to drop out, fail, or struggle in a traditionally formatted course were doing better than that in our lab-centric version. That is, it seemed that the lab-centric format was serving the most at-risk students better than a traditional course. Without accurately tracking students that drop courses, however, we are unable to analyze this hypothesis.

However, if less able students perceive lab-centric courses as more accessible and supportive, this may be a reason that they disproportionately enroll in them. It is also important to note that some of the characteristics of these self-selected samples suggest that lab-centric offerings will have worse outcomes than traditional offerings, all things being equal. The analyses in the section above on learning effects bear this out for exam outcomes. That is, when controlling for these self-selection differences in the larger linear model, the lab-centric students relative exam scores rose. However, the opposite was shown for project scores.

Differences in procrastination ratings

We hypothesized that the lab-centric format would have a limiting effect on student procrastination, for several reasons. These include its more visible content outline and timeline, daily quizzes and frequent opportunities for self-assessment, plentiful closed-lab hours, smaller increases in complexity between activities, and the close monitoring of student progress on projects.

Interviews in our CS1 for non-majors support this. In spring 2006, several students commented that the detailed schedule helped them keep up with their work. In spring 2009, one student commented on the long hours in lab: "So, it's actually kind of nice for me, because it forces me to sit down and do all the homework, instead of just - oh I'll get to it eventually, then do it, you know, the night before really fast, it just gives me time to do it."

In the fall 2007 computer architecture course, we asked students whether procrastination caused them to get started late and/or do less work on either the projects or exams, as shown in Table 6. While there was an essentially uniform spread in responses, we found no significant difference between the formats on procrastination. We did confirm that procrastination had a strong effect on relevant grade outcomes, with the students who reported procrastinating more doing less well ($p<.01$ for 3 of the four questions, $p<.05$ for whether procrastination caused them to be late studying for exams).

1	Because of procrastination, did you get started late on the project(s)?
2	Because of procrastination, did you do less work on the project(s) than you would otherwise have done?
3	Because of procrastination, did you get started late studying for the exams?
4	Because of procrastination, did you study less for the exams than you would have otherwise?
Responses	Yes; For some of them; No, not for the exams/projects; I didn't really procrastinate in this class; I don't know/can't say.

Table 6. Procrastination questions in course surveys.

In the combined CS2 offerings, the relationship of procrastination to format and grade outcomes was very similar to above: students that reported more procrastination did more poorly on the relevant outcome, and, between the lab-centric and traditional formats, students admitted to procrastinating about equally.

Our hypothesis, then, is rejected: we can't find any effect of the lab-centric format on self-reports of procrastination. However, we do think that further research is warranted into exploring the link between procrastination and the lab-centric format. A better way of measuring student procrastination is important, rather than relying on a self-rating at the end of the semester. Edwards *et al.* (2009), for instance, uses submit times to an autograder to track student work. This is especially important if students choose their course format—that is, self-select—based on their tendencies to procrastinate. It may also be that any effect will be different in the different courses and/or student populations, implying that our observations in CS1 and results in CS2 and Computer Architecture may not be at odds.

Differences in ratings of pacing

We hypothesized that students in the lab-centric course would perceive the pace as slower, or at least not too fast, because of the greater level of scaffolding and support. Students answered a survey question at the end of the course asking them whether they agreed with the statement "The pace of the course is too fast", providing a rating from 1 (not at all true) to 5 (very true).

Students in traditionally formatted courses of CS2 rated the pace as faster than students in the lab-centric courses ($t=3.30$; $p=0.001$), with a overall mean difference of 0.29 points. However, who the instructor was played a very large part in ratings of pace in the CS2 courses. Since we never varied instructor by course format in CS2—that is, one instructor taught all the lab-centric offerings while two other instructors taught the traditional offerings—we are unable to discount this as a factor in our results.

Students in the fall 2007 computer architecture course did not differ significantly in their mean rating of pace. The different results between the two courses may be due to curricular flow. As mentioned earlier, the computer architecture course is a series of small, independent units that don't build much upon each other, while the CS2 courses build towards a large project.

Conclusions

We have described what we call a lab-centric instructional format, which starts by trading lecture and discussion time for supervised time in a closed lab completing activities delivered online. The increased lab time allows for a variety of activities over and above what students do in

traditional courses. In lab-centric versions of three of our lower-division computer science courses, we include not only programming-related exercises but also embedded assessments, supports for reflection and self-monitoring, and offline as well as online collaboration.

The increased lab time potentially provides several benefits:

- replacement of passive learning in lecture by active learning in lab;
- arrangement of activities to minimize the gradient between easy and difficult concepts;
- increased awareness by course staff of how well individual students are doing;
- more and timelier staff availability (because of reallocation of contact hours) for help throughout the course; and
- development of a comfortable classroom climate that supports collaborative learning.

More benefits can result from an appropriate online delivery vehicle:

- rapid feedback to lab instructors about student misunderstanding, and correspondingly timely engaging with the student in targeted tutoring to clear up confusion;
- more opportunities for instructors to make pedagogical contributions other than lecturing.

There are potential disadvantages as well:

- the challenge of curriculum construction;
- space limitations; and
- staff scheduling constraints.

Evidence suggests that we have largely achieved the benefits and avoided the pitfalls, although much of this evidence precludes formal analyses. In fall 2007, however, students in a lab-centric section of a course were given the same exams, homework, and assignments as students in traditionally formatted sections. Comparisons showed that

- students in the lab-centric section had lower grade expectations at the start of the semester and a greater reluctance to ask questions during lab;
- when controlling for these differences, students in the lab-centric section did significantly better on the exams but significantly worse on the projects; and
- there was no difference in self reports of procrastination between students in the two sections, and limited evidence that students were more comfortable with the pace of the lab-centric section.

We continue to focus research and development effort on lab-centric instruction. For instance, we plan on performing analyses of our extensive logging data to better understand how students view and interact with the activities in our courses and, we hope, gain further insight into which characteristics and behaviors successful students exhibit. We also hope to look at student progress across several courses to examine whether students benefit from early lab-centric instruction in later courses or, perhaps, do worse in later courses when the lab-centric support structures are removed.

Additionally, in order to ease the challenge of curriculum construction we are designing an authoring system and instructor community portal to enable sharing and collaborative development of curricular materials. If you are interested in viewing and/or using our curricular materials or tools, contact us via email or go to http://ucwise.berkeley.edu/info, where we will continue to provide updated access to our research, curriculum, and tools.

Acknowledgments

Work described in this paper was supported by the CITRIS project (Center for Information Technology in the Interest of Society), Hewlett-Packard Corporation, and National Science Foundation grants DUE-0443121 and CNS-0722339. We also wish to acknowledge the support of Marcia Linn and Jim Slotta (directors of the WISE

project) and Lois Wei (chief architect of our original online delivery system). Finally, we gratefully recognize the research and implementation contributions of numerous students and colleagues associated with our project over the past nine years. Suggestions by the reviewers improved this paper significantly.

References

Barker, L.J., Garvin-Doxas, K., & Jackson, M. (2002). "Defensive Climate in the Computer Science Classroom", *ACM SIGCSE Bulletin*, 34(1), 43-47.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher.* 13, 4-16.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.). (1999). *How People Learn: Brain, Mind, Experience, and School*. Washington, DC: National Research Council.

Carle, A., Canny, J., & Clancy, M., (2006). PACT: A pattern-annotated course tool. *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2006 (1), 2054-2060.

Carle, A., Clancy, M., & Canny, J. (2007). Working with pedagogical patterns in PACT: Initial applications and observations. *ACM SIGCSE Bulletin*, 38 (1), 238-242.

Chen, Z. & Klahr, D. (1999). All other things being equal: Acquisition and transfer of the control of variables strategy. *Child Development*. 70(5), 1098-1120.

Chi, M. T. H., de Leeuw, N., Chiu, M. H., & La Vancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science*. 18 (3), 439-477.

Clancy, M.J., Titterton, N., Ryan, C., Slotta, J.D., & Linn, M.C. (2003). "New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course", *ACM SIGCSE Bulletin*, 35(1), 132–136.

Cohen, P. A., Kulik, J. A., & Kulik, C. L. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal*, 19, 237-248.

Collins, A., Brown, J.S., & Newman, S.E. (1989). "Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics". In L.B. Resnick (Ed.), *Cognition and Instruction: Issues and Agendas*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Davis, E.A., Linn, M.C., & Clancy, M. (1995). "Learning to use parentheses and quotes in LISP." K. Barker (Ed.), *Computer Science Education*, 6 (1), 15-31. Norwood, NJ: Ablex.

diSessa, A. and Minstrell, J. (1998). Cultivating Conceptual Change with Benchmark Lessons, In J. Greeno and S. Goldman (Eds.), *Thinking Practices in Mathematics and Science Learning.* Mahwah, NJ: Lawrence Erlbaum Associates.

Dougiamas, M. & Taylor, P. (2003). Moodle: Using Learning Communities to Create an Open Source Course Management System. *ED-MEDIA 2003*.

Dubinsky, E., Mathews, D., & Reynolds, B.E. (Eds.) (1997). *Readings in Cooperative Learning for Undergraduate Mathematics*, MAA Notes 44. Washington, DC: Mathematical Association of America.

Edwards, S.H., Snyder, J., Pérez-Quiñones, M.A., Allevato, A., Kim, D., Tretola, B., (2009). Comparing Effective and Ineffective Behaviors of Student Programmers. *ICER 2009*, 3-14.

Elbe, K.E. (1976). *The Craft of Teaching*. San Francisco: Jossey-Bass Publishers.

Heller, P. & Hollabaugh, M. (1992). Teaching problem solving through cooperative grouping, part 2: designing problems and structuring groups. *American Journal of Physics*, 60(7), 637-644.

Hoadley, C.M. (2004). "Fostering Productive Collaboration Offline and Online: Learning from Each Other". In *Internet Environments for Science Education*, M.C. Davis, & P.L. Bell (Eds.). Mahwah, NJ: Lawrence Erlbaum Associates.

Hofstein, A. & Lunetta, V.N. (2003). "The Laboratory in Science Education: Foundations for the Twenty-First Century," *Science Education*, 8(1), 28-54.

Johnson, D.W., Johnson, R.T., & Smith, K.A. (1991). *Cooperative Learning: Increasing College Faculty Instructional Productivity*, ASHE-ERIC Higher Education Report No. 4, Washington DC: The George Washington University.

Koshman, T. (Ed.). (1996). *CSCL: Theory and practice of an emerging paradigm*. Mahwah, NJ: Lawrence Erlbaum Associates.

Koshman, T., Hall, R., & Miyake, N. (Eds.). (2002). *CSCL 2: Carrying forward the conversation*. Mahwah, NJ: Lawrence Erlbaum Associates.

Laurillard, D. (1993) *Rethinking University Teaching*, Routledge/Falmer, London: (1993).

Leventhal, L.M., Teasley, B., & Rohlman, D.S. (1994). Analyses of factors related to positive test bias in software testing. *International Journal of Human-Computer Studies*, 41(5), 714-749.

Linn, M.C. & Clancy, M.J. (1992). The case for case studies of programming problems. *Communications of the ACM*, 35(3), 121-132, March 1992.

Linn, M. C., Davis, E.A., & Bell, P. Eds. (2004). Internet Environments for Science Education. Mahwah, NJ, Lawrence Erlbaum Associates.

Linn, M. & Kessel, C. (2003). Gender differences in cognition and educational performance. In L. Nadel (Ed.), *The Encyclopedia of Cognitive Science* (Vol. 2, pp. 261-267). New York: Macmillan.

Mazur, E. (1997). *Peer instruction: a user's manual.* Upper Saddle River, NJ: Prentice Hall.

McCauley, R. & Manaris, B. (2002) "Report on the Annual Survey of Departments Offering CSAC/CSAB-Accredited Computer Science Degree Programs", http://www.cs.cofc.edu/~mccauley/survey/.

McCauley, R., Pharr, W., Pothering, G., & Starr, C. (2004). "A Proposal to Evaluate the Effectiveness of Closed Laboratories in the Computer Science Curriculum," *Journal of Computing Sciences in Colleges*, 19 (3), 191-198.

McCauley, R., Starr, C., Pharr, W., Stalvey, R., & Pothering, G. (2006). "Is CS 1 Better with the Same Lecture and Lab Instructor?", *ACM SIGCSE Bulletin*, 38 (2), 54-60.

McKeachie, W. J. (2005). *Teaching Tips* (12th edition). Boston, MA: Houghton Mifflin.

Parker, J., Cupper, R., Kelemen, C., Molnar, D., & Scragg, G. (1990). "Laboratories in the Computer Science Curriculum", *Computer Science Education*, 1, 205-221.

Ross, L. & Nisbett, R. (1991). *The person and the situation.* New York: McGraw-Hill.

Ryan, C. (2006). "Analogies Are Like Bowling Balls, or Why Analogies to English Need Some Explanation to Help Students Learn Scheme", Technical Report No. UCB/EECS-2006-75, Berkeley, CA: EECS Department, University of California.

Shaffer, C.A. & Edwards, S.H. (2006). Scheduling, Pair Programming and Student Programming Assignment Performance. *IEEE Transactions on Education*, 1 (11), 1-6.

Slavin, E. (1995). *Cooperative learning: Theory, research, and practice* (2nd ed.). Boston: Allyn and Bacon.

Slotta, J.D. & Linn, M.C. (2009). *WISE Science*. New York: Teachers College.

Soloway, E. & Spohrer, J. (Eds.) 1988. *Studying the novice programmer*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1988

Tanimoto, S., Carlson, A., Husted, J., Hunt, E., Larsson, J., Madigan, D., & Minstrell, J. (2002). *Text Forum Features for Small Group Discussions with Facet-Based Pedagogy.* Paper presented at the Computer Supported Collaborative Learning 2002, Boulder, Colorado.

Titterton, N. & Clancy, M. (2007). "Adding some lab time is good, adding more must be better: the benefits and barriers to lab-centric courses", presented at the 2007 International Conference on Frontiers in Education: Computer Science (FECS'07: June 25-28, 2007).

Titterton, N., Clancy, M., & Lai, T. (2008). "Experiences with a Lab-Centric CS 2." Poster presented at SIGCSE 2008, Portland, OR, March 12, 2008.

Urban-Lurain, M. & Weinshank, D. (1999). "'I Do and I Understand:' Mastery Model Learning for a Large Non-Major Course." *ACM SIGCSE Bulletin*, 31(1), 150-154.

Wesp, R. (1986). Reducing Procrastination Through Required Course Involvement. *Teaching of Psychology*, 13 (3), 128-130.

Williams, L. & Upchurch, R.L. (2001). In Support of Student Pair-Programming. *SIGCSE 2001.* 327-331.

Wolters, C.A. (2003). Understanding procrastination from a self-regulated learning perspective. *Journal of Educational Psychology*, 95(1), 179-187.

Zweben, S., Stringfellow, C., & Barton, R. (1989). Exploratory studies of the software testing methods used by novice programmers. Paper presented at the Software Engineering Education Conference, Pittsburgh, Pennsylvania. 169-188.