Peer Instruction Contributes to Self-Efficacy in CS1

Daniel Zingaro
OISE
University of Toronto
Toronto, ON, Canada
daniel.zingaro@utoronto.ca

ABSTRACT

Recent work in computing suggests that Peer Instruction (PI) is a valuable interactive learning pedagogy: it lowers fail rates, increases retention, and is enjoyed by students and instructors alike. While these findings are promising, they are somewhat incidental if our goal is to understand whether PI is "better" than lecture in terms of student outcomes. Only one recent study in computing has made such a comparison, finding that PI students outperform traditionally-taught students on a CS0 final exam. That work was conducted in a CS0, where the same instructor taught both courses, and where the only outcome measure was final exam grade. Here, I offer a study that complements their work in two ways. First, I argue for and measure self-efficacy as a valued outcome, in addition to that of final exam grade. Second, I offer an inter-instructor CS1 study, whose biases differ from those of intra-instructor studies. I find evidence that PI significantly increases self-efficacy and suggestively increases exam scores compared to a traditional lecture-based CS1 class. I note validity concerns of such an in-situ study and offer a synthesis of this work with the extant PI literature.

Categories and Subject Descriptors

K.3.2 [**Computer Science Education**]: Computer and Information Science Education

Keywords

CS1; peer instruction; clickers; classroom response; outcomes

1. INTRODUCTION

CS1 (Computer Science 1) is the first course taken by CS majors at the post-secondary degree level. As this course serves as entry into the CS curriculum, it has received significant attention by the CS education community. A recurring concern in the contemporary CS1 research is that students are not learning what we expect and that disproportionate numbers of students worldwide are failing CS1 [3]. Bimodal grade distributions [23] and assertions that programming skill cannot be taught [7] have quickened the pace with which student's grade-based outcomes have been measured and predicted. In general, this program of research seeks to predict students' CS1 grade or final exam grade through characteristics such as prior programming experience [10], math ability [4] and learning style [12], to name a few. The breadth of student predictors is both broad and creative, and serves as a testament to the effort put forth by the research community to thoroughly explore the hypothesized link between "what students have" and whether they succeed.

However, while the sheer number of studied predictors is comprehensive, two other areas of the research have gone largely unquestioned. First, the research narrowly defines success in CS1 as the student's score on an exam or the course as a whole. It rarely acknowledges the importance of self-efficacy as a valued outcome in and of itself. Second, it rarely acknowledges the potential importance of pedagogy in the link between student-based predictors and outcomes. This is a particularly important gap given the recent interest in Peer Instruction (PI) and other active learning pedagogies for teaching CS courses.

This paper contributes in two ways to the gaps mentioned here. I offer a comparison of two sections of a CS1, one taught using PI and the other taught traditionally. In this way, I can compare outcomes from courses that are otherwise held constant (subject to validity concerns as described later). Second, I take both final exam grade and post-course self-efficacy as independent, valued outcomes of CS1. This work complements a similar study of a CS0 that compared PI to a traditional section [26]. In that work, the PI section scored significantly higher on the final exam than the traditional section. In the present work, I find suggestive (but not statistically significant) evidence of the same finding. However, as I also measure self-efficacy, I document that students in the PI section show higher self-efficacy than the control section. The benefit of a broad conception of "success" is evident in that the self-efficacy increases would have been missed through a sole examination of final grade.

SIGCSE'14, March 3–8, 2014, Atlanta, GA, USA.

http://dx.doi.org/10.1145/2538862.2538878.

2. BACKGROUND AND LITERATURE REVIEW

2.1 Peer Instruction

Peer Instruction (PI) is a pedagogical technique developed in physics that has since been used with considerable success in CS. At the core of this pedagogy is the ConcepTest [6]: a multiple-choice question answered by students typically using clickers. Each ConcepTest sets off a well-defined pedagogical protocol: students first answer the question individually (solo vote), then discuss the same question for several minutes with their neighbors, and finally re-vote on the question in light of the group discussion (group vote). Following the group vote, the instructor facilitates a classwide discussion and explanation of the ConcepTest, before dynamically adjusting the class based on student performance. The instructor may lecture briefly before or after a ConcepTest, but the vast majority of lecture time engages students in discussions with peers. The role of the instructor is to help foster substantive discussions of core course concepts, and to focus students on each distractor rather than on answering the question correctly [25, 28]. This downplaying of lecture means that instructors cannot "cover" as much material, so students are typically required to read one or two textbook sections before class and complete a small preparatory reading quiz [6, 28].

In a taxonomy of observable learning activities, Chi [5] argues that "interactive" learning activities may be most powerful for engendering learning. Such activities are characterized by learners dialoguing and discussing concepts, where each peer makes substantive contributions to the discourse. Simply grouping students for PI does not guarantee that their discussions will be interactive as described here, but recent work suggests that students see PI as more interactive than traditional lecture [24]. We therefore expect PI students to exhibit strong gains, and for the most part this is borne out by the literature. For example, PI reduces failure rates [19], contributes to increased retention [20], and yields exam-inferred learning gains in CS0 courses [26]. Students learn from their discussions with peers [18] and students and teachers alike value the pedagogy for its effects on learning and argumentation [17]. See the recent reviews [29, 28] for further commentary on these and other benefits.

2.2 Self-Efficacy as Valued Outcome

The most observable measure of course success, and certainly one of the most important, is student grade. The vast majority of CS studies concerned with measuring outcomes have operationalized success as course grade or final exam grade. Further, these studies tend to study lecture-based offerings of CS1. The typical finding is that students with prior programming experience earn higher grades compared to inexperienced students. This finding is robust in the sense that even a small amount of prior programming experience is helpful [10]. Having taken a CS course [8], studied a programming language [10], or dabbled in programming as a hobby [11] all positively predict performance in a CS1. To be clear, there are isolated studies where this relationship is deemed not to hold, but the relevant features of those examples have resisted capture or replication. For example, in an objects-first CS1, no prior experience gap was evident [27]. Yet, in other "objects-first" courses, the gap is back [2]. I argue that a well-defined pedagogical shift — from traditional to PI — may yield more reliable results than particulars of the content being taught.

Most relevant to this discussion, Simon et al. [26] have recently reported a comparison of a PI section of a course to a traditional section. To determine whether PI students demonstrated enhanced learning, the sections' final exam grades were compared. They found that PI students scored almost 6% higher on the common final exam than traditional students.

While grade is an important outcome, it is not the only important outcome. Elsewhere, I have argued for the importance of measuring CS students' interest and enjoyment of CS to complement grade-based outcomes (Zingaro, in submission). Students take CS courses for a variety of reasons including job prospects, challenging themselves with difficult material, and understanding societal trends and changes [13, 9]. These are only peripherally-related to the grades students earn. There are other outcomes that may suggest that students are getting what they want from our courses, and we must measure these outcomes in order to broaden our understanding of student success.

One such outcome measure that may only partially overlap grades is self-efficacy. Self-efficacy is the most-studied sociocognitive attribute of CS1 students, but it is typically studied as a predictor of (grade) success, not as a legitimate outcome in itself. It is defined as the conviction that one can successfully orchestrate behavior to accomplish something [1]. Self-efficacy expectations are differentiated from outcome expectations, the latter of which refer to the instrumental relationship between a behavior and an outcome. Bandura [1] explains that efficacy judgments help determine whether coping behavior will be initiated, the amount of effort expended, and level of perseverance in the face of obstacles. It is for this reason that I take self-efficacy as an important measure of what students "get" from CS1. If students' self-efficacy is strengthened, then this is valuable whether or not it manifests immediately in heightened grades. For example, students with high self-efficacy following CS1 may be more likely to continue to a second CS course, where perhaps grade will catch-up to their belief in what they can do. Of course, self-efficacy expectations in and of themselves are not sufficient to produce successful behavior; but in the presence of sufficient skill and motivation, efficacy expectations are crucial in shaping consequent action. The most powerful mediator of self-efficacy is performance accomplishments, though vicarious experience, verbal persuasion, and associated physiological states also play a role.

The only widely-used self-efficacy scale for computer programming is that developed by Ramalingam and Wiedenbeck [22]. This scale contains 32 questions primed to object-oriented programming in C++. Students rate on a seven-point scale their confidence that they could carry out specific and relevant programming tasks. These authors administered the scale once prior to any instruction, and again after the final lecture. Factor analysis on the pre-self-efficacy administration suggests that this scale is composed of four factors: independence and persistence, complex programming tasks, self-regulation, and simple programming tasks. In the traditionally-taught CS1 courses, self-efficacy increases from pre-course to post-course [22, 21].

2.3 Hypotheses

PI courses give students considerable opportunities to experience small successes. Each class has students discuss and answer three or more important but small questions. If self-efficacy is tied largely to performance accomplishment, then it should be that PI increases self-efficacy beyond that of a traditional course offering. In addition, as PI has shown to improve grades over a traditional offering, I hypothesize that PI students' final exam scores will be higher than those of traditionally-taught students. My first two hypotheses are therefore:

- PI students leave CS1 with higher self-efficacy than traditionally-taught students.
- PI students will earn higher grades on a final exam than their traditionally-taught counterparts.

In addition, I suggest that links between previous programming experience and gender on the one hand and outcomes on the other might be moderated by pedagogy.

As noted earlier, the literature shows strong links between prior programming experience and grades, but that work was conducted solely in lecture sections. To the extent that PI gives students of lesser experience opportunities to learn from experienced peers, I hypothesize that the gap generated by prior experience might be smaller in the PI section.

In an ethnographic study of gendered experiences in CS, Margolis and Fisher [14] argue that contextualized learning is important for women's success. They describe the computing-related narratives of many male college students as approximating an in-born, magnetic attraction to computers. These males are attracted to the physical computer for its own sake, in contrast to females who often understand computers as serving larger societal goals. Women's stories run counter to the idea that CS majors "hack for hacking's sake". They are motivated by a desire to help others, an appreciation of the versatility of CS, and encouragement from family and friends. Large, impersonal and competitive lectures disadvantage women who are bolstered by social and academic support. Margolis and Fisher [14] give three suggestions for creating what they call a course in "contextualized computer science": situating technology in realistic settings, making connections to other disciplines, and using diverse problems and teaching methods. I contend that PI addresses, at least partially, the first and third of these guidelines, and therefore that the gender gap (if one exists in my setting) will be reduced in the PI section. My second set of hypotheses is therefore:

- PI will reduce the prior experience gap.
- PI will reduce the gender gap.

3. METHOD

I report on a CS1 taught in Fall 2012 at an undergraduate campus of a large Canadian research-intensive university. Two sections of the course, taught by different instructors, were offered: a PI offering and a traditional offering. The course covers traditional CS1 topics in imperative programming, and also spends one week each on sorting, complexity, and object-oriented programming. The course took place over 12 weeks, with three 50-minute lectures and one lab session per week. The course has been taught in Python 2 since 2008 and the two Fall 2012 instructors worked together to revamp the course for Python 3.

In the PI section, the instructor began the course by introducing the rationale for using PI, covering some of the research findings and goals for the peer discussions. Prior to each lecture, students completed a reading quiz; the instructor read the responses to help shape the following lecture. The reading quizzes were marked based on completion (not correctness) and were worth 4% of students' final grade; in-class clicker participation accounted for a further 5% of students' grade. Each lecture was focused on three to four ConcepTests, with mini-lectures interspersed when planned by the instructor or required based on question performance. The course instructor was a senior education graduate student with significant PI and CS teaching experience, and had taught CS1 using PI several times.

The traditional section was taught by an instructor with significant CS and Python teaching experience. This section used the same labs, assignments, midterm, and final exam as the PI section. The two sections of the course were synchronized both in the topics to be covered each class and the examples used to teach those topics. The PI instructor used multiple choice questions and the PI process, while the traditional instructor introduced and worked examples through a lecture format. Importantly, both instructors were teaching in their preferred modes (PI or traditional) and developed and followed agreed-upon plans for each lecture.

As the traditional offering did not have reading quizzes or class participation marks, those students were required to submit three small exercises throughout the semester that were each worth 3%. The PI instructor used some of the very same exercise questions in the students' reading quizzes so as to attempt to equalize exposure to these small programming questions.

Consenting students were asked to respond to two surveys: one at the start of term and one in the last two weeks of classes. On the first questionnaire, an initial measure of self-efficacy was collected, along with the number of prior CS courses taken. Prior programming experience was collected because of its reliable role in predicting student grade (e.g. [10, 15]). On the end-of-term questionnaire, a post-self-efficacy measure was obtained. The self-efficacy measure was adapted from [22] by replacing "Java" with "Python". Student final exam grades were obtained as the second outcome measure.

4. THREATS TO VALIDITY

As a quasi-experimental study, there are threats to the validity of the results here. I survey these threats, and in some cases describe how they were partially controlled:

Different Instructors. This is an inter-instructor study, where one instructor taught the PI section and the other taught the traditional section. This is to be contrasted with an intra-instructor study where the instructor teaches both sections [26]. These designs each have limitations that are only partially-overlapping. For example, an instructor may prefer one mode over the other, so intra-instructor studies force the instructor to teach in the mode they disprefer. In both designs, the quality of instruction in one section may be better than that in the other section. Inter-instructor studies, as the present one, successfully engage instruc-

tors in teaching in their preferred modes, but do so through a loss of control of general instructor effectiveness. For example, perhaps the PI instructor is globally "better" or overall works harder than the traditional instructor, or vice versa. In the present study, both instructors were teaching the Python 3 CS1 for the first time and developed their own lectures following the same teaching plan. Post-course student comments were positive in both cases, but of course such evaluations cannot disentangle instructor effects from pedagogy effects.

Different Students. The other obvious confound is that qualitatively different students enrolled in the two sections. However, the two sections were offered back-to-back, the two instructors were unknown to the students, and the students were unaware that the sections would use different pedagogies. Students could have switched sections early in the term, but anecdotally this was observed very infrequently. As an additional check, I found that pre-self-efficacy and prior experience measures did not statistically differ per section ($p > .05$ in both cases), giving credence to the suggestion that students were similar.

5. RESULTS

Across both sections, 221 students wrote the final exam. Data for the present study includes 109 (49%) of these students: these are the students that responded to both questionnaires and took at least 40 seconds to respond to the post-self-efficacy questionnaire. (The self-efficacy scale contains 32 questions; 40 seconds was used to remove submissions where students were unlikely to have read many of the questions at all.) The Cronbach's reliability of the self-efficacy questionnaire was 0.98.

Multiple regressions were used to test the predictors of post-self-efficacy and exam grade. Regressors were entered in two blocks. In the first block, main effects of gender, section, and prior experience were added. In the second block, interactions between section and gender and between section and prior experience were added. These models allow for the exploration of the four hypotheses of this study.

Post-Self-Efficacy.

The initial models failed to reject a test of non-constant variance; a square root transform on post-self-efficacy corrected the problem. Using a nested chi-square test, I compared the predictive power of the block-1 model (containing only the main effects) with the block-2 model (additionally containing the interactions). A non-significant result ($p = .65$) shows that the block-2 model is no better than the block-1 model. That is, there are no significant interactions between pedagogy and gender, or between pedagogy and prior experience.

The block-1 (main effects) model of self-efficacy is given in the left-hand column of Table 1. I find a significant effect of section ($p = .015$), such that PI students' self-efficacy (5.13) was higher than the self-efficacy of traditionally-taught students (4.65). A significant effect of gender ($p = 0$) shows that females have lower post-self-efficacy (4.12) than males (5.32). Finally, prior course-based programming experience was associated with higher post-self-efficacy ($p = .003$).

	self-efficacy	exam grade
(Intercept)	24.16 (1.06)*	57.08 (1.82)*
section	2.46 (1.00)*	2.43 (1.71)
gender	−4.34 (1.08)*	−3.52 (1.85)
prior_courses	3.00 (0.98)*	3.97 (1.68)*
R^2	0.27	0.12
Adj. R^2	0.25	0.09

*$p < 0.05$

Table 1: Multiple regressions for post-self-efficacy and final exam grade.

Exam Grade.

Using a nested chi-square test, I compared the predictive power of the block-1 model (containing only the main effects) with the block-2 model (additionally containing the interactions). A non-significant result ($p = .74$) shows that the block-2 model is no better than the block-1 model. That is, there are no significant interactions between pedagogy and gender, or between pedagogy and prior experience. (This mirrors the finding for post-self-efficacy.)

The block-1 (main effects) model of final exam grade is given in the right-hand column of Table 1. Section is insignificant ($p = .16$), suggesting that PI and traditionally-taught students did not differ on final exam grade. PI students were predicted to score 2.4% higher on the exam, but this difference was not statistically significant (c.f. [26]). Using the full student roster (not just the subset of students that responded to the questionnaires), PI students scored 4.4% higher, but a t-test remained non-significant ($p = .10$). Moving on, I find a significant effect of prior experience ($p = .0198$), such that students who took prior programming courses performed better on the final exam. A near-significant effect of gender ($p = .06$) suggests that females performed more poorly than males when controlling for prior experience. On the final exam, males scored an average of 61% and females scored on average 52%.

6. DISCUSSION

I now return to the four hypotheses of the study.

As predicted, PI students' post-self-efficacy was higher than that of students in the traditional section. This is perhaps due to the numerous opportunities for quick, accurate feedback in the PI class, where students could experience small successes before tackling larger labs and assignments [22].

However, the second hypothesis — that PI students would also score higher on the final exam — was not supported. Considered together, this suggests an apparent contradiction: if PI students believe they can organize activity toward success, why were they no more successful on the final assessment? I suspect that this may have to do with the type of final exam that is typically administered in CS1s. Prior research shows that CS1 exams typically involve several large, integrative code-writing questions [16]. Students are required to synthesize several concepts in order to arrive at a coherent program to solve a problem, and this tends to limit opportunities for students to demonstrate what they know. (That is, there is a concern that these integrative questions measure all-or-nothing.) The instructors of the

present CS1 intentionally designed such an exam to correspond to this CS1 practice. That is, the exam was mostly composed of integrative code-writing questions. This is to be contrasted with the earlier study by Simon et al. [26] where a PI section of a CS0 outperformed a traditional section. In that context, the final exam contained many multiple choice questions in addition to some short answer questions. What I am arguing is that the elevated self-efficacy of PI students did not present in terms of higher grades because the students may not yet have the required practice and experience to successfully orchestrate the mechanics required for integrative code-writing questions. Such self-efficacy, however, may prove useful for these students going forward.

The third and fourth hypotheses were not supported. I found overall evidence that males performed better than females, and that those with prior CS course experience performed better than those who hadn't taken a CS course. However, the use of PI did not moderate these relationships. That is, gender- and experienced-based gaps existed in this CS1, and they were not lessened by a change in pedagogy.

Overall, PI marginally (but insignificantly) led to increased final exam scores, and significantly contributed to post-self-efficacy. While these results are more equivocal than those previously reported for a CS0 [26], they do suggest the importance of multiple outcome measures to provide more nuanced understandings of pedagogical interventions. In future work, I seek to study a more balanced CS1 exam to determine (a) whether PI students outperform traditionally-taught students on some tasks and, if so, (b) precisely where the differences lie. That is, I do not argue here that PI did not help students conceptually, only that students may not have had the opportunity to demonstrate this understanding on the type of CS1 exam in common use. Of course, it may be that PI is more effective in a CS0 compared to a CS1, particularly when many questions on the final exam are multiple choice as in the Simon et al. study [26]. Questions of when and in what ways PI helps our students across different courses are exciting possibilities for future research.

7. CONCLUSION

When outcomes include more than final exam grade, our understanding of course success correspondingly broadens. In this CS1 study, I examine outcomes of final exam grade and self-efficacy for students taught using Peer Instruction (PI) and those taught in traditional lecture style. I find significant gains in self-efficacy for the PI students, but no significant evidence that final exam scores differed (though PI students did score marginally higher). I offer that the increased self-efficacy is a win in itself, and urge the community toward a multi-faceted conception of success in CS1. Students earn more than grades in our courses: they possibly become interested in CS, enjoy the lectures, gain self-efficacy, and so on. To truly understand the effect of a pedagogical shift requires an understanding of more than grade-based outcomes.

Acknowledgments.

I thank anonymous reviewer "Spino" for feedback on an early version of this work

8. REFERENCES

[1] A. Bandura. Self-efficacy: toward a unifying theory of behavioral change. *Psychological Review*, 84(2):191–215, 1977.

[2] J. Bennedsen and M. E. Caspersen. An investigation of potential success factors for an introductory model-driven programming course. In *Proceedings of the first international workshop on Computing education research*, pages 155–163. ACM, 2005.

[3] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *SIGCSE Bulletin*, 39:32–36, 2007.

[4] P. Byrne and G. Lyons. The effect of student attributes on success in programming. *SIGCSE Bulletin*, 33:49–52, 2001.

[5] M. T. H. Chi. Active-constructive-interactive: A conceptual framework for differentiating learning activities. *Topics in Cognitive Science*, 1(1):73–105, 2009.

[6] C. H. Crouch, J. Watkins, A. P. Fagen, and E. Mazur. Peer instruction: Engaging students one-on-one, all at once. In E. F. Redish and P. J. Cooney, editors, *Research-Based Reform of University Physics*. American Association of Physics Teachers, College Park, MD, USA, 2007.

[7] S. Dehnadi, R. Bornat, and R. Adams. Meta-analysis of the effect of consistency on success in early learning of programming. In *21st Annual Workshop of the Psychology of Programming Interest Group*, 2009.

[8] G. E. Evans and M. G. Simkin. What best predicts computer proficiency? *Communications of the ACM*, 32:1322–1327, 1989.

[9] A. Fisher, J. Margolis, and F. Miller. Undergraduate women in computer science: experience, motivation and culture. *SIGCSE Bulletin*, 29:106–110, 1997.

[10] D. Hagan and S. Markham. Does it help to have some programming experience before beginning a computing degree program? *SIGCSE Bulletin*, 32:25–28, 2000.

[11] E. Holden and E. Weeden. The impact of prior experience in an information technology programming course sequence. In *Proceedings of the 4th conference on Information technology curriculum*, pages 41–46. ACM, 2003.

[12] M. A. Hudak and D. E. Anderson. Formal operations and learning style predict success in statistics and computer science courses. *Teaching of Psychology*, 17(4):231–234, 1990.

[13] T. Jenkins. The motivation of students of programming. *SIGCSE Bulletin*, 33(3):53–56, 2001.

[14] J. Margolis and A. Fisher. *Unlocking the clubhouse: Women in computing*. The MIT Press, Cambridge, MA, USA, 2002.

[15] M. Morrison and T. S. Newman. A study of the impact of student background and preparedness on outcomes in CS I. *SIGCSE Bulletin*, 33(1):179–183, 2001.

[16] A. Petersen, M. Craig, and D. Zingaro. Reviewing CS1 exam question content. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 631–636. ACM, 2011.

[17] L. Porter, C. Bailey Lee, B. Simon, Q. Cutts, and D. Zingaro. Experience report: a multi-classroom report on the value of peer instruction. In *Proceedings of the 16th annual SIGCSE conference on Innovation and technology in computer science education*, pages 138–142. ACM, 2011.

[18] L. Porter, C. Bailey-Lee, B. Simon, and D. Zingaro. Peer instruction: Do students really learn from peer discussion in computing? In *Proceedings of the Seventh international Workshop on Computing Education Research*. ACM, 2011.

[19] L. Porter, C. B. Lee, and B. Simon. Halving fail rates using peer instruction: A study of four computer science courses. In *Proceedings of the 44th ACM technical symposium on Computer science education*, pages 177–182. ACM, 2013.

[20] L. Porter and B. Simon. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proceedings of the 44th ACM technical symposium on Computer science education*, pages 165–170. ACM, 2013.

[21] V. Ramalingam, D. LaBelle, and S. Wiedenbeck. Self-efficacy and mental models in learning to program. *SIGCSE Bulletin*, 36:171–175, 2004.

[22] V. Ramalingam and S. Wiedenbeck. Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, 19:365–379, 1998.

[23] A. Robins. Learning edge momentum: A new account of outcomes. *Computer Science Education*, 20(1):37–71, 2010.

[24] B. Simon, S. Esper, L. Porter, and Q. Cutts. Student experience in a student-centered peer instruction classroom. In *Proceedings of the Ninth International Workshop on Computing Education Research*. ACM, 2013.

[25] B. Simon, M. Kohanfars, J. Lee, K. Tamayo, and Q. Cutts. Experience report: Peer instruction in introductory computing. In *Proceedings of the 41st SIGCSE technical symposium on Computer science education*, pages 341–345. ACM, 2010.

[26] B. Simon, J. Parris, and J. Spacco. How we teach impacts student learning: peer instruction vs. lecture in cs0. In *Proceedings of the 44th ACM technical symposium on Computer science education*, pages 41–46. ACM, 2013.

[27] P. Ventura. *On the origins of programmers: Identifying predictors of success for an objects first CS1*. PhD thesis, SUNY at Buffalo, 2003.

[28] D. Zingaro, C. Bailey-Lee, and L. Porter. Peer instruction in computing: the role of reading quizzes. In *Proceedings of the 44th ACM technical symposium on Computer Science Education*, pages 47–52. ACM, 2013.

[29] D. Zingaro, A. Petersen, and M. Craig. Stepping up to integrative questions on CS1 exams. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 253–258. ACM, 2012.