Peer Instruction: a Link to the Exam

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

Leo Porter
Dept. of Mathematics and Computer Science
Skidmore College
Saratoga Springs, NY, USA
leo.porter@skidmore.edu

ABSTRACT

In computer science, the active learning pedagogical practice of Peer Instruction (PI) has been shown to improve final exam performance, reduce student failure rates, and improve student retention. PI consists of two major parts: group discussion and follow-up instructor intervention. We expect that PI performance as a whole will correlate with final exam performance, but it is unclear whether or how each piece of PI is involved in these relationships. In this work, we use isomorphic questions to isolate the effects of peer discussion and instructor intervention, and examine scores on a final exam and its code-writing and code-tracing questions. We find that both pieces of PI correlate with the final exam as a whole, code-tracing question (similar to PI questions), and code-writing question (not similar to PI questions). This is further evidence that both PI components are important to the success of PI.

Categories and Subject Descriptors

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

Keywords

peer instruction; clickers; final exam

1. INTRODUCTION

Concerns that our computer science students are failing at alarming rates [2] and that many of our students are not demonstrating acceptable learning outcomes [7] have spurred the CS research community to examine how we might respond as teachers. Prompted by research from other disciplines such as psychology and physics education, CS researchers have begun questioning the lecture as the pedagogical basis of CS courses. This is evident in the recent interest in inverted classrooms [8] and in-class collaboration [6], each of which decenters the lecture as the core element of teaching. One particularly promising alternative to lecture is Peer Instruction (PI) [3, 15]. In a PI class, students participate in multiple iterations of individually answering a question (using a clicker), discussing that question in a group, answering the question again, and then participating in an instructor-led classwide discussion of the question.

Recent research has shown PI to offer a number of benefits over traditional lecture in CS courses. For example, PI can contribute to substantially lower failure rates [12] and improved retention of majors [13]. In addition, similar to other collaborative learning domains [6], students report being particularly engaged in PI classes [14].

As the momentum in favour of new pedagogies builds, it is paramount that we measure student learning. A growing body of research therefore seeks to verify that students learn from the PI process. Some of this work uses isomorphic (same-concept) questions to measure performance before and after peer discussion [11]. Such questions have been used to demonstrate that students learn from the peer-discussion part of the PI process. Other work shows that students in a PI class outperform students in a matched lecture class [16, 20]. As yet, no work has investigated links between PI performance and final exam performance. To be sure, we expect such links to exist — we know that PI is correlated with enhanced performance — but it is not clear whether the peer discussion, the instructor intervention, or both, are responsible for these expected links.

We examine student performance during the in-class PI process using isomorphic questions and use regression modeling to determine whether this performance predicts final-exam grades. We anticipate there to be a great deal of noise between a student demonstrating learning a concept in a class during the early weeks of a term and the final exam months later, due to other in-class content, laboratory assignments, programming assignments, outside-class discussions, studying, programming practice, etc.

Despite this noise, we expect, and find, that students who come to class already prepared to answer questions correctly do better on a final exam consisting of code-tracing and code-writing questions. We also find, controlling for baseline performance, that student learning during the PI process, both from peers and from the instructor, relates to higher exam scores. This is the first work, to our knowledge, that directly connects learning during the in-class PI process with student performance at the end of the term. We discuss the elements of PI that we believe are particularly valuable and address potential threats to validity.

ITiCSE'14, June 21–25, 2014, Uppsala, Sweden
Copyright 2014 ACM 978-1-4503-2833-3/14/06 ...$15.00.
http://dx.doi.org/10.1145/2591708.2591711 ...$15.00.

2. BACKGROUND AND LITERATURE REVIEW

2.1 Active Learning in Computer Science

Computing educators have long used active and collaborative learning techniques outside of lecture, most notably in the use of pair programming [9]. Recently, notions of an "inverted classroom" have brought active learning to the forefront of CS lectures as well. An inverted classroom involves asking students to prepare for lecture, perhaps by reading, watching a video, or exploring programming techniques [8]. The justification is that our contact time with students is minimal, so it should be used in ways that maximize the utility of the instructor. Students know how to read but may not know how to integrate their reading with other disciplinary knowledge, hence the divide between reading before lecture and instructor-led learning during lecture.

Other disciplines, such as physics and psychology, have rich histories of pedagogical interventions that are presently being exercised in computing courses. For example, think-pair-share (TPS) is a popular cooperative learning technique that has students individually ponder, discuss with ad hoc groups, and then share with the entire class. Recent work has found that students are engaged in each of the three phases of the technique [6] and that much of this engagement is combined with active learning such as discussing with peers and contributing to discussion. PI has much in common with TPS and other active learning pedagogies through its focus on group discussion and conceptual understanding.

2.2 Peer Instruction

Peer Instruction (PI) focuses students on several conceptual multiple choice questions ("conceptests") per class meeting. The PI protocol includes the instructor posing a multiple choice question, giving students a minute or two to respond with a clicker, and providing time for students to discuss in groups and submit a second response. Then, the instructor leads a wrap-up discussion of the particular concept, and moves on to a mini-lecture or the next PI conceptest.

Do students learn from PI in CS? It is interesting to trace the progression of research on this question. Initial studies demonstrated that students' more often answered correctly in the second vote (the group vote) compared to the first vote (the individual vote) [15, 19]. This might mean that students are learning from group discussion, but it could also mean that students are copying from neighbors. To distinguish these alternatives, other research has used isomorphic questions where students vote a third time on a new question very similar to the first. As students vote individually on this "isomorphic vote", students are unable to passively copy from neighbors. These isomorphic studies use common terminology that we will also use below, so we introduce these terms:

- Q1: the individual vote on the first question.
- Q1$_{ad}$: the group vote on the first question. This occurs after students have discussed the question in groups.
- Q2: the individual vote on the second (isomorphic) question.

Porter et al. [11] used isomorphic questions to study PI learning in Computer Architecture and Theory of Computation. They particularly focused on those students who incorrectly answered Q1 and correctly answered $q_1 ad$, since these are the students that may have learned (or copied) during the peer discussion. The authors found that 76% and 62% of these students correctly answered Q2, suggesting that real learning was happening during peer discussion.

Of course, the complete PI cycle — including Q1, peer discussion, and Q2 — occurs within a few minutes, and all measurements have been taken during this small time window. That is, learning might be happening, but is the learning enduring? A logical follow-up to the work of Porter et al. [11] is to examine the link between Q2 and the final exam, when Q2 represents learning from various parts of the PI process. Students' correctness on Q2 should correlate with correctness on the final exam to the extent that peer- and instructor-based learning are long-lasting. We follow this line of inquiry in the present paper.

3. METHOD

3.1 Study Context

Data for this study comes from a CS1 taught in Python at a large campus of a Canadian research university. The course uses Python 3 and studies traditional CS1 topics in the following order: introduction, functions, booleans, conditionals, while- and for-loops, lists (including nesting and aliasing), dictionaries, file I/O, testing and test coverage, introduction to object-oriented programming, and introduction to sorting and complexity. 131 students wrote the final exam.

The course took place over 12 weeks, with 3 50-minute lectures per week. Prior to each lecture, students were required to read 10-15 pages of the textbook and submit answers to three questions as part of a reading quiz. Reading quizzes are common and recommended for use in PI classes to bootstrap the discussion process in lecture [3, 5, 21]. 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. The course instructor was a senior education graduate student with significant PI and CS teaching experience, and had taught CS1 using PI several times. New PI materials were developed for this course offering and are freely available for anyone's use [1].

In addition to facilitating PI questions, the instructor live-programmed during lecture to model code-writing. Students also pair-programmed in weekly labs (supported by TAs) and completed two larger programming assignments. This skill-based focus on code-writing was meant to complement the conceptual focus of PI [22].

3.2 Question Administration

The course instructor developed pairs of isomorphic questions, generally using one pair per lecture. (The other PI questions in each lecture were of the standard PI format, with no isomorphic question.) To verify the isomorphic nature of the questions, the instructor sent the proposed isomorphic questions to a colleague experienced in CS1, PI, and isomorphic question administration. Questions were modified as necessary so that both parties agreed that they were

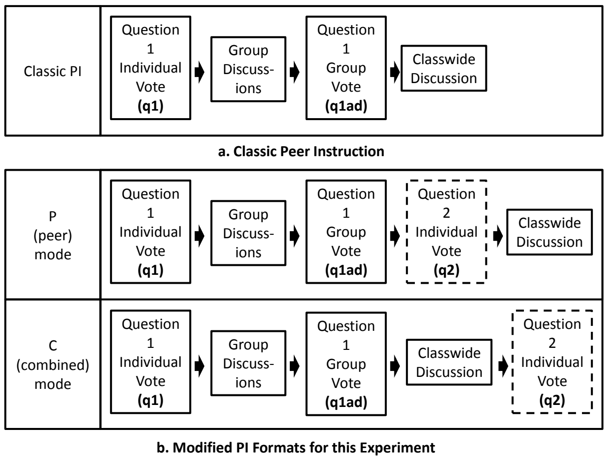
Classic PI
Question 1 Individual Vote (q1)
Group Discuss-ions
Question 1 Group Vote (q1ad)
Classwide Discussion
a. Classic Peer Instruction
P (peer) mode
Question 1 Individual Vote (q1)
Group Discuss-ions
Question 1 Group Vote (q1ad)
Question 2 Individual Vote (q2)
Classwide Discussion
C (combined) mode
Question 1 Individual Vote (q1)
Group Discuss-ions
Question 1 Group Vote (q1ad)
Classwide Discussion
Question 2 Individual Vote (q2)
b. Modified PI Formats for this Experiment

Figure 1: The P (Peer) and C (Combined) administration modes used in this study and [23].

suitably similar and of comparable difficulty. Additional controls (below) were added to help ensure isomorphicity.

For each isomorphic pair, three student votes were taken; we use the notation introduced above and refer to these votes as Q1, Q1_{ad}, and Q2, with the first two votes taken on the first question and the third vote taken on the second. Note that the histogram of responses was never shown between Q1 and Q1_{ad}.

We are interested in the relationship between Q2 and the final exam score when Q2 represents learning from the full PI cycle and when Q2 represents only the effect of peer discussion. To this end, we ran half of the isomorphic questions as follows:

Combined: students individually answered Q1, engaged in peer discussion, and answered Q1_{ad}. Then, the instructor displayed the histogram for Q1_{ad} and proceeded to explain the question, its distractors, and its correct answer. Following instructor intervention, Q2 was shown on which students voted individually. This mode is identical to that of [23].

We ran the other half of the isomorphic questions so that Q2 measured **only** the peer-based learning, not the instructor-imparted learning. That is, these isomorphic questions were run as follows:

Peer: students were shown Q1, individually answered, engaged in peer discussion, and answered Q1_{ad}. Then, students were immediately presented with Q2 and voted individually. Note that between Q1_{ad} and Q2, there was no instructor intervention at all, and that the correct answer to the first question was not displayed. This mode is identical to that of [11, 23].

See Figure 1 for these two administration modes and how they compare to classic PI.

We included the following experimental controls in order to validate isomorphic questions, avoid mode biases, and eliminate poor questions:

- A colleague was engaged in the validation of isomorphic questions (explained earlier).
- The presentation order (Q1 and Q2) within each question pair was randomized at the start of class.
- The choice of whether to run an isomorphic pair as Peer or Combined was randomized at the start of class.
- We removed questions that were too easy — operationalizing this as Q1 of 80% or above — as these questions leave little room for learning [17].

In the Combined mode, we expect that Q2 will be a close approximation to what students know of the concept being tested, since no further class time was spent explicitly discussing the question's topic. Most certainly, CS1 topics recur repeatedly throughout the course, so students would have seen the questions from the PI conceptests throughout the semester. But the concept itself was never explicitly tested and discussed again in isolation. In the Peer mode, by contrast, we expect that Q2 performance will less faithfully represent what students ultimately come to know, since Q2 in this mode does not measure learning from instructor follow-up. When correlating PI performance to final exam performance, therefore, we expect that Peer correctness will relate to the final exam score and that Combined will have additional predictive power over and above Peer.

3.3 Final Exam Questions

The final exam for the CS1 studied here consisted largely of code-writing questions. Such exams are common in CS1, to the extent that research finds many exams consisting mostly or entirely of code-writing questions [10]. Researchers argue that these exams offer students minimal opportunities to demonstrate elements of knowledge as opposed to coherent and complete knowledge structures. For example, progress on the way to being able to write code might not be evident in students' responses if we skip the preliminary steps and ask only for written code. To be sure, we are not advocating code-heavy exams: our final exam was produced in such a way for a different project involving between-lecture comparisons on a "traditional" CS1.

That said, the exam does contain some non-code-writing questions, and we used this limited variety to delineate performance on the final exam. We use three final exam measures in our correlations of PI scores and exam scores:

- The final exam score as a whole out of 100. The exam contained ten questions: one tracing, one sorting, one object-oriented programming, one describing code, and six mostly or completely code-writing. The mean was 38.69 with standard deviation 22.63.
- The score on a code-tracing question containing six questions each worth two marks (mean 4.23, standard deviation 2.95). The questions involved short code segments that added up integers in a loop or manipulated a Python list or dictionary; some of the questions involved nesting or aliasing. These questions are in many ways similar to PI questions: they are small, require students to trace/understand existing code (common in PI questions), and focus on single concepts. For example, one of the subparts asked students for the output of this code segment, focusing on aliasing:

```
composers = ['Kondo', 'Tamura']
composers2 = composers
composers2[0] = 'Ito'
composers2[1] = composers[0]
print(composers)
print(composers2)
```

Final Exam Topics	Block 1 Q1 Alone	Block 2 Q1, Q2 Peer	Block 3 Q1, Q2 Peer, Q2 Combined
Overall R^2	0.21	0.30	0.34
Code-Tracing R^2	0.13	0.16	0.19
Code-Writing R^2	0.16	0.22	0.25

Table 1: R^2 values for each statistical model predicting various facets of final exam performance. Across each row, the change in R^2 between adjacent columns is statistically significant.

- The score on a code-writing question worth 10 marks (mean 3.07, standard deviation 3.03). Students were given a "compressed string" like:
  `abc#2,3#5,2`
  and had to decompress it to `abcabcab`. The `#x,y` directives mean "go back `x` characters and copy `y` characters from there". Note that this question is not similar to PI questions, as it is difficult to ask students to write code in a multiple choice format [22].

4. RESULTS

Multiple regressions were used to test relationships between final exam grades and PI performance. We performed three sets of regressions: one for the final exam at large, one for the code-tracing question, and one for the code-writing question. In each case, we included our predictors in three blocks:

Block 1. We first added two covariates: the number of Peer questions answered correctly and the number of Combined questions answered correctly. We expect that these two predictors will significantly predict exam score because they represent incoming ability prior to each lecture. The more questions that students answer correctly before any peer discussion or instructor intervention, the better they should do on the final exam.

Block 2. In this block, we add the number of Peer Q2 questions that each student answered correctly. If the R^2 change from Block 1 to Block 2 is significant, it suggests that what students learn in the Peer mode (from Q1 to Q2) impacts performance on the final exam. Note that the Block-1 predictors are also included in this model and control for Q1 performance.

Block 3. In this block, we add the number of Combined Q2 questions that each student answered correctly. If the R^2 for this model is significantly larger than the R^2 for Block-2, it suggests that Combined questions correlate with exam performance over and above the relationship between Peer and exam performance.

Of primary interest is whether the change in R^2 (i.e. the change in variance explained) is significant when moving from the Block-1 model to the Block-2 model, and from the Block-2 model to the Block-3 model. We find that for each of the three regression analyses (entire final exam, code-tracing, and code-writing), each of the changes in R^2 is statistically significant.

These R^2 changes are in Table 1. The first row of the table gives the R^2 values for the final exam as a whole. We see that Q1 explains 21% of the variance, Q2 Peer increases this to 30%, and Q2 Combined increases this further

	Block 1	Block 2	Block 3
Q1 Peer	5.07 (2.01)*	3.87 (1.92)*	2.04 (1.99)
Q1 Combined	7.24 (2.01)*	4.29 (2.04)*	3.53 (2.01)
Q2 Peer		7.62 (1.95)*	4.37 (2.23)
Q2 Combined			6.56 (2.38)*
R^2	0.21	0.30	0.34

*$p < 0.05$

Table 2: Statistical models predicting overall final exam performance

to 34%. In terms of the significance of these R^2 changes, we find that the Block-2 model explains significantly more variance than the Block-1 model (p = .0001), and the Block-3 model explains significantly more variance than the Block-2 model (p = .007). This means that Peer questions correlate with final exam grade (controlling for baseline performance), and Combined questions also correlate with final exam grade (controlling for baseline performance and Peer performance).

The second row of the table gives the R^2 values for code-tracing, and they tell a similar story as the row above. The R^2 values increase from left to right, showing that adding Q2 Peer and then Q2 Combined both increase the proportion of explained variance. Specifically, the Block-2 model explains significantly more variance than the Block-1 model (p = .041), and the Block-3 model explains significantly more variance than the Block-2 model (p = .023).

Finally, the third row of the table gives the R^2 values for code-writing. Again, the Block-2 model explains significantly more variance than the Block-1 model (p = .003), and the Block-3 model explains marginally more variance than the Block-2 model (p = .055).

In Table 2, Table 3, and Table 4, we give the coefficients for each block for the overall, code-tracing, and code-writing regressions, respectively.

One might argue that Combined questions explain new variance (over and above Peer questions) simply because adding the Combined questions more accurately models how much students know. That is, perhaps we could have replaced Combined questions with more Peer questions to see the same effect. While that may be true, we document an interesting finding here to the contrary. If we add the Combined questions to the regressions first, and then add the Peer questions second, the Peer questions do not significantly increase the model R^2, though the first is marginal (p = .053, p = .60, and p = .113, respectively). What this means is that, controlling for the Combined questions, the Peer questions do not further improve our final exam pre-

	Block 1	Block 2	Block 3
Q1 Peer	0.52 (0.28)	0.43 (0.28)	0.21 (0.29)
Q1 Combined	0.73 (0.28)*	0.51 (0.29)	0.42 (0.29)
Q2 Peer		0.56 (0.28)*	0.17 (0.32)
Q2 Combined			0.79 (0.34)*
R^2	0.13	0.16	0.19

$^*p < 0.05$

Table 3: Statistical models predicting final exam code-tracing performance

	Block 1	Block 2	Block 3
Q1 Peer	0.59 (0.28)*	0.45 (0.27)	0.27 (0.29)
Q1 Combined	0.86 (0.28)*	0.53 (0.29)	0.46 (0.29)
Q2 Peer		0.84 (0.28)*	0.51 (0.32)
Q2 Combined			0.66 (0.34)
R^2	0.16	0.22	0.25

$^*p < 0.05$

Table 4: Statistical models predicting final exam code-writing performance

dictions. Combined questions seem to subsume the variance explained by the Peer questions, lending support to the hypothesis that measuring the full PI process in the Combined mode affords more predictive accuracy than does the Q2 measure in the Peer mode.

5. ANCILLARY ANALYSES

As Q1$_{ad}$ could represent learning from peer discussion, we decided to investigate Q1$_{ad}$ as a predictor of final exam performance. Q1$_{ad}$ necessarily conflates active learning with passive peer influence, so we suspected that it would have a weak or null relationship with the final exam (over and above the relationship between Q1 and the final exam). Using the Peer and Combined isomorphic questions as above, we performed three regressions: one for the final exam as a whole, one for the code-tracing question, and one for the code-writing question. In each case, we included Q1 and Q1$_{ad}$ as predictors. Q1, as expected, was a significant predictor in each case, but in no case was Q1$_{ad}$ a significant predictor. That is, Q1$_{ad}$ gives us no predictive power over and above Q1. We then repeated this analysis using the complete dataset of Q1 and Q1$_{ad}$; i.e. using all PI questions from the semester rather than just the Peer and Combined questions. For the final exam as a whole, code-tracing question, and code-writing question, we similarly see no positive and significant increment in R^2 when adding Q1$_{ad}$ to the models.

6. DISCUSSION

In this paper, we have shown that the learning conferred through the PI process is evidenced in higher scores on the final exam. In addition, both key portions of the PI process — peer discussion and instructor intervention — contribute to the relationship between PI correctness and final exam score.

We believe that these findings are both interesting and surprising. PI questions are not like code-writing questions. Indeed, in this PI offering, there were no PI questions that asked students to write code. (The reason is that students used clickers; while such devices allow students to submit numbers or select among multiple choice responses, they do not permit code-entry.) There were questions that asked students to fill-in the missing code and choose the correct code, but this is not the same as writing code from scratch [4]. PI questions are much more similar to code-tracing than they are to code-writing. Yet, we find relationships between PI correctness and both code-tracing and code-writing questions on a final exam. The latter relationship is a welcome surprise to us. We expected PI correctness to correlate with similarly small "chunks" on the final exam, but not to correlate much with code-writing questions. It has been argued that code-writing is at the top of a hierarchy of tasks that includes code-reading, code-tracing, and code-explaining [18]. It is encouraging that PI performance not only correlates with questions similar to PI questions, but also to dissimilar code-writing questions.

One might wonder whether it is simply the number of PI questions being answered, or indeed the number of lectures attended, that is responsible for the observed significant relationships. This is certainly part of the story, because Q1 significantly predicts final exam score. However, recall that we have controlled for Q1 in our regressions, as we investigate the additional predictive power of Q2. That is, for students who answered the same number of Q1 questions correctly, those who answer more Q2 questions correctly do better on the final exam.

Note that in this paper we are not comparing the effectiveness of PI to some other pedagogical method. In particular, we could imagine using a pre-test, post-test design in a lecture course, possibly finding that post-test scores explain final exam variance over and above pre-test scores. In addition, we cannot preclude the possibility that the students that learn from PI are simply those students that would have learned the material regardless of intervention. That is, the increase from Q1 to Q2 could simply represent the learning that would have occurred by these "potential learners" in some other learning setting (such as self-guided reading or lecture). That said, we believe that PI itself is at least partly responsible for the correlations between learning and exam performance. We found that gains from the Combined mode were related to final exam performance once we had controlled for Peer performance. We deem it unlikely that students in a lecture-only or self-guided configuration would exhibit both the peer-based and instructor-based learning gains that are associated with PI. That is, PI seems to aggregate learning from both peers and instructor, and we suggest that this pairing is powerful for both engendering and holding learning.

7. CONCLUSION

Prior Peer Instruction research has demonstrated that students score higher on the group vote than the individual vote, learn from group discussion as measured by isomorphic questions, and outperform lecture-taught peers. In the present work, we complement these findings with an analysis of the relationship between in-class clicker correctness and scores on the final exam. Using isomorphic questions and two modes of question administration, we find that learning from peers and learning from the instructor are each correlated with final exam scores. Furthermore, instructor-based learning is seen even when controlling for peer learning, suggesting additional benefits over and above the peer portion of PI. That is, while prior work has shown that PI students demonstrate learning when tested immediately after discussion, the present work shows that relationships between PI learning and performance are long-lasting. PI learning correlates with both code-tracing and code-writing on exams, even though the latter are far-removed from the types of questions we ask using PI. Future research should continue with a more fine-grained analysis of final exam performance, using exams that tap the wide array of question types that CS1 students should be able to answer. In addition, we urge the community to investigate the mechanisms through which PI learning correlates with final exam performance in order to deepen our understanding of potential causal processes.

8. ACKNOWLEDGMENTS

Thank you to the reviewers for their helpful suggestions. This work was supported in part by NSF grant 1140731.

9. REFERENCES

[1] Peer instruction for computer science. peerinstruction4cs.org, 2013.

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

[3] 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.

[4] P. Denny, A. Luxton-Reilly, and B. Simon. Evaluating a new exam question: Parsons problems. In *Proceedings of the Fourth International Workshop on Computing Education Research*, pages 113–124, 2008.

[5] S. Esper, B. Simon, and Q. Cutts. Exploratory homeworks: An active learning tool for textbook reading. In *Proceedings of the Eighth international Workshop on Computing Education Research*, 2012.

[6] A. Kothiyal, R. Majumdar, S. Murthy, and S. Iyer. Effect of think-pair-share in a large cs1 class: 83% sustained engagement. In *Proceedings of the Ninth international Workshop on Computing Education Research*, 2013.

[7] R. Lister, B. Simon, E. Thompson, J. L. Whalley, and C. Prasad. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *SIGCSE Bulletin*, 38(3):118–122, 2006.

[8] K. Lockwood and R. Esselstein. The inverted classroom and the cs curriculum. In *Proceedings of the 44th ACM technical symposium on Computer Science Education*, pages 113–118, 2013.

[9] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald. Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, 49(8):90–95, 2006.

[10] 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, 2011.

[11] 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*, 2011.

[12] 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, 2013.

[13] 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, 2013.

[14] 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*, 2013.

[15] 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, 2010.

[16] 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, 2013.

[17] M. Smith, W. Wood, K. Krauter, and J. Knight. Combining peer discussion with instructor explanation increases student learning from in-class concept questions. *CBE-Life Sciences Education*, 10(1):55–63, 2011.

[18] A. Venables, G. Tan, and R. Lister. A closer look at tracing, explaining and code writing skills in the novice programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, pages 117–128, 2009.

[19] D. Zingaro. Experience report: Peer instruction in remedial computer science. In *Proceedings of the 22nd World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pages 5030–5035, 2010.

[20] D. Zingaro. Peer instruction contributes to self-efficacy in CS1. In *Proceedings of the 45th ACM technical symposium on Computer Science Education*, 2014.

[21] 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, 2013.

[22] D. Zingaro, A. Petersen, Y. Cherenkova, and O. Karpova. Facilitating code-writing in pi classes. In *Proceedings of the 44th ACM technical symposium on Computer Science Education*, pages 585–590, 2013.

[23] D. Zingaro and L. Porter. Peer instruction in computing: The value of instructor intervention. *Computers & Education*, 71:87–96, 2014.