Closed Labs in Programming Courses: A Review

Radu P. Mihail and Krishnendu Roy
Department of Mathematics and Computer Science
Valdosta State University
1500 N Patterson St.
Valdosta, GA
{rpmihail, kroy}@valdosta.edu

ABSTRACT

Recent trends show nationwide increases in undergraduate CS enrollment. Retention of students in CS programs continues to be an area with room for improvement. Research to increase retention focuses on modifications in the introductory computer science courses (CS1 and CS2). Many CS1/2 courses have a laboratory component. In these labs students typically write code, most often under the supervision of the instructor or a lab assistant. While labs are perceived by many CS instructors to be essential for introductory CS courses, we ask the following question: what characteristics of labs help increase the percentage of students with satisfactory learning outcomes in introductory CS courses? Based on the vast amount of evidence showing that retention is higher when students are more successful in the introductory level classes, we survey existing CS education literature that address CS1/2 closed labs. In this paper we present what we learned about common practices in the design and implementation of CS1/2 labs.

Keywords

CS1, CS2, Introductory Programming, Closed Laboratories, Design, Theory, Human Factors, Languages, Measurement

1. INTRODUCTION

The Association for Computing Machinery (ACM) released in 2013 the latest curriculum guidelines for undergraduate programs in computer science. In part because not all topics relevant to a computer scientist can be taught in a two or even a three course sequence, many departments have to make choices that involve trade-offs and potential loss of exposure to topics. This multidimensional design space leads to hot debates regarding the relative importance of a set of learning outcomes and sparse coverage due to lack of time.

In spite of this debate, the vast majority of undergraduate programs offer, at some point in their curriculum, one or more courses where concepts in computer science are taught using a programming language. In most cases, students are exposed for the first time to syntax and core concepts, and these courses are the first in a sequence of two or more, which we call introductory programming courses. Introductory programming courses typically have a laboratory component, where students can practice programming.

Lab components of introductory programming courses come in two main flavors: open and closed labs. Open labs offer students a space to work, mostly not constrained by time (i.e., open most of the day) and staffed by tutors or teaching assistants. The courses using open labs may encourage students to work on the homework in these labs, but attendance is not required. Closed laboratories consist of scheduled class time, supervised by an instructor or a teaching assistant. In these sessions, the students are given concrete instructions and their work is graded. The instructor is there to provide help in the form of direct instruction.

In this paper, we review existing literature in an attempt to summarize current practices in closed labs. We conjecture that retention is highly dependent on a constructive and positive experience with introductory programming courses served by closed labs. Although a vast amount of universities offer introductory programming courses, closed laboratory assignments and their learning outcomes are largely a matter of local choices.

More specifically, we ask the following questions:

1. What are the characteristics of an effective lab?
2. What are the assessment methodologies employed effective closed labs?
3. What types of lab design choices result in effective introductory programming closed labs?
4. What are the benefits and harms of contextualization (i.e., focusing on a specific application domain such as game development or software engineering)?
5. How do teams play a role in student learning?
6. How does the size of the university affect the use of labs?

In this paper we reviewed publications in the ACM Digital Library, IEEE Explore, Journal of Computer Science in Colleges, and Taylor and Francis CS Education Journal related to closed labs. In Sections $2 - 6$ we present a summary of research answering the questions posed above.

2. ASSESSING EFFECTIVE LABS

In this section we present details about how various closed lab related publications reported their assessment methodology.

Closed lab sessions can be measured with respect to retention, performance on exams and homework assignments and self reported measures. The literature that directly addresses lab efficacy is relatively sparse. Several studies include discussions on benefits of labs in CS1 courses. However, not many include rigorous assessment of student learning in closed labs. We believe (as also briefly alluded to in [31]) that one of the reasons assessment is difficult is simultaneous changes for all sections. Therefore when labs are introduced, there are no non-lab sections that can be used as a baseline for comparison. Researchers have to fall back on data from previous offerings of the course.

Titterton et al. [30] present their experience with offering a "lab centric" course. They describe common problems with the adoption a lab centric approach, such as shortage of facilities, staff and the determination of teaching credit. Kumar et al. [14] compare student learning and retention in a course with closed labs to that with open labs. Their result shows that there was no significant gain in student retention from the open lab to the closed lab course. Their research did show some improvement in test scores in the closed lab section, compared to the open labs. The authors also found similar results in the context of closed labs and online testing. These results were reported in [15].

Titterton et al. [31] used surveys and analyzed student performance in lab-centric and non-lab versions of the same course. They did this in two different courses (CS2 - Data Structures, and Computer Architecture). They found that lab-centric courses helped students increase exam scores. They also found conflicting evidence that students in lab-centric courses did worse on homework projects.

McCauley at al. [18] proposes an experimental design to evaluate the effectiveness of closed laboratories. This design accounted for reports that different instructors for lecture and lab make a difference. In a follow-up study by McCauley et al. [19], they reported no effect on learning outcomes given different lecture and lab instructors, but significant increased student satisfaction was observed when the lecture and lab were taught by the same instructor.

Soh et al. [26] measured the effects of closed labs on students' performance on placement exams, take home assignments, end of course exams, and attitude towards CS1 course. They included pre- and post-tests in their labs and analyzed student performance. They found students' lab performance was a significant predictor of exam outcomes as well as homework assignment scores. The lab performance was not a significant predictor of their placement exams. The same authors published subsequent results in Soh et al. [25] where they found positive correlation between lab performance and exams/homework-assignments.

Horton et al [10] compared and contrasted traditional and flipped CS1. They report better learning outcomes as measured by the final exam, but no changes in student attitudes or likelihood to pass was observed.

2.1 Conclusions on Lab Characteristics

In this section we reviewed the literature to answer the question: what are the assessment methodologies employed effective closed labs? We found no consensus on how to assess the effectiveness of closed labs. When formal assessment was done, student retention, their grades on exams, and end of course surveys were some of the assessment metrics. There was no discussion on whether any of these methods provide better insights over others. Many papers use anecdotal and experiential (here is what we tried and here is what happened) data to support the positive effects of their labs.

3. LAB DESIGN CONSIDERATIONS

The variety of curriculum choices in computer science programs result in a diverse set of structural designs for closed labs. We found that there is insufficient research to determine which types of lab designs result in effective closed labs. We conjecture that effectiveness of closed labs may be influened by local contraints (e.g., university resources and student-faculty ratio). In this section, we review historical uses of labs and the methods for evaluating lab design to support our finding.

Beginning around the late 80s to early 90s, when microcomputers became financially accessible to many universities, many researchers acknowledged that programming labs are useful, and presented their own recipes for success. Examples of early work are [3, 28]. Two decades later, works that address programming labs are similar, in that they rarely offer a comparison to other approaches. Unfortunately, there is no clear evidence that 20 years have produced any measurable progress in terms of student achievement following closed lab courses. One possible way to measure improvement over time is to evaluate designs by separating domain specific and instructional design choices.

A framework provides a conceptual model that allows the separation of domain specific and instructional design choices in the overall design and administration of labs. Separation of these choices is non-trivial. Consider the size of a CS department and the type of institution it is home to. Large departments in research institutions are more likely to hire an instructor who is responsible for large lectures and coordinating teaching assistants who administrate labs (open and/or closed). In small departments, introductory courses are typically taught in a rotating fashion by tenure-track faculty who have their own notions of optimal choices of content and instructional design.

Two decades ago, Waller [33] proposed a framework for the development and administration of lab materials in CS1 and CS2 courses. His system allowed for the incorporation of extant materials, but provided no explicit instructional design research. Kumar [14] compared open and closed labs with respect to student retention, performance and project completion. They found that closed lab did not improve retention or project-completion, but observed qualitative improvement in student learning for closed labs in the first part of the course.

Soh et al. [25] present a framework for closed laboratories where instructional design is separated from content design. They embed instructional design features as well as an infrastructure to keep track of resources over time, in light of constantly changing software development and information technology. Their proposed framework consists of a lab design phase, in which CS faculty and researchers discuss and select CS topics. After this discussion, labs are implemented as stand-alone modules, each with handouts, worksheets, and pre- and post-tests. Instructional research questions can be answered by making modifications at the implementation stage. Deployment of labs is done in collaboration with course and lab instructors, who also monitor the labs, suggest fixes to problems and provide additional help to students.

3.1 Lab Design Conclusions

We reviewed literature to answer the following question: what types of lab design choices result in effective introductory programming closed labs? There is insufficient evidence to determine which lab designs lead to effective closed labs. Closed labs have been designed to fit the specific needs of a local university and thus are unique to each university (or instructor). Furthermore, case studies of labs address and do a great job of discussing implementation choices, but lack a framework that separates instructional design and pedagogical research. We recommend that researchers use a design framework (such as described by Soh et al. [25]) when designing and evaluating labs. We then hope that a set of consensus of best practices can emerge using a hypothesis driven research methodology that separates content from delivery.

4. CONTEXTUALIZATION

The topics taught in introductory programming courses can easily be grounded in applications (e.g. video games) to make the learning experience more motivating and to inspire students early in the program. We refer to the use of such applications as contextualization. We review the literature and discuss approaches that use games and real-world scenarios. The literature indicates that contextualizing introductory programming courses is beneficial for retention and student learning. Furthermore, there is evidence that lab effectiveness in contextualized courses is increased, but generalizing is premature due to limited sample sizes.

4.1 Games

Most CS faculty were told by some of their students that the students' choice of major was driven by video games and desire to write their own. This has motivated work on creating game-centric introductory programming courses, where labs are framed around the various aspects of game development.

Many game-centric approaches have been proposed [2, 4, 20], and the ones specifically aimed at CS1/2 courses [16, 23, 11, 5, 29] tend to focus on the course as a whole, and not on the specifics of the laboratory experience. This body of literature presents evidence that supports the hypothesis that using games as contextualization improves student learning outcomes and retention.

Chen et al. [4] report their experience with a game-centric course. Their labs were incremental, as object oriented programming concepts were introduced as a means to deal with the growing complexity of implementing the functionality of a simple flying-shooting game. The labs were supported by a framework library called *Game* that handles the game loop, windowing, controls and displaying graphics. The authors discuss the potential pitfalls with such an approach. For example, students might spend too much time on art or they might reduce functionality of the game to meet their programming abilities. The format of their lab sessions was closed, three hour sessions with two student groups. The authors reported that the lab time was not sufficient, but support from TAs and office hours alleviated the problem.

4.2 Real-World

Real-world scenarios are another form of contextualization. We live in the age of of massive data collection and analysis. Real-world scenarios can motivate students by showing them how to use skills learned in the classroom to analyze data and solve current problems.

Efforts to make introductory programming more relevant include using data from other domains. For example, Anderson et al. [1] use publicly available data from oceanography, DNA, elections, etc. They reported success as defined by the ability of students in CS1 to do real-world data analysis, but no quantitiative data was given.

Reilly et al. [24] discuss the introduction of new CS1 labs based on real-world problems. They reported more students completing the new labs compared to their status-quo labs. The researchers believe students did better on exams, but have no way to confirm due to the experimental design.

4.3 Interactive Multimedia

Another form of contextualization consists of incorporating digital media (images, sounds, and video clips) manipulations in introductory programming courses. Media Computation, is perhaps the most widely adopted example of media focused contextualization. In Media Computation students manipulate image pixels to create various effects and filtering. They also work with video and sound editing through code. Guzdial [8] describes his work over ten years on a media-centric introductory programming course. We refer the interested reader to read [8] for a comprehensive guide to media computation related literature.

4.4 Conclusions about Benefits and Pitfalls of Contextualization

We reviewed the literature to answer the following question: what are the potential benefits and pitfalls of contextualization? Given the success of using contextualization in courses as a whole as we discussed above, we believe that contextualization in labs has many benefits. Perhaps the greatest benefit is increasing student motivation to learn computer programming as we discussed in 4.2. Furthermore, the literature on real-world contextualization found that more students completed labs when the labs incorporated real-world problems.

While using contextualization in labs has benefits, there are also pitfalls to consider. One possibility is that using contextualization could increase workload for instructors. Chen et al. found that students needed more TA support and office hours to overcome insufficient time to complete games during labs. When planning to use contextualization, instructors need to be able to provide more student support. Additionally, creating the game assignments or finding real-world data may increase the time needed to plan lesson plans for labs.

Another common concern with contextualization is whether students can easily bridge the gap between contextualized knowledge and more general decontextualized knowledge that they will acquire in various CS classes beyond CS1/2. Guzdial addressed this difficult question in [7].

5. TEAMS

Introductory programming courses often include active learning activities like pair programming, group assignments, peer instruction etc., with the goal of fostering students' engagement and participation. We expected similar approaches to be broadly adopted in closed labs, but found few studies that provided details about active learning activities in the context of closed labs. In this section we present our findings.

Perhaps the most common intervention used in closed labs is collaboration where students are asked to work on an assignment together. Pair programming, where one student is acting as the driver and the other as the navigator, is a common form of collaboration. Walker [32] shows examples of closed lab implementation with collaboration. He describes a lab-centric course where most of the classes were labs, not lectures.

Nagappan et al. [21] and Williams et al. [34] present details about the effectiveness of pair programming in CS1 labs. Their results show that pair programming has positive effects on students' self-sufficiency, performance on tests and projects, and retention. Porter et al. [22] discuss efficacy of pair programming in their open as well as closed labs and how that increased retention.

Titterton et al. [31] discuss gated collaboration. In gated collaboration, during a lab, a question is posed to students. After a student answers the question (usually online), they get to view other students' answers and discuss various answers.

5.1 Conclusions about Teams

We reviewed the literature to answer the following question: how do teams play a role in student learning? There is some evidence that teamwork plays an important role in lab efficacy. Pair programming is the most common approach to implement teamwork and collaborative learning. The Community of Inquiry framework (widely adoped in the education community), proposed by Garrison et al. [6] proposes three "presences" for effective learning: teaching, cognitive and social. Pair programming falls under the social presence by facilitating engagement with peers and discourse relevant to student learning. The relatively limited number of publications related specifically to closed-lab student-engagements interventions leads us believe that this will be an active research area in near future.

6. OTHER CONSIDERATIONS

Closed laboratory for introductory programming appears to be the most widely adopted style by universities across the country and world. Despite their popularity, few studies with solid instructional and research designs have been done to assess best practices in a data-driven manner. A decade ago, Soh et al. [25] proposed a design framework for labs that incorporates instructional research. Their follow-up work [27], has been cited 3 times (excluding self-citations). In much greater numbers, case studies seem to dominate the literature regarding closed labs.

We observed that a subset of recent work within the introductory programming realm report shifting focus toward more dynamic (i.e., more stimuli) teaching tools (e.g., game-centric). There is a growing number of studies outside of the CS education community that indicate benefits of serious game-based approaches (e.g., [17, 13, 12]). Within CS education, controlled trials rarely have large ($n \geq 100$) samples sizes. We believe game-centered approaches have great potential, but support from data is imperative for further progress.

We often encounter distracted students, who, in spite of our best efforts, lag behind others who are more focused. Anecdotally, attention span is reduced today. We have not found any work in the CS education literature that takes a critical look at how distracting factors (e.g., cell phones, social media) impact learning. This problem is most acute when both the lecture and closed laboratory are in a room full of social-media-capable machines. We believe we need to take a closer look at course time allotment and be aware of attention span. Hakimzadeh et al. [9] took an encouraging step in this direction by proposing short 10-15 minute lecture sessions followed by individual learning activities, group analysis, peer learning, pair programming and discussion.

We believe closed laboratories should primarily contribute to maximizing the percentage of students' with satisfactory learning outcomes. A multi-dimensional design space and a continually evolving toolbox makes it difficult to establish a set of best practices. We believe that evidence-based controlled trials will eventually converge to a set of best practices, invariant to teaching technologies, but we have much work to do.

7. WHAT DID WE LEARN?

The main motivation of this work is to identify the best practices for success in CS1/2 closed labs. The main question we posed was: what are the characteristics of an effective lab? We surveyed five aspects of closed labs to answer this broad question. They were broad design frameworks, assessment of effectiveness, role of teams, contextualization, and local constraints like university size.

We found that there is insufficient evidence to conclude about any specific lab design framework being superior. We found a few ways to design closed labs. Some of the frameworks embedded instructional design aspects, while others did not. Assessing the effectiveness of a specific closed lab design framework is challenging in itself. We do not believe comparing design frameworks to decide the superiority of a framework is practical.

Assessing effectiveness of closed lab is another area where we found limited results from evidence-based assessments. There is no standardization about what is considered effective and metrics for measuring effectiveness. While some papers measured student retention as a metric for success, others used qualitative data from surveys.

There is some consensus about positive effects of contextualization in closed labs. Research shows benefits of various types of contextualizations in CS1/2 closed labs. We believe we will continue to see more contextualized CS1/2 courses and labs in future which will branch out to newer domains like cyber security, mobile computing etc.

Just like contextualization, there is consensus about the role of team work in increasing student success in CS1/2 labs. Pair programming was shown to work in many studies with convincing quantitative data. Other team oriented interventions like gated collaboration have also been shown to work.

To summarize, we can conclude that incorporating contextualization and teamwork related activities helps CS1/2 closed labs in several metrics. While there is some research to design effective frameworks that can be easily adopted, there is no broadly adopted framework. Consensus guidelines for assessing the success of closed lab is also an open issue.

8. FUTURE RESEARCH

We identified several future research directions related to closed labs. Most of them stem from the inconclusive nature of our conclusions presented in the previous section.

Below is a list of future research directions, we believe, are suitable for closed CS1/2 labs:

- While it is difficult to compare closed lab design frameworks, it might useful to investigate how other STEM fields can be leveraged to improved closed CS1/2 labs.
- We do not expect a universally adopted assessment methodology in near future, but we believe we should learn from other STEM fields to see how their assessment methodologies can better inform CS closed lab assessment strategies.
- We learned that contextualization helps. We did not find any study that addressed how the background of an instructor teaching a contextualized course affects student learning outcomes. We would like to know what the minimum level of expertise in the contextual area that a CS faculty should have to successfully teach a contextualized introductory programming course/lab.
- Finally, a central repository of information to relevant to success in closed labs should be created.

9. CONCLUDING REMARKS

To the best of our knowledge, no review paper on closed lab practices in CS education, similar to this one has been recently published. This paper was motivated by the aforementioned gap and discussions among colleagues in our department. As a medium-sized regional comprehensive institution, our faculty teach introductory programming on a rotating schedule. While we keep learning outcomes and student outcomes consistent, the administration of the course (e.g., choice, duration and type of lab) is left to the discretion of the course instructor. In light of this reality, we took an opportunity to conduct controlled, theory-driven experiments (currently underway) about novel enhancements to closed laboratories. We hope that this will inspire others to consider possible improvements to student learning by tweaks in their current practice. A large majority of our written end-of-semester student comments on teacher/course evaluations for introductory programming courses mention labs as an important and useful experience, thus memorable. We conjecture that gender-sensitive tweaks to labs have a strong potential to increase diversity in our major. Moreover, we ask the astute reader-practitioner to consider other cultural sensitivities in their lab designs, for these first impressions of our field of study are long-lasting.

10. REFERENCES

[1] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and B. Tribelhorn. A data programming cs1 course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 150–155. ACM, 2015.

[2] K. Becker. Teaching with games: the minesweeper and asteroids experience. *Journal of Computing Sciences in Colleges*, 17(2):23–33, 2001.

[3] D. Chavey. A structured laboratory component for the introductory programming course. In *ACM SIGCSE Bulletin*, volume 23, pages 287–295. ACM, 1991.

[4] W.-K. Chen and Y. C. Cheng. Teaching object-oriented programming laboratory with computer game programming. *Education, IEEE Transactions on*, 50(3):197–203, 2007.

[5] D. C. Cliburn. The effectiveness of games as assignments in an introductory programming course. In *Frontiers in Education Conference, 36th Annual*, pages 6–10. IEEE, 2006.

[6] D. R. Garrison, T. Anderson, and W. Archer. Critical inquiry in a text-based environment: Computer conferencing in higher education. *The internet and higher education*, 2(2):87–105, 1999.

[7] M. Guzdial. Does contextualized computing education help? *ACM Inroads*, 1(4):4–6, 2010.

[8] M. Guzdial. Exploring hypotheses about media computation. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 19–26. ACM, 2013.

[9] H. Hakimzadeh, R. Adaikkalavan, and R. Batzinger. Successful implementation of an active learning laboratory in computer science. In *Proceedings of the 39th annual ACM SIGUCCS conference on User services*, pages 83–86. ACM, 2011.

[10] D. Horton, M. Craig, J. Campbell, P. Gries, and D. Zingaro. Comparing outcomes in inverted and traditional cs1. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 261–266. ACM, 2014.

[11] R. Ibrahim, R. C. M. Yusoff, H. M. Omar, and A. Jaafar. Students perceptions of using educational games to learn introductory programming. *Computer and Information Science*, 4(1):p205, 2010.

[12] D. Ifenthaler, D. Eseryel, and X. Ge. *Assessment for*

game-based learning. Springer, 2012.

[13] K. M. Kapp. *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

[14] A. N. Kumar. The effect of closed labs in computer science i: an assessment. *Journal of Computing Sciences in Colleges*, 18(5):40–48, 2003.

[15] A. N. Kumar. Closed labs in computer science i revisited in the context of online testing. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 539–543. ACM, 2010.

[16] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin*, 39(1):115–118, 2007.

[17] C. S. Loh, Y. Sheng, and D. Ifenthaler. *Serious Games Analytics: Methodologies for Performance Measurement, Assessment, and Improvement*. Springer, 2015.

[18] R. McCauley, W. Pharr, G. Pothering, and C. Starr. A proposal to evaluate the effectiveness of closed laboratories in the computer science curriculum. *Journal of Computing Sciences in Colleges*, 19(3):191–198, 2004.

[19] R. McCauley, C. Starr, W. Pharr, R. Stalvey, and G. Pothering. Is cs1 better with the same lecture and lab instructor? *ACM SIGCSE Bulletin*, 38(2):54–60, 2006.

[20] R. P. Mihail, J. Goldsmith, N. Jacobs, and J. W. Jaromczyk. Teaching graphics for games using microsoft xna. In *Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games (CGAMES), 2013 18th International Conference on*, pages 36–40. IEEE, 2013.

[21] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik. Improving the cs1 experience with pair programming. In *ACM SIGCSE Bulletin*, volume 35, pages 359–362. ACM, 2003.

[22] L. Porter and B. Simon. Retaining nearly one-third more majors with a trio of instructional best practices in cs1. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 165–170. ACM, 2013.

[23] R. Rajaravivarma. A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, 37(4):98–102, 2005.

[24] C. F. Reilly and N. De La Mora. The impact of real-world topic labs on student performance in cs1. In *Frontiers in Education Conference (FIE), 2012*, pages 1–6. IEEE, 2012.

[25] L.-K. Soh, A. Samal, and G. Nugent. A framework for cs1 closed laboratories. *Journal on Educational Resources in Computing (JERIC)*, 5(4):2, 2005.

[26] L.-K. Soh, A. Samal, S. Person, G. Nugent, and J. Lang. Analyzing relationships between closed labs and course activities in cs1. In *ACM SIGCSE Bulletin*, volume 37, pages 183–187. ACM, 2005.

[27] L.-K. Soh, A. Samal, S. Person, G. Nugent, and J. Lang. Closed laboratories with embedded instructional research design for cs1. *ACM SIGCSE Bulletin*, 37(1):297–301, 2005.

[28] G. Struble. Most effective lab exercises. *Computer Science Education*, 1(2):177–180, 1989.

[29] K. Sung, P. Shirley, and B. R. Rosenberg. Experiencing aspects of games programming in an introductory computer graphics class. In *ACM SIGCSE Bulletin*, volume 39, pages 249–253. ACM, 2007.

[30] N. Titterton and M. J. Clancy. Adding some lab time is good, adding more must be better: the benefits and barriers to lab-centric courses. In *FECS*, pages 363–367, 2007.

[31] N. Titterton, C. M. Lewis, and M. J. Clancy. Experiences with lab-centric instruction. *Computer Science Education*, 20(2):79–102, 2010.

[32] H. M. Walker. A lab-based approach for introductory computing that emphasizes collaboration. In *Computer Science Education Research Conference*, pages 21–31. Open Universiteit, Heerlen, 2011.

[33] W. A. Waller. A framework for cs1 and cs2 laboratories. In *ACM SIGCSE Bulletin*, volume 26, pages 198–202. ACM, 1994.

[34] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller. In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):197–212, 2002.