Experience Report: Peer Instruction in Remedial Computer Science in Proceedings of Ed-Media 2010

Daniel Zingaro
University of Toronto
daniel.zingaro@utoronto.ca

Abstract: We report on our experience implementing Peer Instruction (PI) in a remedial first-year computer science (CS) course in a large North American research university. Quantitative analysis of question data shows modest learning gains following group discussion; qualitative survey data show student endorsement and appreciation for the PI model. We urge the continued study and adoption of PI in CS, and make our reading quizzes and ConcepTests freely available as a takeoff point.

1 Introduction

Peer Instruction (PI) is a classroom management technique that involves a conceptual shift away from a classroom dominated by one-way transmission of knowledge to a more collaborative, active learning environment (Crouch & Mazur 2001). For the past twenty years, PI has contributed to improvements in conceptual understanding among physics students (Crouch & Mazur 2001), and is now being adopted by educators of other sciences (Knight & Wood 2005; Caldwell 2007). We hypothesized that PI might help our students, re-taking a first-year computer science course, make gains in understanding and applying core computer science (CS) concepts. Below, we describe PI, survey the PI-CS literature, motivate our PI research, and provide analysis of correctness and survey data.

2 Background

PI can be succinctly described by a three-step mnemonic, which not only outlines the procedure itself, but also captures several of the core features that contribute to its effectiveness:

ENGAGE. We begin each PI iteration by posing a ConcepTest (Crouch & Mazur 2001) — a multiple-choice question designed both to focus attention on and raise awareness of a key course concept. We strive to create concepTests that encourage reflection over recall, reasoning over rightness, and thinking flexibly over thinking fixedly. Questions recommended in the literature include those that ask students to compare and contrast ideas, extend the context of what they already know, or interpret multiple representations of the same concept (Beatty, Gerace, Leonard, & Dufresne 2006). After individually thinking about and voting on the correct answer (the solo vote), students discuss the question in small groups, reach a consensus, and vote again (the group vote). Students are encouraged to discuss each answer, verbalizing why it is correct or incorrect (Simon, Kohanfars, Lee, Tamayo, & Cutts 2010). The discussion typically results in more students giving the correct answer, and there is evidence that this improvement reflects gains in conceptual understanding rather than peer influence (Smith et al. 2009).

GAUGE. While student voting data can be estimated using flashcards (Lasry 2008), the effectiveness of the "gauge" step is substantially enhanced by using electronic clickers. Clickers afford immediate, accurate vote counts to the instructor, as well as compelling graphical displays for the students. Ideally, each response option for a ConcepTest will correspond to a common student misconception, so that the instructor can identify the range of understandings present among the students (Cutts, Kennedy, Mitchell, & Draper 2004). Seeing the histogram of results, students come to realize that they are not alone in their confusion (Knight & Wood 2005), which may make them more comfortable voicing their concerns in subsequent, class-wide discussion.

AGE. Having engaged students in collaborative debate and gauged their progress and understanding, the instructor leads a class discussion meant to solidify a mature conception of the lessons learned. The most productive discussion is catalyzed by ConcepTests that have multiple defensible answers, highlight misconceptions, or catch application of unjustified assumptions (Beatty et al. 2006). The results of this discussion can be used by the instructor to adapt the lecture in real-time, filling gaps in student knowledge before progressing to the next topic and ConcepTest.

PI takes away from lecture time, thereby reducing the amount of material we can "cover" in class (Knight & Wood 2005; Crouch & Mazur 2001; Burnstein & Lederman 2001). To compensate, it is suggested that instructors require the

completion of textbook reading and an associated reading quiz prior to each class (Crouch & Mazur 2001). In what is referred to as just-in-time teaching (JiTT) (Davis 2009), instructors can use the reading quiz submissions to structure lecture around material found problematic by students.

3 Related Work

The CS education literature contains very few examples of PI usage. In (Pargas & Shah 2006), PI was used to restructure a Data Structures and Algorithms course. Rather than using clickers, students responded to ConcepTests using web-based software from their laptops. Pre-lecture reading quizzes were used both to ask students about the reading and to allow students to voice their own questions or concerns. One or two ConcepTests were used in each 75-minute class period, far below what is common with PI (Crouch & Mazur 2001). This is partially explained by the authors' use of in-class written exercises in addition to ConcepTests, and the time spent on mini-lectures prior to each ConcepTest. Students responded very positively to PI: 96% of students agreed that ConcepTests and the ensuing discussion helped them better understand the material being introduced, and 76% indicated that reading quizzes helped them recognize and focus on difficult concepts.

Simon et al. (2010) provide a compelling discussion of PI in CS1 and CS1.5. Again, the authors deviated from traditional PI practice: reading quizzes and mini-lectures were not used, and the histogram of results was sometimes displayed after the solo vote and before group discussion. Normalized gain — the proportion of students who answered incorrectly in the first vote but correctly in the second – was found to be 41% in CS1 and 35% in CS1.5. Importantly, the authors acknowledge the difficulty of devising CS ConcepTests in the absence of a standardized concept inventory such as that available in physics.

PI has also been used to help students learn from coursework feedback (Cutts, Carbone, & van Haaster 2004) . In order to give formative value to what are otherwise summative assessments, we often provide students with performance-based comments on their assignments and tests. We hope that students will use this feedback to correct misconceptions and recalibrate their learning; unfortunately, we typically do little to promote such reflection. Cutts and colleagues suggest a remedy in the form of an interactive feedback session. While marking term work, the instructor takes note of common misconceptions and generates challenging questions to address each one. The questions are then presented to students, who respond using clickers. PI is used to build consensus for those questions that elicit significant disagreement. Data show that the group vote always improves understanding, and that students find the increased emphasis on feedback to be beneficial for learning.

4 Setting and Motivation

We report on our implementation of PI in a small (40 students), remedial CS1 course for Computer Engineers. All students in the class were unsuccessful in completing a traditional (non-interactive) version of the same course the previous semester, and require this course to continue in the engineering program. We used C to teach the major topics of the course: selection, iteration, functions, arrays, recursion, sorting and memory allocation. We met for three fifty-minute lectures each week for ten weeks, and were supported by weekly practical lab sessions and weekly tutorials.

We are not aware of any literature that describes or legitimizes PI in a remedial CS course. Still, recent findings give hope that PI might be successful with these students (Lasry, Mazur, & Watkins 2008). These authors compared the utility of PI in a two-year college physics course against previous results obtained from Harvard. They found that students with low background knowledge gain as much from PI as students with high background knowledge gain from traditional instruction. If we assume that the students in our CS course lack significant CS background knowledge, we should be able to use PI to teach them much of what the better-prepared students learned in the previous, non-interactive offering of the course. Lasry et al. (2008) also showed that reducing class time spent on problem-solving does not negatively impact students' ability to solve problems algorithmically. We are therefore comfortable moving the solution of larger programming problems from the classroom to the lab and tutorial sessions (described further in Section 5.3).

5 PI Mechanics

We next describe our implementation of PI, dwelling on the benefits of each step for the instructor and the course as a whole.

5.1 Pre-Lecture: Reading Quizzes

Prior to each class, students were required to read one or two sections of our textbook (Carter 2008), and complete a three-question online reading quiz. We used the format suggested in (Crouch & Mazur 2001): we asked two content questions, and then asked students what they found most difficult or confusing about what they had read. 6% of the students' grade was allocated to completion of the quizzes: marks were awarded for effort, not correctness. In general, reading quiz questions were designed to skim the surface of what they had read; deeper application of that material was designated for in-class ConcepTests. Our reading quizzes are available from our resource website (Zingaro 2010); a sampling of questions appears in (Fig. 1).

(After reading about the syntax
of function definitions) What is the return type of `mystery`? What is the type of the parameter accepted by `mystery`?

```
int mystery (double i)
  { ... }
```

(After reading about creating strings) Is the following array a string?

```
char a[3] = {'D', 'A', 'n'};
```

Figure 1: Sample reading quiz questions, emphasizing what students read rather than its application.

Having attempted the course the previous semester, students knew in advance the topics we would cover, and also the topics that they struggled with last time. This sometimes lead students to respond to the third reading question by indicating that they were confused about a topic we had not yet reached! We addressed this several times in lecture, pleading that "we'll cross that bridge when we get there". We also mentioned that they may have found the material difficult last time because of a lack of prerequisite knowledge — exactly the knowledge we were currently obtaining. Yet, we could not completely prevent students from casting a gloomy gaze into the future rather than focusing on the present.

Prior to analyzing the reading quiz responses for each lecture, we created an outline of the topics we felt were most important to understanding the content at hand. Then, through the reading quiz responses, we reshaped our preliminary outline to better correspond with what students told us about their learning. Most frequently, this meant using examples that were meaningful to students, rather than using examples that we thought would be meaningful. For example, in the third lecture on loops, we intended to develop code to print the digits of a number in reverse. However, from the reading quiz responses, it was clear that many students were unable to use loops to print patterns (such as squares and triangles of asterisks). Hoping to capitalize on student interest, we used such examples in class rather than the numeric example we had originally intended.

Sometimes, students expressed misconceptions that were shared by only one or two others. In such situations, we did not feel that it was practical to restructure the lecture in order to address these unique views (Cutts, Kennedy, et al. 2004). Instead, we emailed or met with the student, not only to help them with the specific question at hand, but to make them aware that we carefully read and reflect on everyone's input.

5.2 In-class: PI Proper

We very carefully implemented PI as described in Section 2, including the use of mini-lectures and post-group-vote discussion. 6% of the students' grade was allocated to participation in ConcepTests; again, marks were awarded for effort, rather than correct answers. In fact, we often emphasized that students would learn a lot about their thinking and understanding, even if they ultimately answer the question incorrectly.

ConcepTests were developed using a process similar to that described in (Simon et al. 2010). Based on our prior experience with the course material, and our best efforts to find confusing descriptions or examples in the course textbook, we developed questions and plausible distractors. All of our ConcepTests are available (Zingaro 2010).

One type of question we found particularly effective was the Parson puzzle: an exercise that asks students to reorder given code into a configuration that solves a specified problem (Denny, Luxton-Reilly, & Simon 2008). This type of question has been suggested for inclusion on course exams, because it helps distinguish logical errors from syntactic errors, and allows students to demonstrate their knowledge even if they would not have been able to write the program from scratch. We found discussion more lively when using Parson puzzles. As an example of such an exercise, consider (Fig. 2), used in our seventh class meeting. We preceded this ConcepTest with an explanation of what was being asked, as well as several questions that prompted students to give the output for various test inputs. We wanted the time for the solo vote and group discussion to be spent considering each response, rather than on parsing and understanding what was being asked. The solo vote yielded a correctness rate of 37%, compared to 53% correctness for the group vote. Of course, we cannot discount the possibility that correctness improved simply because the group discussion gave students more time to think about this rather complex exercise. However, we did give extra time for the solo vote (over two minutes as opposed to one minute), terminating it only when we noticed many students stealthily (so they thought) beginning to talk.

Though some questions did evoke purposeful discussion, we had difficulty consistently engaging our students early on, in spite of our efforts to convince them of the utility of PI. As suggested in (Simon et al. 2010), students were encouraged to "talk louder", and were asked to discuss all distractors in addition to the correct answer. Still, we would often pose a ConcepTest, only to have many students fail to discuss at all with their peers. In an act of desperation, we decided to stream a prerecorded auditorium crowd noise throughout the lecture hall, hypothesizing that the room was just too quiet for many students to feel comfortable. Adding this noise, we found many more students were liberated to talk. Further analysis of students' performance on ConcepTests is in section 6.

As was the case with reading quizzes, ConcepTest performance often necessitated a shift in the priority of lecture, and occasionally outright changed the entire lecture plan. At the beginning of our second lecture on recursion, for example,

Write
a program that prompts the user for a sequence of integers, until 0 is entered. The program counts and prints the number of times that consecutive values are equal. For example, if the input is `3 6 7 7 4 4 4 6 0`, the program should print `3`.

```
(I)   int num, old, consec = 0;
(II)  }
      printf ("%d\n", consec);
    (III) while (num > 0) {
    (IV)  old = num;
(V)   if (old == num) consec++;
(VI)  printf ("Enter a number: ");
      scanf ("%d", &num);
```

Which of the following orders is correct?

- A. (I), (III), (IV), (V), (VI), (II)
- B. (I), (VI), (III), (IV), (V), (II)
- C. (I), (VI), (III), (IV), (VI), (V), (II)
- D. (I), (VI), (III), (VI), (IV), (V), (II)

Figure 2: Using a Parson Puzzle as a ConcepTest.

we believed that students would be able to trace a function that included a recursive call in a loop, and so we used such an exercise as our "warm-up" ConcepTest. We were shocked by the results: only 48% of students answered the question correctly. Rather than proceed to a discussion of recursive backtracking as planned, we spent the rest of lecture on two isomorphic ConcepTests generated on-the-fly. The first of these questions again yielded extremely poor performance (21% correctness), indicating that there may not have been sufficient understanding to make PI productive. We therefore carefully traced that question with the class, before offering the second of our on-the-fly ConcepTests. Here, we received a group vote correctness of 69% (after a solo vote correctness of 54%), offering some indication that the extra remediation facilitated considerable understanding.

5.3 Post-class: Tutorial Sessions

The scope of each ConcepTest is necessarily small, drawing focus to one crucial concept at a time. Yet, our students were required to complete five lab assignments throughout the term, each of which required them to write larger programs on their own. To bridge the gap between conceptual understanding and practical implementation, we provided a weekly tutorial session in which to demonstrate the problem-solving process on a larger scale. While the purpose of tutorial may differ from the purpose of lecture, we wanted the collegial PI-facilitated culture to remain. We specifically felt that it was crucial for dialogue begun in lecture to extend to the tutorial, so that misconceptions uncovered in the lecture could be further targeted. One option is to provide the tutorial leader a spreadsheet of the clicker response patterns from lecture, so that they might glean a general understanding of where students struggled (Cutts, Kennedy, et al. 2004). Instead, we decided that the tutorial leader would attend each lecture: raw clicker data is no substitute for directly engaging with, observing and probing students' understanding.

6 Data Analysis

We asked an average of 2.4 questions per fifty-minute class period. The average question correctness for the solo vote was 51%; the average correctness for the group vote was 63%. The stacked histogram of (Fig. 3) contrasts solo vote correctness with group vote correctness. Each box along the X-axis represents one question, and questions were sorted based on solo vote correctness (Simon et al. 2010). Some questions resulted in a decrease in correct responses between the solo and group vote; these are depicted as patterned bars extending below the X-axis.

We calculated the normalized gain (NG) for each question using the technique of (Simon et al. 2010). NG captures the proportion of students who answer incorrectly in the solo vote but answer correctly in the group vote. We found an average NG of 29%.

PI literature suggests that the most fruitful discussion emerges from ConcepTests whose solo vote correctness is between 35% and 70% (Crouch & Mazur 2001). As (Tab. 1) indicates, approximately half of our questions fell in this recommended range. Corroborating a finding from (Simon et al. 2010), the table also shows that we obtained largest average NG among those questions whose solo vote correctness exceeded 70%. Over all questions, we find moderate correlation (Spearman's rank coefficient 0.45) between solo vote correctness and NG.

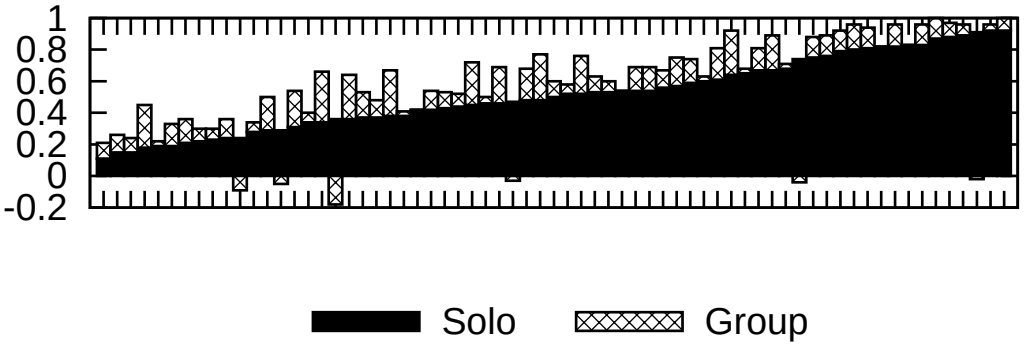

Figure 3: Solo and group vote correctness, per question. Dark bars indicate solo vote correctness, patterned bars indicate group vote correctness.

	$< 35\%$	(35%, 70%)	$> 70\%$
Proportion of questions	25%	51%	24%
Average NG	13%	26%	53%

Table 1: Proportion and average NG for questions having solo vote correctness less than 35%, between 35% and 70%, and greater than 70 %.

Statement	**A**	**B**	**C**	**D**	**E**	**F**
Thinking about clicker questions on my own, before discussing with people around me, helped me learn the course material.	50%	31%	19%	0%	0%	0%
Pre-lecture reading quizzes helped me recognize what was difficult in the reading.	25%	31%	38%	0%	6%	0%
Discussing course topics with my seatmates in class helped me better understand the course material.	38%	44%	19%	0%	0%	0%
The immediate feedback from clickers helped me focus on weaknesses in my understanding of the course material.	50%	38%	6%	6%	0%	0%
Clickers are an easy-to-use class collaboration tool.	81%	13%	6%	0%	0%	0%
I recommend that other instructors use our approach (reading quizzes, clickers, in-class discussion) in their courses.	94%	0%	0%	6%	0%	0%

Table 2: End-of-term survey data ($N = 16$). Each question was answered on a scale from A (agree very strongly) to F (disagree very strongly).

It is admittedly difficult to ascribe meaning to our average NG. On the one hand, it is substantially lower than the 41% NG found in (Simon et al. 2010) for a CS1 course covering similar material. On the other hand, our study is the first in CS to use PI in a remedial course, and so it may be misleading to directly compare NGs. Our students very likely have more deeply rooted misconceptions about the subject matter, proceed through the course under the influence of their negative first attempt, and have different motivations and attitudes compared to students considered in other PI studies.

We administered an end-of-term survey, adapted from (Pargas & Shah 2006), to help us better appreciate students' attitudes regarding the effectiveness and usefulness of PI. Students responded to each of the questions with a rating from **A** (agree very strongly) to **F** (disagree very strongly). The results are overwhelmingly positive (Tab. 2).

7 Recommendations and Future Work

Based on our experience, we offer several recommendations for CS educators considering the use of PI.

- Carefully gauge the class' alertness and interest before presenting a ConcepTest in the last few minutes of class. Each time we tried a last-minute ConcepTest, it failed as a learning aid because students did not "have anything left".
- Resist the temptation to over-prepare for lecture (Davis 2009). Early on, we made the mistake of carefully preparing sets of slides, only to throw many of them away when we realized they did not correspond with what students required from class.
- Permeate PI culture throughout tutorials and practical labs, in order to provide continuity across the course. Do not assume that students will identify their difficulties to tutorial and lab leaders.

- If students are hesitant to talk out loud, provide encouragement, and explain your rationale for using PI. If a classroom speaker system is installed, stream auditorium ambiance to help students get louder.
- Be prepared for a small percentage of students to occasionally but intentionally choose the incorrect response (such as choosing option E for a ConcepTest that has only four responses). It may be that these students did not know the correct answer, or that they knew they were getting marks in spite of answering incorrectly.

We will continue to refine our ConcepTests, working on ways to increase solo vote correctness without sabotaging the challenging nature of the questions. We will also further examine the applicability of Parson puzzles, and increase frequency of usage if they prove to be as useful as we imagine. Finally, we will analyze reading quiz response data so as to better understand the level of student effort elicited, possible links between reading quiz response rate and class preparedness, and types of questions that best scaffold the reading.

8 Conclusion

We have found modest normalized learning gains and obtained positive student feedback in response to our implementation of PI in a remedial CS course. We are particularly pleased by our students' positive attitudes to a course that they would rather not be taking. We hope that our implementation recommendations, reading quizzes and ConcepTests (Zingaro 2010) will help other CS instructors use PI, and will speed the creation and validation of further PI materials.

References

(Beatty, Gerace, Leonard, & Dufresne 2006) Beatty, I. D., Gerace, W. J., Leonard, W. J., & Dufresne, R. J. (2006). Designing effective questions for classroom response system teaching. *American Journal of Physics*, *74*, 31-39.

(Burnstein & Lederman 2001) Burnstein, R. A., & Lederman, L. M. (2001). Using wireless keypads in lecture classes. *The Physics Teacher*, *39*, 8-11.

(Caldwell 2007) Caldwell, J. E. (2007). Clickers in the large classroom: Current research and best-practice tips. *CBE-Life Sciences Education*, *6*, 9-20.

(Carter 2008) Carter, J. (2008). *Introduction to computer science using c*. McGraw-Hill Ryerson.

(Crouch & Mazur 2001) Crouch, C. H., & Mazur, E. (2001). Peer instruction: Ten years of experience and results. *American Journal of Physics*, *69*, 970-977.

(Cutts, Carbone, & van Haaster 2004) Cutts, Q., Carbone, A., & van Haaster, K. (2004). *Using an electronic voting system to promote active reflection on coursework feedback.* International conference on computers in education. `www.dcs.gla.ac.uk/~quintin/papers/ICCE04QC.pdf` (accessed March 27, 2010).

(Cutts, Kennedy, Mitchell, & Draper 2004) Cutts, Q., Kennedy, G., Mitchell, C., & Draper, S. (2004). *maximising dialogue in lectures using group response systems.* 7th IASTED international conference on computers and advanced technology in education. `www.dcs.gla.ac.uk/~quintin/papers/cate2004.pdf` (accessed March 27, 2010).

(Davis 2009) Davis, J. (2009). Experiences with just-in-time teaching in systems and design courses. In *SIGCSE '09: Proceedings of the 40th SIGCSE technical symposium on computer science education* (p. 71-75). New York, NY, USA: ACM.

(Denny, Luxton-Reilly, & Simon 2008) Denny, P., Luxton-Reilly, A., & Simon, B. (2008). Evaluating a new exam question: Parsons problems. In *ICER '08: Proceedings of the fourth international workshop on computing education research* (p. 113-124). New York, NY, USA: ACM.

(Knight & Wood 2005) Knight, J. K., & Wood, W. B. (2005). Teaching more by lecturing less. *Cell Biology Education*, *4*, 298-310.

(Lasry 2008) Lasry, N. (2008). Clickers or flashcards: Is there really a difference? *The Physics Teacher*, *46*, 242-244.

(Lasry, Mazur, & Watkins 2008) Lasry, N., Mazur, E., & Watkins, J. (2008). Peer instruction: From harvard to the two-year college. *American Journal of Physics*, *76*, 1066-1069.

(Pargas & Shah 2006) Pargas, R. P., & Shah, D. M. (2006). Things are clicking in computer science courses. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on computer science education* (p. 474-478). New York, NY, USA: ACM.

(Simon, Kohanfars, Lee, Tamayo, & Cutts 2010) Simon, B., Kohanfars, M., Lee, J., Tamayo, K., & Cutts, Q. (2010). Experience report: Peer instruction in introductory computing. In *SIGCSE '10: Proceedings of the 41st SIGCSE technical symposium on computer science education* (p. 341-345). New York, NY, USA: ACM.

(Smith et al. 2009) Smith, M. K., Wood, W. B., Adams, W. K., Wieman, C., Knight, J. K., Guild, N., et al. (2009). Why peer discussion improves student performance on in-class concept questions. *Science*, *323*, 122-124.

(Zingaro 2010) Zingaro, D. (2010). *Pi-cs resource page.* `www.danielzingaro.com/pics.php`.

Acknowledgments We thank Beth Simon for her encouragement in this work, and for introducing us to PI and its applicability to CS. Thanks also to Jim Clarke for his support throughout this study.